emnlp2016

November 1–5, 2016 Austin, Texas, USA

Conference on Empirical Methods in Natural Language Processing

CONFERENCE PROCEEDINGS

www.emnlp2016.net

EMNLP 2016 gratefully acknowledges the following sponsors for their support:

Platinum

Gold

Silver

Bronze

Student Volunteer Sponsor

Order copies of this and other ACL proceedings from:

Curran Associates
57 Morehouse Lane
Red Hook, New York 12571
USA
Tel: +1-845-758-0400
Fax: +1-845-758-2633
curran@proceedings.com

ISBN 978-1-945626-25-8

Table of Contents

Preface by the General Chair

October 17, 2016

Welcome to the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016) in Austin, Texas, USA!

EMNLP is annually organized by SIGDAT, the Association for Computational Linguistics' special interest group on linguistic data and corpus-based approaches to NLP. At EMNLP 2016, one of the top tier conferences in Natural Language Processing (NLP), we have witnessed how our field thrives. This is not only reflected in the number of paper submissions but also in the number of sponsors. The number of long paper submissions has increased 14.5% over that of 2015. This year, we also have seen a record high number of sponsors in EMNLP history. We're honored and grateful to have Amazon, Baidu, Google and Grammarly as the Platinum Sponsors, Bloomberg, Citadel, eBay, Facebook, IBM Research, Maluuba and Microsoft as the Gold Sponsors, AI@ISI as the Silver Sponsor, Nuance, VoiceBox and Yandex as the Bronze Sponsors. We also have Oracle as the Student Volunteer Sponsor.

A large number of people worked hard to bring this annual meeting to fruition, to whom I'm very grateful. Program Chairs, **Kevin Duh** and **Xavier Carreras**, the Area Chairs, reviewers, best paper committee members put in an immense amount of work to develop the technical program. Tutorial Chairs, **Bishan Yang** and **Rebecca Hwa**, Workshop Chairs, **Annie Louis** and **Greg Kondrak** conducted a competitive selection process in collaboration with NAACL and ACL to select 6 tutorials and 8 workshops. Sponsorship Chairs, **Michel Galley**, **Hang Li** (ACL International Sponsorship Committee Representative for EMNLP) did an excellent job to attract the record number of sponsors. Publication Chairs, **Siddharth Patwardhan**, **Daniele Pighin** (advisor), Handbook Chair, **Swapna Somasundaran** worked with a very tight schedule to assemble the proceedings, C4Me Mobile app, and handbooks. Publicity Chair, **Saif M. Mohammad** disseminated the call for papers, call for participation and other announcements in a timely manner. Webmaster, **Jackie C.K. Cheung** kept the website updated all the time, providing a professional outlook of the conference. Student Scholarship Chair and Student Volunteer Coordinator, **Vincent Ng**, played two critical roles, managing the NSF and SIGDAT scholarship, and the review of applications, coordinating the student volunteers to support the conference. SIGDAT Secretary, **Chris Callison-Burch** acted as the liaison between SIGDAT and the conference organizers. He is always available to provide great suggestions.

As usual, the conference cannot be done without Local Arrangements Chair, **Priscilla Rasmussen**, who single-handedly took care of all conference logistics. I would like to mention that I benefited greatly from last year's General Chair, **Lluís Màrquez**, for the monthly progress reports and other valuable experience. We are also grateful to the invited speakers, **Christopher Potts**, **Andreas Stolcke** and **Stefanie Tellex** who will share with us their exciting research.

I really appreciate the trust from SIGDAT officers, including previous secretary, **Noah Smith**, to coordinate the conference as the General Chair.

Finally, I'd like to thank all the authors and attendees. Your participation made a difference to the conference. I hope that you have an enjoyable and productive time at Austin. My best wishes for a successful conference!

Jian Su
EMNLP 2016 General Chair

Preface by the Program Committee Co-Chairs

October 17, 2016

Welcome to the 2016 Conference on Empirical Methods in Natural Language Processing! This year we received 1,087 valid submissions, of which 687 were long papers and 400 were short papers. We accepted 177 long papers (25.8% acceptance rate) and 87 short papers (21.8% acceptance rate), for a total of 264 papers and an overall acceptance rate of 24.3%.

The technical program at EMNLP 2016 consists of a total of 273 papers, including 9 journal papers accepted by the Transactions of ACL. We have structured the conference into three parallel oral sessions in the day and two poster sessions in the evening. Borrowing from recent NAACL conference innovations, we also run poster spotlight sessions (also called *HMM: Half-Minute Madness*[1]), where poster presenters of long papers have 30 seconds and one slide to advertise their work. Poster sessions are becoming larger due to the rapid growth in our field, and we believe it is important to ensure that all papers receive the exposure they deserve.

We are excited and grateful to have three distinguished speakers for our invited keynote talks. Christopher Potts (Stanford University) will present recent advances in rational speech acts and pragmatics. Andreas Stolcke (Microsoft Research) will talk about the challenges and opportunities in human-human-machine dialog. Stefanie Tellex (Brown University) will discuss novel methods and frameworks for enabling human-robot collaboration. We think that these are exciting research areas that can potentially impact—and be impacted by—the EMNLP community in the near future. We look forward to their keynotes and the conversations afterwards.

The program committee includes 823 primary reviewers and 99 secondary reviewers. The committee was structured into 12 thematic areas, handled by 41 area chairs. We are grateful to all program committee members for their effort and dedication during our tight reviewing schedule; without them we cannot create a strong high-quality program. We are also thankful for all authors who submitted papers, which overall cover a diverse range of topics.

Best paper awards were organized around three categories: best paper, best short paper, and best resource paper. The latter category was introduced at EMNLP 2015. Since resources have become central for scientific progress in our field, we would like this category of award to become a standard. The selection process was bottom-up: reviewers and area chairs suggested candidates, which were short-listed by us program chairs. Then, for each category we created a committee of experts to discuss the papers in depth, and we chaired the committees.

For **best paper**, the committee members were Stephen Clark, Hal Daumé III, Chris Dyer, and Julia

[1] Neologism coined by Joel Tetreault, our HMM chair.

	Long	Short	Total
Initial submissions	747	438	1,185
Withdrawn or rejected without review	60	38	98
Submissions reviewed	687	400	1,087
Submissions accepted	177	87	264
Acceptance rate	25.76%	21.75%	24.29%
TACL papers	9	0	9
Papers at EMNLP 2016	186	87	273
Oral talks	87	22	109
Poster presentations	99	65	164

Table 1: Submission statistics of EMNLP 2016

Hockenmaier. The committee selected two best long papers:

- Best Paper: *Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning*, by Karthik Narasimhan, Adam Yala and Regina Barzilay.
- Best Paper: *Global Neural CCG Parsing with Optimality Guarantees*, by Kenton Lee, Mike Lewis and Luke Zettlemoyer.

In addition, two papers were given an honorable mention for best paper:

- Honorable Mention for Best Paper: *Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles*, by James Cross and Liang Huang.
- Honorable Mention for Best Paper: *Sequence-to-Sequence Learning as Beam-Search Optimization*, by Sam Wiseman and Alexander M. Rush.

For **best short paper**, the committee had Stefan Riezler, Anoop Sarkar, and Noah Smith, and the award went to:

- Best Short Paper: *Learning a Lexicon and Translation Model from Phoneme Lattices*, by Oliver Adams, Graham Neubig, Trevor Cohn, Steven Bird, Quoc Truong Do and Satoshi Nakamura.

For **best resource paper**, the committee consisted of Eneko Agirre, Mirella Lapata, and Sebastian Riedel, and the award went to:

- Best Resource Paper: *SQuAD: 100,000+ Questions for Machine Comprehension of Text*, by Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev and Percy Liang.

We are grateful to the many people who helped us at various stages of the program preparation. In particular, we would like to thank:

- Jian Su and Chris Callison-Burch, who gave us advice and support throughout the whole process, not only in their capacity as program chairs of EMNLP 2015, but also as general chair of EMNLP 2016 (Jian) and SIGDAT secretary-treasurer (Chris).
- The 41 area chairs, whose expertise and dedication we relied on heavily. They selected reviewers, coordinated the review process, led discussions, and made recommendations. We owe you a favor: Yoav Artzi, Tim Baldwin, Guillaume Bouchard, Nate Chambers, Kyunghyun Cho, Michael Collins, John DeNero, Georgiana Dinu, Sanja Fidler, Alex Fraser, Kuzman Ganchev, Ed Grefenstette, Julia Hockenmaier, Dirk Hovy, Liang Huang, Ruihong Huang, Min-Yen Kan, Daisuke Kawahara, Yang Liu, Bing Liu, André F.T. Martins, Saif Mohammad, Ray Mooney, Smaranda Muresan, Preslav Nakov, Vivi Nastase, Ariadna Quattoni, Laura Rimell, Eric Ringger, Alan Ritter, Brian Roark, David Smith, Manfred Stede, Suzanne Stevenson, Michael Strube, Joel Tetreault, Lucy Vanderwende, Dekai Wu, Wei Xu, Scott Wen-Tau Yih, and Geoff Zweig.
- Priscilla Rasmussen, our local organizer who performed amazing feats to make everything work.
- Siddharth Patwardhan and Daniele Pighin, the publication chairs.
- Swapna Somasundaran, handbook chair.
- Joel Tetreault, Brendan O'Connor, and Courtney Napoles for organizing and chairing *the HMM sessions*.
- The session chairs: Regina Barzilay, Alexandra Birch, Phil Blunsom, Yejin Choi, Ido Dagan, Marie-Catherine de Marneffe, Katrin Erk, Pascale Fung, Alona Fyshe, Rebecca Hwa, Heng Ji, Diane Litman, Yang Liu, Lluís Màrquez, André F.T. Martins, Kathy McKeown, Raymond Mooney, Preslav Nakov, Hinrich Schütze, Thamar Solorio, Hiroya Takamura, Kristina Toutanova, Bonnie Webber, and Wei Xu.
- Jackie C.K. Cheung, who maintained the EMNLP 2016 website with up-to-date information.
- Yejin Choi, who kept us connected with the ACL Exec.
- Kristina Toutanova and Lillian Lee, who helped us regarding TACL papers.
- Janyce Wiebe, Michael Strube, and Anoop Sarkar, who provided detailed advice about chairing a program committee of a large conference at the initial planning stages of the process.
- Ani Nenkova, Owen Rambow, Katrin Erk and Noah Smith (program co-chairs of NAACL and ACL this year), with which we coordinated several aspects of the major conferences this year.
- The Softconf support team, Rich Gerber and Paolo Gai, who assisted us in using the Start Conference Manager.

On behalf of all attendees at the conference, we would also like to acknowledge the generosity of our sponsors: Amazon, Baidu, Google, Grammarly, Bloomberg, Citadel, eBay, Facebook, IBM Research, Maluuba, Microsoft, AI@ISI, Nuance, VoiceBox, Yandex, and Oracle.

Chairing the program committee of EMNLP has been a great honor and a rich scientific experience. We are grateful to SIGDAT for giving us this opportunity. And we hope that you will find the program as exciting and enjoyable as we do!

Xavier Carreras and Kevin Duh
EMNLP 2016 Program Committee Co-Chairs

Organizing Committee

General Chair

Jian Su, Institute for Infocomm Research (I2R)

Program Co-chairs

Kevin Duh, Johns Hopkins University
Xavier Carreras, Xerox Research Centre Europe

Workshop Co-chairs

Annie Louis, University of Essex
Greg Kondrak, University of Alberta

Tutorial Co-chairs

Bishan Yang, Carnegie Mellon University
Rebecca Hwa, University of Pittsburgh

Publication Co-chairs

Siddharth Patwardhan, Apple
Daniele Pighin (advisor), Google

Publicity Chairs

Saif Mohammad, National Research Council Canada

Handbook Chair

Swapna Somasundaran, Educational Testing Services

Website Chair

Jackie C.K. Cheung, McGill University

Sponsorship Team

Michel Galley, Microsoft Research
Hang Li (ISC Representative for EMNLP), Huawei Technologies

Student Scholarship Chair and Student Volunteer Co-ordinator

Vincent Ng, University of Texas at Dallas

SIGDAT Liason

Chris Callison-Burch, University of Pennsylvania

Local Arrangements Chair

Priscilla Rasmussen, ACL Business Manager

Program Committee

Program Co-chairs

Xavier Carreras, Xerox Research Centre Europe
Kevin Duh, Johns Hopkins University

Area chairs

Information Extraction, Information Retrieval, and Question Answering
Nathanael Chambers, US Naval Academy
Ruihong Huang, Texas A&M University
Min-Yen Kan, National University of Singapore
Alan Ritter, The Ohio State University
Scott Wen-tau Yih, Microsoft Research

Language and Vision
Sanja Fidler, University of Toronto
Julia Hockenmaier, University of Illinois Urbana-Champaign

Linguistic Theories and Psycholinguistics
Suzanne Stevenson, University of Toronto

Machine Learning
Guillaume Bouchard, University College London
Kyunghyun Cho, New York University
Kuzman Ganchev, Google
Ariadna Quattoni, Xerox Research Centre Europe
Eric Ringger, Facebook and Brigham Young University

Machine Translation and Multilinguality
John DeNero, UC Berkeley
Alexander Fraser, Ludwig-Maximilians-Universität München
Yang Liu, Tsinghua University
Dekai Wu, HKUST

Segmentation, Tagging, and Parsing
Michael Collins, Columbia University and Google
Liang Huang, Oregon State University
Daisuke Kawahara, Kyoto University
André F.T. Martins, Unbabel and Instituto de Telecomunicações

Semantics

Yoav Artzi, Cornell University
Georgiana Dinu, IBM Watson
Edward Grefenstette, Google DeepMind
Raymond Mooney, University of Texas at Austin
Laura Rimell, University of Cambridge

Sentiment Analysis and Opinion Mining

Dirk Hovy, University of Copenhagen
Bing Liu, University of Illinois at Chicago
Saif Mohammad, National Research Council Canada

Social Media and Computational Social Science

Timothy Baldwin, The University of Melbourne
Smaranda Muresan, Columbia University

Spoken Language Processing

Brian Roark, Google Inc.
Geoffrey Zweig, Microsoft Research

Summarization, Generation, Discourse, Dialogue

Manfred Stede, University of Potsdam
Michael Strube, Heidelberg Institute for Theoretical Studies
Lucy Vanderwende, Microsoft Research
Wei Xu, The Ohio State University

Text Mining and NLP Applications

Preslav Nakov, Qatar Computing Research Institute
Vivi Nastase, University of Heidelberg
David Smith, Northeastern University
Joel Tetreault, Grammarly

Primary Reviewers

Balamurali A R; Omri Abend; Amjad Abu-Jbara; Željko Agić; Eneko Agirre; Julien Ah-Pine; Lars Ahrenberg; Salah Ait-Mokhtar; Yaser Al-Onaizan; Mohammed Alam; Chris Alberti; Nikolaos Aletras; Jan Alexandersson; Enrique Alfonseca; Tamer Alkhouli; Miltiadis Allamanis; Alexandre Allauzen; Yasemin Altun; Carlos Alzate; Bharat Ram Ambati; Hadi Amiri; Waleed Ammar; Daniel Andor; Jacob Andreas; Nicholas Andrews; Yuki Arase; Ron Artstein; Ramón Astudillo; Giuseppe Attardi; Isabelle Augenstein; Eleftherios Avramidis; Amittai Axelrod;

Dzmitry Bahdanau; JinYeong Bak; Alexandra Balahur; Kalika Bali; Borja Balle; Miguel Ballesteros; David Bamman; Mohit Bansal; Libby Barak; Chitta Baral; Regina Barzilay; Riza Theresa Batista-Navarro; Daniel Bauer; Timo Baumann; Daniel Beck; Beata Beigman Klebanov; Lisa Beinborn; Núria Bel; David Belanger; Kedar Bellare; I. Beltagy; Anja Belz; Fabrício Benevenuto; Jonathan Berant; Taylor Berg-Kirkpatrick; Nicola Bertoldi; Laurent Besacier; Steven Bethard; Rahul Bhagat; Suma Bhat; Chris Biemann; Or Biran; Alexandra Birch; Yonatan Bisk; John Blitzer; Michael Bloodgood; Gemma Boleda; Kalina Bontcheva; Johan Bos; Matko Bosnjak; Jan A. Botha; Houda Bouamor; Samuel Bowman; Fabienne Braune; Chris Brew; Chris Brockett; Julian Brooke; Caroline Brun; William Bryce; Paul Buitelaar; Florin Bulgarov; Razvan Bunescu; David Burkett; Jill Burstein; Bill Byrne;

Elena Cabrio; Aoife Cahill; Chris Callison-Burch; Berkant Barla Cambazoglu; Erik Cambria; Liangliang Cao; Ziqiang Cao; Fabienne Cap; Cornelia Caragea; Claire Cardie; Marine Carpuat; John Carroll; Tommaso Caselli; Vittorio Castelli; Giuseppe Castellucci; Mauro Cettolo; Joyce Chai; Yee Seng Chan; Muthu Kumar Chandrasekaran; Angel Chang; Jonathan Chang; Jason Chang; Kai-Wei Chang; Ming-Wei Chang; Wanxiang Che; Ciprian Chelba; Boxing Chen; Danqi Chen; Bin Chen; Zhiyuan Chen; Hsin-Hsi Chen; John Chen; Tao Chen; Wenliang Chen; Yun-Nung Chen; Colin Cherry; Sean Chester; Jackie Chi Kit Cheung; David Chiang; Martin Chodorow; Do Kook Choe; Eunsol Choi; Jinho D. Choi; Yejin Choi; Monojit Choudhury; Christos Christo-doulopoulos; Grzegorz Chrupała; Mark Cieliebak; Philipp Cimiano; Stephen Clark; Ann Clifton; Shay B. Cohen; Trevor Cohn; Nigel Collier; Miriam Connor; Paul Cook; Ryan Cotterell; Benoit Crabbé; Danilo Croce;

Jennifer D'Souza; Walter Daelemans; Ido Dagan; Lena Dankin; Dipanjan Das; Pradipto Das; Rajarshi Das; Hal Daumé III; Munmun De Choudhury; Adrià de Gispert; Marie-Catherine de Marneffe; Gerard de Melo; Judith Degen; Felice Dell'Orletta; Dina Demner-Fushman; Steve DeNeefe; Pascal Denis; Michael Denkowski; Ludovic Denoyer; Anoop Deoras; Tejaswini Deoskar; Leon Derczynski; Aliya Deri; Ann Devitt; Jacob Devlin; Giuseppe Di Fabbrizio; Mona Diab; Laura Dietz; Jesse Dodge; Qing Dou; Doug Downey; Eduard Dragut; Mark Dras; Markus Dreyer; Gregory Druck; Nan Duan; Nadir Durrani; Greg Durrett; Chris Dyer; Marc Dymetman;

Richard Eckart de Castilho; Yo Ehara; Vladimir Eidelman; Jacob Eisenstein; Jason Eisner; Ali Elkahky; Desmond Elliott; Micha Elsner; Messina Enza; Keelan Evanini; Stefan Evert;

Benamara Farah; Noura Farra; Manaal Faruqui; Benoit Favre; Geli Fei; Anna Feldman; Paul Felt; Yansong Feng; Raquel Fernandez; Daniel Fernández-González; Michele Filannino; Simone Filice; Katja Filippova; Nicholas FitzGerald; Jeffrey Flanigan; Radu Florian; José A. R. Fonollosa; Mikel Forcada; Victoria Fossum; George Foster; Anette Frank; Stefan L. Frank; Daniel Fried; Annemarie Friedrich; Mario Fritz; Hagen Fuerstenau; Atsushi Fujii; Alona Fyshe;

Nuria Gala; Matthias Gallé; Michel Galley; Michael Gamon; Juri Ganitkevitch; Jianfeng Gao; Wei Gao; Claire Gardent; Matt Gardner; Dan Garrette; Milica Gasic; Tao Ge; Spandana Gella; Kallirroi Georgila; Ulrich Germann; George Giannakopoulos; Daniel Gildea; Jennifer Gillenwater; Daniel Gillick; Kevin Gimpel; Filip Ginter; Dimitra Gkatzia; Goran Glavaš; Amir Globerson; Koldo Gojenola; Yoav Goldberg; Dan Goldwasser; Carlos Gómez-Rodríguez; Graciela Gonzalez; Edgar Gonzàlez Pellicer; Matthew R. Gormley; Amit Goyal; Joao Graca; Yvette Graham; Edouard Grave; Christopher Gravier; Spence Green; Stephan Greene; Ralph Grishman; Cyril

Grouin; Jiafeng Guo; Hongyu Guo; Weiwei Guo; Sonal Gupta; Iryna Gurevych; Andreas Guta;

Ivan Habernal; Ben Hachey; Barry Haddow; Udo Hahn; Hannaneh Hajishirzi; Dilek Hakkani-Tur; John Hale; David Hall; Keith Hall; Shuguang Han; Xianpei Han; Greg Hanneman; Sanda Harabagiu; Christian Hardmeier; Saša Hasan; Kazuma Hashimoto; Hua He; Xiangnan He; He He; Yifan He; Luheng He; Xiaodong He; Yulan He; Kenneth Heafield; Michael Heilman; James Henderson; John Henderson; Matthew Henderson; Aurélie Herbelot; Derrick Higgins; Graeme Hirst; Hieu Hoang; Kristy Hollingshead; Liangjie Hong; Matthew Honnibal; Ales Horak; Takaaki Hori; Veronique Hoste; Yufang Hou; Chun-Nan Hsu; Minlie Huang; Yi-Ting Huang; Fei Huang; Heyan Huang; Shujian Huang; Xuanjing Huang; Zhongqiang Huang; Matthias Huck; Rebecca Hwa;

Gonzalo Iglesias; Iustina Ilisei; Diana Inkpen; Radu Tudor Ionescu; Abe Ittycheriah; Mohit Iyyer;

Guillaume Jacquet; Kokil Jaidka; Peter Jansen; Laura Jehl; Yangfeng Ji; Ping Jian; Wenbin Jiang; Zhiwei Jiang; Jing Jiang; Anders Johannsen; Marcin Junczys-Dowmunt; David Jurgens;

Hetunandan Kamichetty; Hiroshi Kanayama; Justine Kao; Damianos Karakos; Saurabh Kataria; Frank Keller; Mitesh M. Khapra; Chloé Kiddon; Douwe Kiela; Jin-Dong Kim; Suin Kim; Tracy Holloway King; Svetlana Kiritchenko; Jamie Ryan Kiros; Sigrid Klerke; Roman Klinger; Alistair Knott; Ekaterina Kochmar; Tomáš Kočiský; Philipp Koehn; Mamoru Komachi; Grzegorz Kondrak; Lingpeng Kong; Ioannis Konstas; Georgios Kontonatsios; Anna Korhonen; Yannis Korkontzelos; Leila Kosseim; Zornitsa Kozareva; Martin Krallinger; Jayant Krishnamurthy; Anastasia Krithara; Canasai Kruengkrai; Germán Kruszewski; Lun-Wei Ku; Roland Kuhn; Shankar Kumar; Jonathan K. Kummerfeld; Tsung-Ting Kuo; Sadao Kurohashi; Nate Kushman; Tom Kwiatkowski;

Igor Labutov; Patrik Lambert; Vasileios Lampos; Man Lan; Phillippe Langlais; Mirella Lapata; Shalom Lappin; Angeliki Lazaridou; Nevena Lazic; Joseph Le Roux; John Lee; Sungjin Lee; Kenton Lee; Lung-Hao Lee; Tao Lei; Alessandro Lenci; Gregor Leusch; Omer Levy; Roger Levy; Mike Lewis; Jiwei Li; Chen Li; Fangtao Li; Huayi Li; Junyi Jessy Li; Haibo Li; Jing Li; Qi Li; Sujian Li; Mu Li; Yanen Li; Zhenghua Li; Maria Liakata; Victoria Lin; Xiao Ling; Wang Ling; Tal Linzen; Marina Litvak; Fei Liu; Qian Liu; Qun Liu; Kang Liu; Shujie Liu; Yang Liu; Yiqun Liu; Nikola Ljubešić; Edward Loper; Adam Lopez; Zhengdong Lu; Bin Lu; Wei Lu; Marco Lui; Michal Lukasik; Xiaoqiang Luo; Zhunchen Luo; Minh-Thang Luong; Teresa Lynn;

Ji Ma; Yanjun Ma; Klaus Macherey; Wolfgang Macherey; Nitin Madnani; Walid Magdy; Pierre Magistry; Shervin Malmasi; Gideon Mann; Christopher D. Manning; Mehdi Manshadi; Saab Mansour; Amin Mantrach; Daniel Marcu; Anna Margolis; Lluís Màrquez; Bruno Martins; Yuval Marton; Sebastian Martschat; Yuji Matsumoto; Austin Matthews; Arne Mauser; Jonathan May; Diana Maynard; Andrew McCallum; David McClosky; Kathy McKeown; Louise McNally; Beata Megyesi; Yashar Mehdad; Yelena Mejova; Pablo Mendes; Arul Menezes; Paola Merlo; Florian Metze; Haitao Mi; Yishu Miao; Claudiu Mihăilă; Rada Mihalcea; David Mimno; Bonan Min; Shachar Mirkin; Seyed Abolghasem Mirroshandel; Paramita Mirza; Dipendra Misra; Margaret Mitchell; Makoto Miwa; Samaneh Moghaddam; Mitra Mohtarami; Karo Moilanen; Manuel Montes; Taesun Moon; Véronique Moriceau; Alessandro Moschitti; Nasrin Mostafazadeh; Dragos Munteanu; Yugo Murawaki;

Vinita Nahar; Seiichi Nakagawa; Courtney Napoles; Jason Naradowsky; Shashi Narayan; Tahira Naseem; Borja Navarro; Roberto Navigli; Adeline Nazarenko; Mark-Jan Nederhof; Arvind Neelakantan; Sapna Negi; Aida Nematzadeh; Graham Neubig; Guenter Neumann; Hwee Tou Ng; Jun-Ping Ng; Vincent Ng; Viet-An Nguyen; Jian-Yun Nie; Jan Niehues; Zheng-Yu Niu; Pierre Nugues;

Diarmuid Ó Séaghdha; Brendan O'Connor; Stephan Oepen; Kemal Oflazer; Alice Oh; Naoaki Okazaki; Tsuyoshi Okita; Miles Osborne; Mari Ostendorf;

Ulrike Padó; Sebastian Padó; Muntsa Padró; Alexis Palmer; Martha Palmer; Alessio Palmero Aprosio; Sinno Jialin Pan; Denis Paperno; Aasish Pappu; Ankur Parikh; Devi Parikh; Patrick Paroubek; Michael J. Paul; Adam Pauls; Ellie Pavlick; Lisa Pearl; Andreas Peldszus; Hao Peng; Gerald Penn; Julien Perez; Verónica Pérez-Rosas; Bryan Perozzi; Jan-Thorsten Peter; Slav Petrov; Nghia The Pham; Peter Phandi; Olivier Pietquin; Manfred Pinkal; Yuval Pinter; Emily Pitler; Barbara Plank; Massimo Poesio; Maja Popović; Fred Popowich; Matt Post; Vinodkumar Prabhakaran; John Prager; Alessandro Presta; Prokopis Prokopidis; Emily Prud'hommeaux; Matthew Purver;

Ashequl Qadir; Longhua Qian; Xian Qian; Lu Qin; Long Qiu; Lizhen Qu; Chris Quirk;

Ella Rabinovich; Will Radford; Afshin Rahimi; Altaf Rahman; Nazneen Fatema Rajani; Rafal Rak; Bhuvana Ramabhadran; Vivek Kumar Rangarajan Sridhar; Ari Rappoport; Mohammad Sadegh Rasooli; Siva Reddy; Roi Reichart; Ehud Reiter; Xiang Ren; Matthew Richardson; Sebastian Riedel; Mark Riedl; Jason Riesa; Stefan Riezler; Ellen Riloff; Fabio Rinaldi; Kirk Roberts; Tim Rocktäschel; Marcus Rohrbach; Stephen Roller; Andrew Rosenberg; Mihai Rotaru; Michael Roth; Johann Roturier; Salim Roukos; Mickael Rouvier; Alla Rozovskaya; Frank Rudzicz; Alexander M. Rush;

Markus Saers; Horacio Saggion; Patrick Saint-dizier; Hassan Sajjad; Keisuke Sakaguchi; Mohammad Salameh; Rajhans Samdani; Mark Sammons; Felipe Sánchez-Martínez; Germán Sanchis-Trilles; Murat Saraclar; Ruhi Sarikaya; Manabu Sassano; Asad Sayeed; Carolina Scarton; David Schlangen; Jonathan Schler; Natalie Schluter; Helmut Schmid; William Schuler; Lane Schwartz; Hansen Andrew Schwartz; Roy Schwartz; Holger Schwenk; Djamé Seddah; Satoshi Sekine; Jean Senellart; Rico Sennrich; Burr Settles; Aliaksei Severyn; Kashif Shah; Serge Sharoff; Shuming Shi; Xiaodong Shi; Chaitanya Shivade; Avirup Sil; Fabrizio Silvestri; Yanchuan Sim; Khalil Sima'an; Michel Simard; Patrick Simianer; Kiril Simov; Sameer Singh; Gabriel Skantze; Steve Skiena; Noam Slonim; Kevin Small; Jan Šnajder; Richard Socher; Anders Søgaard; Thamar Solorio; Swapna Somasundaran; Linfeng Song; Akshay Soni; Alessandro Sordoni; Radu Soricut; Caroline Sporleder; Rohini Srihari; Vivek Srikumar; Christian Stab; Edward Stabler; Sanja Štajner; Miloš Stanojević; Mark Steedman; Benno Stein; Josef Steinberger; Pontus Stenetorp; Amanda Stent; Evgeny Stepanov; Brandon Stewart; Veselin Stoyanov; Karl Stratos; Jannik Strötgen; Jinsong Su; Qi Su; Fabian Suchanek; Kazunari Sugiyama; Aixin Sun; Huan Sun; Le Sun; Fei Sun; Mihai Surdeanu; Swabha Swayamdipta; Gabriel Synnaeve;

Oscar Täckström; Hiroya Takamura; David Talbot; Partha Talukdar; Kumiko Tanaka-Ishii; Hristo Tanev; Duyu Tang; Jiliang Tang; Jian Tang; Xavier Tannier; Makarand Tapaswi; Kapil Thadani; Jörg Tiedemann; Christoph Tillmann; Ivan Titov; Takenobu Tokunaga; Nadi Tomeh; Sara Tonelli;

Kentaro Torisawa; Isabel Trancoso; Oren Tsur; Yoshimasa Tsuruoka; Yulia Tsvetkov; Gokhan Tur;

Raghavendra Udupa; Lyle Ungar; L. Alfonso Urena Lopez; Raquel Urtasun; Nicolas Usunier; Jakob Uszkoreit; Naushad UzZaman;

Sowmya Vajjala; Marten van Schijndel; Vasudeva Varma; Ashish Vaswani; Paola Velardi; Sriram Venkatapathy; Giulia Venturi; Ashish Venugopal; Marc Verhagen; Yannick Versley; David Vilar; Aline Villavicencio; Andreas Vlachos; Rob Voigt; Svitlana Volkova;

Marilyn Walker; Matthew Walter; Stephen Wan; Chuan Wang; Josiah Wang; Lu Wang; Sida I. Wang; Houfeng Wang; William Yang Wang; Zhongqing Wang; Zhiguo Wang; Zeerak Waseem; Taro Watanabe; Aleksander Wawer; Bonnie Webber; Julie Weeds; Zhongyu Wei; Gerhard Weikum; Ralph Weischedel; Michael White; Michael Wiegand; John Wieting; Jason D. Williams; Colin Wilson; Shuly Wintner; Sam Wiseman; Silke Witt-Ehsani; Kam-Fai Wong; Stephen Wu; Hua Wu; Yuanbin Wu; Joern Wuebker;

Rui Xia; Yunqing Xia; Chunyang Xiao; Boyi Xie; Deyi Xiong; Liheng Xu; Peng Xu; Ruifeng Xu; Nianwen Xue;

Bishan Yang; Diyi Yang; Roman Yangarber; Helen Yannakoudakis; Mark Yatskar; Wenpeng Yin; Dani Yogatama; Kai Yu; Liang-Chih Yu; Lei Yu; Zhou Yu; François Yvon;

Marcos Zampieri; Fabio Massimo Zanzotto; Alessandra Zarcone; Amir Zeldes; Richard Zens; Torsten Zesch; Luke Zettlemoyer; Congle Zhang; Yue Zhang; Hao Zhang; Hui Zhang; Jiajun Zhang; Qi Zhang; Lei Zhang; Xingxing Zhang; Yuan Zhang; Min Zhang; Min Zhang; Wei Zhang; Hai Zhao; Wayne Xin Zhao; Jun Zhao; Bowen Zhou; Guodong Zhou; Yu Zhou; Xinjie Zhou; Xiaodan Zhu; Jingbo Zhu; Muhua Zhu; Larry Zitnick; Chengqing Zong; Pierre Zweigenbaum;

Secondary Reviewers

Nitish Aggarwal; Khalid Al-Khatib; Mihael Arcan; Aitziber Atutxa; Wilker Aziz;

Vit Baisa; JinYeong Bak; Jeremy Barnes;

Hongshen Chen; Wei-Te Chen; Nicolas Collignon;

Thomas Demeester; Shichao Dong;

Liat Ein-Dor; Akiko Eriguchi;

Federico Fancellu; Wes Feely; Lorenzo Ferrone; Marjorie Freedman;

Jinghan Gu; James Gung;

Jialong Han; Bradley Hauer; Gerold Hintz;

Mengxiao Jiang; Salud María Jiménez-Zafra; Melvin Johnson;

Johannes Kiesel; Nikita Kitaev; Hayato Kobayashi; Vojtech Kovar; Mikhail Kozhevnikov; Ákos Kádár;

Ran Levy; Bo Li; Chen Li; Yitong Li; Lizi Liao; Ming Liao; Jiangming Liu; Sijia Liu; Jackie Chi-kiu Lo; Nikhil Londhe; Adrian Pastor López-Monroy;

Zongyang Ma; Suraj Maharjan; Hector Martínez Alonso; Eugenio Martínez Cámara; Lu Meng; Todor Mihaylov; Benjamin Milde; Sadegh Mirshekarian; Arindam Mitra; Tomoya Mizumoto;

Garrett Nicolai; Azadeh Nikfarjam; Scott Nowson;

Tim O'Gorman; Lydia Odilinye; Jong-Hoon Oh; Takeshi Onishi;

Grabrela Ramírez-de-la-Rosa; Steffen Remus; Claude Roux;

Abeed Sarker; Andrew Schneider; Minjoon Seo; Hui Shen; Prasha Shrestha; Vered Shwartz; Suzanna Sia; Edwin Simpson; Gaurav Singh; Edmundo Pavel Soriano Morales; P.K. Srijith; Gabriel Stanovsky;

Tasnia Tahsin; Ke Tao; Milan Tofiloski; Khoa Tran; Kateryna Tymoshenko;

Jason Utt;

Esaú Villatoro-Tello; Yogarshi Vyas;

Henning Wachsmuth; Boli Wang; Feixiang Wang; Huimin Wang; Shuai Wang; Wenya Wang; Yanshan Wang; Shawn Tsung-Hsien Wen; Zhongyu Wei; Andy Wetta; Guillaume Wisniewski; Shumin Wu;

Haitong Yang; Seid Muhie Yimam;

Yunxiao Zhou;

Invited Speaker: Christopher Potts

Learning in Extended and Approximate Rational Speech Acts Models

Abstract: The Rational Speech Acts (RSA) model treats language use as a recursive process in which probabilistic speaker and listener agents reason about each other's intentions to enrich, and negotiate, the semantics of their language along broadly Gricean lines. RSA builds on early work by the philosopher David Lewis and others on signaling systems as well as more recent developments in Bayesian cognitive modeling. Over the last five years, RSA has been shown to provide a unified account of numerous core phenomena in pragmatics, including metaphor, hyperbole, sarcasm, politeness, and a wide range of conversational implicatures. Its precise, quantitative nature has also facilitated an outpouring of new experimental work on these phenomena. However, applications of RSA to large-scale problems in NLP and AI have so far been limited, because the exact version of the model is intractable along several dimensions. In this talk, I'll report on recent progress in approximating RSA in ways that retains its core properties while enabling application to large datasets and complex environments in which language and action are brought together.

Bio: Christopher Potts is Professor of Linguistics and, by courtesy, of Computer Science, at Stanford, and Director of the Center for the Study of Language and Information (CSLI) at Stanford. He earned his BA in Linguistics from NYU in 1999 and his PhD from UC Santa Cruz in 2003. He was on the faculty in Linguistics at UMass Amherst from 2003 until 2009, when he headed west once again, to join Stanford Linguistics. He was a co-editor at Linguistic Inquiry 2004–2006, an associate editor at Linguistics and Philosophy 2009–2012, and has been an Action Editor at TACL since 2014. In his research, he uses computational methods to explore how emotion is expressed in language and how linguistic production and interpretation are influenced by the context of utterance. He is the author of the 2005 book The Logic of Conventional Implicatures as well as numerous scholarly papers in computational and theoretical linguistics.

Invited Speaker: Andreas Stolcke

You Talking to Me? Speech-based and Multimodal Approaches for Human versus Computer Addressee Detection

Abstract: As dialog systems become ubiquitous, we must learn how to detect when a system is spoken to, and avoid mistaking human-human speech as computer-directed input. In this talk I will discuss approaches to addressee detection in this human-human-machine dialog scenario, based on what is being said (lexical information), how it is being said (acoustic-prosodic properties), and non-speech multimodal and contextual information. I will present experimental results showing that a combination of these cues can be used effectively for human/computer address classification in several dialog scenarios.

Bio: Andreas Stolcke received a Ph.D. in computer science from the University of California at Berkeley. He was subsequently a Senior Research Engineer with the Speech Technology and Research Laboratory at SRI International, Menlo Park, CA, and is currently a Principal Researcher with the Speech and Dialog Research Group in the Microsoft Advanced Technology-Information Services group, working out of Mountain View, CA. His research interests include language modeling, speech recognition, speaker recognition, and speech understanding. He has published over 200 papers in these areas, as well as SRILM, a widely used open-source toolkit for statistical language modeling. He is a Fellow of the IEEE and of ISCA, the International Speech Communications Association.

Invited Speaker: Stefanie Tellex

Learning Models of Language, Action and Perception for Human-Robot Collaboration

Abstract: Robots can act as a force multiplier for people, whether a robot assisting an astronaut with a repair on the International Space station, a UAV taking flight over our cities, or an autonomous vehicle driving through our streets. To achieve complex tasks, it is essential for robots to move beyond merely interacting with people and toward collaboration, so that one person can easily and flexibly work with many autonomous robots. The aim of my research program is to create autonomous robots that collaborate with people to meet their needs by learning decision-theoretic models for communication, action, and perception. Communication for collaboration requires models of language that map between sentences and aspects of the external world. My work enables a robot to learn compositional models for word meanings that allow a robot to explicitly reason and communicate about its own uncertainty, increasing the speed and accuracy of human-robot communication. Action for collaboration requires models that match how people think and talk, because people communicate about all aspects of a robot's behavior, from low-level motion preferences (e.g., "Please fly up a few feet") to high-level requests (e.g., "Please inspect the building"). I am creating new methods for learning how to plan in very large, uncertain state-action spaces by using hierarchical abstraction. Perception for collaboration requires the robot to detect, localize, and manipulate the objects in its environment that are most important to its human collaborator. I am creating new methods for autonomously acquiring perceptual models in situ so the robot can perceive the objects most relevant to the human's goals. My unified decision-theoretic framework supports data-driven training and robust, feedback-driven human-robot collaboration.

Bio: Stefanie Tellex is an Assistant Professor of Computer Science and Assistant Professor of Engineering at Brown University. Her group, the Humans To Robots Lab, creates robots that seamlessly collaborate with people to meet their needs using language, gesture, and probabilistic inference, aiming to empower every person with a collaborative robot. She completed her Ph.D. at the MIT Media Lab in 2010, where she developed models for the meanings of spatial prepositions and motion verbs. Her postdoctoral work at MIT CSAIL focused on creating robots that understand natural language. She has published at SIGIR, HRI, RSS, AAAI, IROS, ICAPs and ICMI, winning Best Student Paper at SIGIR and ICMI, Best Paper at RSS, and an award from the CCC Blue Sky Ideas Initiative. Her awards include being named one of IEEE Spectrum's AI's 10 to Watch in 2013, the Richard B. Salomon Faculty Research Award at Brown University, a DARPA Young Faculty Award in 2015, and a 2016 Sloan Research Fellowship. Her work has been featured in the press on National Public Radio, MIT Technology Review, Wired UK and the Smithsonian. She was named one of Wired UK's Women Who Changed Science In 2015 and listed as one of MIT Technology Review's Ten Breakthrough Technologies in 2016.

Conference Program

Tuesday, November 1, 2016

18:30–20:00 Welcome Reception

Wednesday, November 2, 2016

07:30–17:30 Registration Day 1

08:00–08:40 ***Morning Coffee***

08:40–09:00 Session P1: Plenary Session: Opening Remarks

08:40–09:00 *Opening Remarks*
General Chair, PC Co-Chairs

09:00–10:00 Session P2: Plenary Session: Invited Talk by Christopher Potts

09:00–10:00 *Learning in Extended and Approximate Rational Speech Acts Models*
Christopher Potts

10:00–10:30 ***Coffee Break***

10:30–12:10 Session 1A: Parsing and Syntax (Long Papers)

10:30–10:55 *Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles*
James Cross and Liang Huang

10:55–11:20 *Rule Extraction for Tree-to-Tree Transducers by Cost Minimization*
Pascual Martínez-Gómez and Yusuke Miyao

11:20–11:45 *A Neural Network for Coordination Boundary Prediction*
Jessica Ficler and Yoav Goldberg

11:45–12:10 *Using Left-corner Parsing to Encode Universal Structural Constraints in Grammar Induction*
Hiroshi Noji, Yusuke Miyao and Mark Johnson

10:30–12:10 Session 1B: Information Extraction (Long Papers)

10:30–10:55 *Distinguishing Past, On-going, and Future Events: The EventStatus Corpus*
Ruihong Huang, Ignacio Cases, Dan Jurafsky, Cleo Condoravdi and Ellen Riloff

10:55–11:20 *Nested Propositions in Open Information Extraction*
Nikita Bhutani, H V Jagadish and Dragomir Radev

11:20–11:45 *A Position Encoding Convolutional Neural Network Based on Dependency Tree for Relation Classification*
Yunlun Yang, Yunhai Tong, Shulei Ma and Zhi-Hong Deng

11:45–12:10 *Learning to Recognize Discontiguous Entities*
Aldrian Obaja Muis and Wei Lu

10:30–12:10 Session 1C: Psycholinguistics / Machine Learning (Long Papers)

10:30–10:55 *Modeling Human Reading with Neural Attention*
Michael Hahn and Frank Keller

10:55–11:20 *Comparing Computational Cognitive Models of Generalization in a Language Acquisition Task*
Libby Barak, Adele E. Goldberg and Suzanne Stevenson

11:20–11:45 *Rationalizing Neural Predictions*
Tao Lei, Regina Barzilay and Tommi Jaakkola

11:45–12:10 *Deep Multi-Task Learning with Shared Memory for Text Classification*
Pengfei Liu, Xipeng Qiu and Xuanjing Huang

12:10–13:40 ***Lunch***

13:40–15:20 Session 2A: Reading Comprehension and Question Answering (Long Papers)

13:40–14:05 *Natural Language Comprehension with the EpiReader*
Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni and Kaheer Suleman

14:05–14:30 *Creating Causal Embeddings for Question Answering with Minimal Supervision*
Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark and Michael Hammond

14:30–14:55 *Improving Semantic Parsing via Answer Type Inference*
Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa and Xifeng Yan

14:55–15:20 *Semantic Parsing to Probabilistic Programs for Situated Question Answering*
Jayant Krishnamurthy, Oyvind Tafjord and Aniruddha Kembhavi

13:40–15:20 Session 2B: Embeddings of Linguistic Structure (Long Papers)

13:40–14:05 *Event participant modelling with neural networks*
Ottokar Tilk, Vera Demberg, Asad Sayeed, Dietrich Klakow and Stefan Thater

14:05–14:30 *Context-Dependent Sense Embedding*
Lin Qiu, Kewei Tu and Yong Yu

14:30–14:55 *Jointly Embedding Knowledge Graphs and Logical Rules*
Shu Guo, Quan Wang, Lihong Wang, Bin Wang and Li Guo

14:55–15:20 *Learning Connective-based Word Representations for Implicit Discourse Relation Identification*
Chloé Braud and Pascal Denis

13:40–15:20 Session 2C: Sentiment and Opinion Analysis (Long Papers)

13:40–14:05 *Aspect Level Sentiment Classification with Deep Memory Network*
Duyu Tang, Bing Qin and Ting Liu

14:05–14:30 *Lifelong-RL: Lifelong Relaxation Labeling for Separating Entities and Aspects in Opinion Targets*
Lei Shu, Bing Liu, Hu Xu and Annice Kim

14:30–14:55 *Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification*
Jianfei Yu and Jing Jiang

14:55–15:20 *Attention-based LSTM Network for Cross-Lingual Sentiment Classification*
Xinjie Zhou, Xiaojun Wan and Jianguo Xiao

15:20–15:50 ***Coffee Break***

Wednesday, November 2, 2016 (continued)

15:50–17:30 Session 3A: Neural Machine Translation (Long + TACL Papers)

15:50–16:15 *[TACL] Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation*
Jie Zhou, Ying Cao, Xuguang Wang, Peng Li and Wei Xu

16:15–16:40 *Neural versus Phrase-Based Machine Translation Quality: a Case Study*
Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo and Marcello Federico

16:40–17:05 *Zero-Resource Translation with Multi-Lingual Neural Machine Translation*
Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural and Kyunghyun Cho

17:05–17:30 *Memory-enhanced Decoder for Neural Machine Translation*
Mingxuan Wang, Zhengdong Lu, Hang Li and Qun Liu

15:50–17:30 Session 3B: Semi-supervised and Minimally Supervised Learning (Long + TACL Papers)

15:50–16:15 *Semi-Supervised Learning of Sequence Models with Method of Moments*
Zita Marinho, André F. T. Martins, Shay B. Cohen and Noah A. Smith

16:15–16:40 *[TACL] Minimally supervised models for number normalization*
Kyle Gorman and Richard Sproat

16:40–17:05 *Learning from Explicit and Implicit Supervision Jointly For Algebra Word Problems*
Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang and Wen-tau Yih

17:05–17:30 *TweeTime : A Minimally Supervised Method for Recognizing and Normalizing Time Expressions in Twitter*
Jeniya Tabassum, Alan Ritter and Wei Xu

15:50–17:30 Session 3C: Summarization and Generation (Long Papers)

15:50–16:15 *Language as a Latent Variable: Discrete Generative Models for Sentence Compression*
Yishu Miao and Phil Blunsom

16:15–16:40 *Globally Coherent Text Generation with Neural Checklist Models*
Chloé Kiddon, Luke Zettlemoyer and Yejin Choi

16:40–17:05 *A Dataset and Evaluation Metrics for Abstractive Compression of Sentences and Short Paragraphs*
Kristina Toutanova, Chris Brockett, Ke M. Tran and Saleema Amershi

17:05–17:30 *PaCCSS-IT: A Parallel Corpus of Complex-Simple Sentences for Automatic Text Simplification*
Dominique Brunato, Andrea Cimino, Felice Dell'Orletta and Giulia Venturi

17:30–17:45 *Break*

17:45–18:15 Session P3: Plenary Session: Half Minute Madness A

18:15–20:15 Session P4: Poster Session A

[L01][DISCOURSE & DIALOGUE] *Discourse Parsing with Attention-based Hierarchical Neural Networks*
Qi Li, Tianshi Li and Baobao Chang

[L02][DISCOURSE & DIALOGUE] *Multi-view Response Selection for Human-Computer Conversation*
Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu and Rui Yan

[L03][DISCOURSE & DIALOGUE] *Variational Neural Discourse Relation Recognizer*
Biao Zhang, Deyi Xiong, jinsong su, Qun Liu, Rongrong Ji, Hong Duan and Min Zhang

[L04][DISCOURSE & DIALOGUE] *Event Detection and Co-reference with Minimal Supervision*
Haoruo Peng, Yangqiu Song and Dan Roth

[L05][INFORMATION EXTRACTION] *Learning Term Embeddings for Taxonomic Relation Identification Using Dynamic Weighting Neural Network*
Tuan Luu Anh, Yi Tay, Siu Cheung Hui and See Kiong Ng

[L06][INFORMATION EXTRACTION] *Relation Schema Induction using Tensor Factorization with Side Information*
Madhav Nimishakavi, Uday Singh Saini and Partha Talukdar

[L07][INFORMATION EXTRACTION] *Supervised Distributional Hypernym Discovery via Domain Adaptation*
Luis Espinosa Anke, Jose Camacho-Collados, Claudio Delli Bovi and Horacio Saggion

[L08][LANGUAGE MODELING] *Latent Tree Language Model*
Tomáš Brychcín

[L09][LANGUAGE & VISION] *Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics*
Douwe Kiela, Anita Lilla Verő and Stephen Clark

[L10][LANGUAGE & VISION] *Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding*
Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell and Marcus Rohrbach

[L11][MACHINE LEARNING] *The Structured Weighted Violations Perceptron Algorithm*
Rotem Dror and Roi Reichart

[L12][MACHINE LEARNING] *How Transferable are Neural Networks in NLP Applications?*
Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang and Zhi Jin

[L13][MACHINE LEARNING] *Morphological Priors for Probabilistic Neural Word Embeddings*
Parminder Bhatia, Robert Guthrie and Jacob Eisenstein

[L14][MACHINE TRANSLATION] *Automatic Cross-Lingual Similarization of Dependency Grammars for Tree-based Machine Translation*
Wenbin Jiang, Wen Zhang, Jinan Xu and Rangjia Cai

[L15][MACHINE TRANSLATION] *IRT-based Aggregation Model of Crowdsourced Pairwise Comparison for Evaluating Machine Translations*
Naoki Otani, Toshiaki Nakazawa, Daisuke Kawahara and Sadao Kurohashi

[L16][MACHINE TRANSLATION] *Variational Neural Machine Translation*
Biao Zhang, Deyi Xiong, jinsong su, Hong Duan and Min Zhang

[L17][MACHINE TRANSLATION] *Towards a Convex HMM Surrogate for Word Alignment*
Andrei Simion, Michael Collins and Cliff Stein

[L18][QUESTION ANSWERING] *Solving Verbal Questions in IQ Test by Knowledge-Powered Word Embedding*
Huazheng Wang, Fei Tian, Bin Gao, Chengjieren Zhu, Jiang Bian and Tie-Yan Liu

[L19][QUESTION ANSWERING] *Long Short-Term Memory-Networks for Machine Reading*
Jianpeng Cheng, Li Dong and Mirella Lapata

[L20][QUESTION ANSWERING] *On Generating Characteristic-rich Question Sets for QA Evaluation*
Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan and Xifeng Yan

[L21][QUESTION ANSWERING] *Learning to Translate for Multilingual Question Answering*
Ferhan Ture and Elizabeth Boschee

[L22][QUESTION ANSWERING] *A Semiparametric Model for Bayesian Reader Identification*
Ahmed Abdelwahab, Reinhold Kliegl and Niels Landwehr

[L23][SENTIMENT ANALYSIS] *Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora*
William L. Hamilton, Kevin Clark, Jure Leskovec and Dan Jurafsky

[L24][SENTIMENT ANALYSIS] *Attention-based LSTM for Aspect-level Sentiment Classification*
Yequan Wang, Minlie Huang, xiaoyan zhu and Li Zhao

[L25][SENTIMENT ANALYSIS] *Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis*
Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier and Xiaokui Xiao

[L26][SENTIMENT ANALYSIS] *Extracting Aspect Specific Opinion Expressions*
Abhishek Laddha and Arjun Mukherjee

[L27][SENTIMENT ANALYSIS] *Emotion Distribution Learning from Texts*
Deyu ZHOU, Xuan Zhang, Yin Zhou, Quan Zhao and Xin Geng

[L28][SEMANTICS] *Building an Evaluation Scale using Item Response Theory*
John Lalor, Hao Wu and hong yu

[L29][SEMANTICS] *WordRank: Learning Word Embeddings via Robust Ranking*
Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima and S. V. N. Vishwanathan

[L30][SEMANTICS] *Exploring Semantic Representation in Brain Activity Using Word Embeddings*
Yu-Ping Ruan, Zhen-Hua Ling and Yu Hu

[L31][SEMANTICS] *AMR Parsing with an Incremental Joint Model*
Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li and Yanhui Gu

[L32][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Identifying Dogmatism in Social Media: Signals and Models*
Ethan Fast and Eric Horvitz

[L33][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Enhanced Personalized Search using Social Data*
Dong Zhou, Séamus Lawless, Xuan Wu, Wenyu Zhao and Jianxun Liu

[L34][SYNTAX & MORPHOLOGY] *Effective Greedy Inference for Graph-based Non-Projective Dependency Parsing*
Ilan Tchernowitz, Liron Yedidsion and Roi Reichart

[L35][SYNTAX & MORPHOLOGY] *Generating Abbreviations for Chinese Named Entities Using Recurrent Neural Network with Dynamic Dictionary*
Qi Zhang, Jin Qian, Ya Guo, Yaqian Zhou and Xuanjing Huang

[L36][SYNTAX & MORPHOLOGY] *Neural Network for Heterogeneous Annotations*
Hongshen Chen, Yue Zhang and Qun Liu

[L37][SYNTAX & MORPHOLOGY] *LAMB: A Good Shepherd of Morphologically Rich Languages*
Sebastian Ebert, Thomas Müller and Hinrich Schütze

[L38][SYNTAX & MORPHOLOGY] *Fast Coupled Sequence Labeling on Heterogeneous Annotations via Context-aware Pruning*
Zhenghua Li, Jiayuan Chao, Min Zhang and Jiwen Yang

[L39][SYNTAX & MORPHOLOGY] *Unsupervised Neural Dependency Parsing*
Yong Jiang, Wenjuan Han and Kewei Tu

[L40][SUMMARIZATION] *Generating Coherent Summaries of Scientific Articles Using Coherence Patterns*
Daraksha Parveen, Mohsen Mesgar and Michael Strube

[L41][SUMMARIZATION] *News Stream Summarization using Burst Information Networks*
Tao Ge, Lei Cui, Baobao Chang, Sujian Li, Ming Zhou and Zhifang Sui

[L42][TEXT MINING & APPLICATIONS] *Rationale-Augmented Convolutional Neural Networks for Text Classification*
Ye Zhang, Iain Marshall and Byron C. Wallace

[L43][TEXT MINING & APPLICATIONS] *Transferring User Interests Across Websites with Unstructured Text for Cold-Start Recommendation*
Yu-Yang Huang and Shou-De Lin

[L44][TEXT MINING & APPLICATIONS] *Speculation and Negation Scope Detection via Convolutional Neural Networks*
Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo and Wei Luo

[L45][TEXT MINING & APPLICATIONS] *Analyzing Linguistic Knowledge in Sequential Model of Sentence*
Peng Qian, Xipeng Qiu and Xuanjing Huang

[L46][TEXT MINING & APPLICATIONS] *Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter*
Qi Zhang, Yang Wang, Yeyun Gong and Xuanjing Huang

[L47][TEXT MINING & APPLICATIONS] *Solving and Generating Chinese Character Riddles*
Chuanqi Tan, Furu Wei, Li Dong, Weifeng Lv and Ming Zhou

[L48][TEXT MINING & APPLICATIONS] *Structured prediction models for RNN based sequence labeling in clinical text*
Abhyuday Jagannatha and hong yu

[L49][TEXT MINING & APPLICATIONS] *Learning to Represent Review with Tensor Decomposition for Spam Detection*
Xuepeng Wang, Kang Liu, Shizhu He and Jun Zhao

[L50][TEXT MINING & APPLICATIONS] *Stance Detection with Bidirectional Conditional Encoding*
Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos and Kalina Bontcheva

[S01][INFORMATION EXTRACTION] *Modeling Skip-Grams for Event Detection with Convolutional Neural Networks*
Thien Huu Nguyen and Ralph Grishman

[S02][INFORMATION EXTRACTION] *Porting an Open Information Extraction System from English to German*
Tobias Falke, Gabriel Stanovsky, Iryna Gurevych and Ido Dagan

[S03][INFORMATION EXTRACTION] *Named Entity Recognition for Novel Types by Transfer Learning*
Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou and Timothy Baldwin

[S04][INFORMATION EXTRACTION] *Extracting Subevents via an Effective Two-phase Approach*
Allison Badgett and Ruihong Huang

[S05][LANGUAGE & VISION] *Gaussian Visual-Linguistic Embedding for Zero-Shot Recognition*
Tanmoy Mukherjee and Timothy Hospedales

[S06][LANGUAGE & VISION] *Question Relevance in VQA: Identifying Non-Visual And False-Premise Questions*
Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra and Devi Parikh

[S07][LANGUAGE & VISION] *Sort Story: Sorting Jumbled Images and Captions into Stories*
Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh and Mohit Bansal

[S08][LANGUAGE & VISION] *Human Attention in Visual Question Answering: Do Humans and Deep Networks look at the same regions?*
Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh and Dhruv Batra

[S09][MACHINE LEARNING] *Recurrent Residual Learning for Sequence Classification*
Yiren Wang and Fei Tian

[S10][MACHINE LEARNING] *Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling*
Ehsan Shareghi, Trevor Cohn and Gholamreza Haffari

[S11][MACHINE LEARNING] *A General Regularization Framework for Domain Adaptation*
Wei Lu, Hai Leong Chieu and Jonathan Löfgren

[S12][MACHINE TRANSLATION] *Coverage Embedding Models for Neural Machine Translation*
Haitao Mi, Baskaran Sankaran, Zhiguo Wang and Abe Ittycheriah

[S13][SYNTAX & MORPHOLOGY] *Neural Morphological Analysis: Encoding-Decoding Canonical Segments*
Katharina Kann, Ryan Cotterell and Hinrich Schütze

[S14][SYNTAX & MORPHOLOGY] *Exploiting Mutual Benefits between Syntax and Semantic Roles using Neural Network*
Peng Shi, Zhiyang Teng and Yue Zhang

[S15][SEMANTICS] *The Effects of Data Size and Frequency Range on Distributional Semantic Models*
Magnus Sahlgren and Alessandro Lenci

[S16][SEMANTICS] *Multi-Granularity Chinese Word Embedding*
Rongchao Yin, Quan Wang, Peng Li, Rui Li and Bin Wang

[S17][SEMANTICS] *Numerically Grounded Language Models for Semantic Error Correction*
Georgios Spithourakis, Isabelle Augenstein and Sebastian Riedel

[S18][SEMANTICS] *Towards Semi-Automatic Generation of Proposition Banks for Low-Resource Languages*
Alan Akbik, vishwajeet kumar and Yunyao Li

[S19][SENTIMENT ANALYSIS] *A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis*
Sebastian Ruder, Parsa Ghaffari and John G. Breslin

[S20][SENTIMENT ANALYSIS] *Are Word Embedding-based Features Useful for Sarcasm Detection?*
Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya and Mark Carman

[S21][SENTIMENT ANALYSIS] *Weakly Supervised Tweet Stance Classification by Relational Bootstrapping*
Javid Ebrahimi, Dejing Dou and Daniel Lowd

[S22][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *The Gun Violence Database: A new task and data set for NLP*
Ellie Pavlick, Heng Ji, Xiaoman Pan and Chris Callison-Burch

[S23][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Fluency detection on communication networks*
Tom Lippincott and Benjamin Van Durme

[S25][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Characterizing the Language of Online Communities and its Relation to Community Reception*
Trang Tran and Mari Ostendorf

[S26][SPOKEN LANGUAGE PROCESSING] *Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts*
Masashi Yoshikawa, Hiroyuki Shindo and Yuji Matsumoto

[S27][SPOKEN LANGUAGE PROCESSING] *Real-Time Speech Emotion and Sentiment Recognition for Interactive Dialogue Systems*
Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Ho Yin Chan and Pascale Fung

[S28][SUMMARIZATION] *A Neural Network Architecture for Multilingual Punctuation Generation*
Miguel Ballesteros and Leo Wanner

[S29][SUMMARIZATION] *Neural Headline Generation on Abstract Meaning Representation*
Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao and Masaaki Nagata

[S30][TEXT MINING & APPLICATIONS] *Robust Gram Embeddings*
Taygun Kekec and David M. J. Tax

[S31][TEXT MINING & APPLICATIONS] *SimpleScience: Lexical Simplification of Scientific Terminology*
Yea Seul Kim, Jessica Hullman, Matthew Burgess and Eytan Adar

[S32][TEXT MINING & APPLICATIONS] *Automatic Features for Essay Scoring – An Empirical Study*
Fei Dong and Yue Zhang

Thursday, November 3, 2016

07:30–17:30 Registration Day 2

08:00–09:00 ***Morning Coffee***

09:00–10:00 Session P5: Plenary Session: Invited Talk by Stefanie Tellex

09:00–10:00 *Learning Models of Language, Action and Perception for Human-Robot Collaboration*
Stefanie Tellex

10:00–10:30 ***Coffee Break***

10:30–12:10 Session 4A: Semantics and Semantic Parsing (Long Papers)

10:30–10:55 *Semantic Parsing with Semi-Supervised Sequential Autoencoders*
Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom and Karl Moritz Hermann

10:55–11:20 *Equation Parsing : Mapping Sentences to Grounded Equations*
Subhro Roy, Shyam Upadhyay and Dan Roth

11:20–11:45 *Automatic Extraction of Implicit Interpretations from Modal Constructions*
Jordan Sanders and Eduardo Blanco

11:45–12:10 *Understanding Negation in Positive Terms Using Syntactic Dependencies*
Zahra Sarabi and Eduardo Blanco

10:30–12:10 Session 4B: NLP for Social Science and Health (Long + TACL Papers)

10:30–10:55 *Demographic Dialectal Variation in Social Media: A Case Study of African-American English*
Su Lin Blodgett, Lisa Green and Brendan O'Connor

10:55–11:20 *Understanding Language Preference for Expression of Opinion and Sentiment: What do Hindi-English Speakers do on Twitter?*
Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury and Niloy Ganguly

11:20–11:45 *Detecting and Characterizing Events*
Allison Chaney, Hanna Wallach, Matthew Connelly and David Blei

11:45–12:10 *[TACL] Large-scale Analysis of Counseling Conversations: An Application of Natural Language Processing to Mental Health*
Tim Althoff, Kevin Clark and Jure Leskovec

10:30–12:10 Session 4C: Language Models (Long + TACL Papers)

10:30–10:55 *[TACL] Fast, Small and Exact: Infinite-order Language Modelling with Compressed Suffix Trees*
Ehsan Shareghi, Matthias Petri, Gholamreza Haffari and Trevor Cohn

10:55–11:20 *Convolutional Neural Network Language Models*
Ngoc-Quan Pham, Germán Kruszewski and Gemma Boleda

11:20–11:45 *[TACL] Sparse Non-negative Matrix Language Modeling*
Joris Pelemans, Noam Shazeer and Ciprian Chelba

11:45–12:10 *Generalizing and Hybridizing Count-based and Neural Language Models*
Graham Neubig and Chris Dyer

12:10–13:40 *Lunch*

Thursday, November 3, 2016 (continued)

13:00–13:40 Session P6: SIGDAT Business Meeting

13:40–15:20 Session 5A: Text Generation (Long Papers)

13:40–14:05 *Reasoning about Pragmatics with Neural Listeners and Speakers*
Jacob Andreas and Dan Klein

14:05–14:30 *Generating Topical Poetry*
Marjan Ghazvininejad, Xing Shi, Yejin Choi and Kevin Knight

14:30–14:55 *Deep Reinforcement Learning for Dialogue Generation*
Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley and Jianfeng Gao

14:55–15:20 *Neural Text Generation from Structured Data with Application to the Biography Domain*
Rémi Lebret, David Grangier and Michael Auli

13:40–15:20 Session 5B: Discourse and Document Structure (Long Papers)

13:40–14:05 *What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in Web argumentation*
Ivan Habernal and Iryna Gurevych

14:05–14:30 *Recognizing Implicit Discourse Relations via Repeated Reading: Neural Networks with Multi-Level Attention*
Yang Liu and Sujian Li

14:30–14:55 *Antecedent Selection for Sluicing: Structure and Content*
Pranav Anand and Daniel Hardt

14:55–15:20 *Intra-Sentential Subject Zero Anaphora Resolution using Multi-Column Convolutional Neural Network*
Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai and Julien Kloetzer

Thursday, November 3, 2016 (continued)

13:40–15:20 Session 5C: Machine Translation and Multilingual Applications (Long Papers)

13:40–14:05 *An Unsupervised Probability Model for Speech-to-Translation Alignment of Low-Resource Languages*
Antonios Anastasopoulos, David Chiang and Long Duong

14:05–14:30 *HUME: Human UCCA-Based Evaluation of Machine Translation*
Alexandra Birch, Omri Abend, Ondřej Bojar and Barry Haddow

14:30–14:55 *Improving Multilingual Named Entity Recognition with Wikipedia Entity Type Mapping*
Jian Ni and Radu Florian

14:55–15:20 *Learning Crosslingual Word Embeddings without Bilingual Corpora*
Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird and Trevor Cohn

15:20–15:50 *Coffee Break*

15:50–17:30 Session 6A: Neural Sequence-to-Sequence Models (Long Papers)

15:50–16:15 *Sequence-to-Sequence Learning as Beam-Search Optimization*
Sam Wiseman and Alexander M. Rush

16:15–16:40 *Online Segment to Segment Neural Transduction*
Lei Yu, Jan Buys and Phil Blunsom

16:40–17:05 *Sequence-Level Knowledge Distillation*
Yoon Kim and Alexander M. Rush

17:05–17:30 *Controlling Output Length in Neural Encoder-Decoders*
Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura and Manabu Okumura

15:50–17:30 Session 6B: Text Mining and NLP Applications (Long + TACL Papers)

15:50–16:15 *Poet Admits // Mute Cypher: Beam Search to find Mutually Enciphering Poetic Texts*
Cole Peterson and Alona Fyshe

16:15–16:40 *All Fingers are not Equal: Intensity of References in Scientific Articles*
Tanmoy Chakraborty and Ramasuri Narayanam

16:40–17:05 *Improving Users' Demographic Prediction via the Videos They Talk about*
Yuan Wang, Yang Xiao, Chao Ma and Zhen Xiao

17:05–17:30 *[TACL] Understanding Satirical Articles Using Common-Sense*
Dan Goldwasser and Xiao Zhang

15:50–17:30 Session 6C: Knowledge Base and Inference (Long Papers)

15:50–16:15 *AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding*
Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji and Jiawei Han

16:15–16:40 *Mining Inference Formulas by Goal-Directed Random Walks*
Zhuoyu Wei, Jun Zhao and Kang Liu

16:40–17:05 *Lifted Rule Injection for Relation Embeddings*
Thomas Demeester, Tim Rocktäschel and Sebastian Riedel

17:05–17:30 *Key-Value Memory Networks for Directly Reading Documents*
Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes and Jason Weston

17:30–17:45 *Break*

Thursday, November 3, 2016 (continued)

17:45–18:15 Session P7: Plenary Session: Half-minute Madness B

18:15–20:15 Session P8: Poster Session B

[L01][DISCOURSE & DIALOGUE] *Analyzing Framing through the Casts of Characters in the News*
Dallas Card, Justin Gross, Amber Boydstun and Noah A. Smith

[L02][DISCOURSE & DIALOGUE] *The Teams Corpus and Entrainment in Multi-Party Spoken Dialogues*
Diane Litman, Susannah Paletz, Zahra Rahimi, Stefani Allegretti and Caitlin Rice

[L03][DISCOURSE & DIALOGUE] *Personalized Emphasis Framing for Persuasive Message Generation*
Tao Ding and Shimei Pan

[L04][INFORMATION EXTRACTION] *Cross Sentence Inference for Process Knowledge*
Samuel Louvan, Chetan Naik, Sadhana Kumaravel, Heeyoung Kwon, Niranjan Balasubramanian and Peter Clark

[L05][INFORMATION EXTRACTION] *Toward Socially-Infused Information Extraction: Embedding Authors, Mentions, and Entities*
Yi Yang, Ming-Wei Chang and Jacob Eisenstein

[L06][INFORMATION EXTRACTION] *Phonologically Aware Neural Model for Named Entity Recognition in Low Resource Transfer Settings*
Akash Bharadwaj, David Mortensen, Chris Dyer and Jaime Carbonell

[L07][LANGUAGE MODELING] *Long-Short Range Context Neural Networks for Language Modeling*
Youssef Oualil, Mittul Singh, Clayton Greenberg and Dietrich Klakow

[L08][LANGUAGE & VISION] *Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration*
Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-chun Zhu and Joyce Chai

[L09][LANGUAGE & VISION] *Resolving Language and Vision Ambiguities Together: Joint Segmentation & Prepositional Attachment Resolution in Captioned Scenes*
Gordon Christie, Ankit Laddha, Aishwarya Agrawal, Stanislaw Antol, Yash Goyal, Kevin Kochersberger and Dhruv Batra

Thursday, November 3, 2016 (continued)

[L10][MACHINE LEARNING] *Charagram: Embedding Words and Sentences via Character n-grams*
John Wieting, Mohit Bansal, Kevin Gimpel and Karen Livescu

[L11][MACHINE LEARNING] *Length bias in Encoder Decoder Models and a Case for Global Conditioning*
Pavel Sountsov and Sunita Sarawagi

[L12] *[TACL][Machine Learning] Comparing Apples to Apple: The Effects of Stemmers on Topic Models*
Alexandra Schofield and David Mimno

[L13][MACHINE TRANSLATION] *Does String-Based Neural MT Learn Source Syntax?*
Xing Shi, Inkit Padhi and Kevin Knight

[L14][MACHINE TRANSLATION] *Exploiting Source-side Monolingual Data in Neural Machine Translation*
Jiajun Zhang and Chengqing Zong

[L15][MACHINE TRANSLATION] *Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction*
Marcin Junczys-Dowmunt and Roman Grundkiewicz

[L16][MACHINE TRANSLATION] *Incorporating Discrete Translation Lexicons into Neural Machine Translation*
Philip Arthur, Graham Neubig and Satoshi Nakamura

[L17][MACHINE TRANSLATION] *Transfer Learning for Low-Resource Neural Machine Translation*
Barret Zoph, Deniz Yuret, Jonathan May and Kevin Knight

[L18][QUESTION ANSWERING] *MixKMeans: Clustering Question-Answer Archives*
Deepak P

[L19][QUESTION ANSWERING] *It Takes Three to Tango: Triangulation Approach to Answer Ranking in Community Question Answering*
Preslav Nakov, Lluís Màrquez and Francisco Guzmán

[L20][QUESTION ANSWERING] *Character-Level Question Answering with Attention*
Xiaodong He and David Golub

[L21][QUESTION ANSWERING] *Learning to Generate Textual Data*
Guillaume Bouchard, Pontus Stenetorp and Sebastian Riedel

[L22][QUESTION ANSWERING] *A Theme-Rewriting Approach for Generating Algebra Word Problems*
Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer and Hannaneh Hajishirzi

[L23][SENTIMENT ANALYSIS] *Context-Sensitive Lexicon Features for Neural Sentiment Analysis*
Zhiyang Teng, Duy Tin Vo and Yue Zhang

[L24][SENTIMENT ANALYSIS] *Event-Driven Emotion Cause Extraction with Corpus Construction*
Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu and Yu Zhou

[L25][SENTIMENT ANALYSIS] *Neural Sentiment Classification with User and Product Attention*
Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin and Zhiyuan Liu

[L26][SENTIMENT ANALYSIS] *Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification*
Jiacheng Xu, Danlu Chen, Xipeng Qiu and Xuanjing Huang

[L27][SENTIMENT ANALYSIS] *Deep Neural Networks with Massive Learned Knowledge*
Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov and Eric Xing

[L28][SEMANTICS] *De-Conflated Semantic Representations*
Mohammad Taher Pilehvar and Nigel Collier

[L29][SEMANTICS] *Improving Sparse Word Representations with Distributional Inference for Semantic Composition*
Thomas Kober, Julie Weeds, Jeremy Reffin and David Weir

[L30][SEMANTICS] *Modelling Interaction of Sentence Pair with Coupled-LSTMs*
Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen and Xuanjing Huang

[L31][SEMANTICS] *Universal Decompositional Semantics on Universal Dependencies*
Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins and Benjamin Van Durme

[L32][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Friends with Motives: Using Text to Infer Influence on SCOTUS*
Yanchuan Sim, Bryan Routledge and Noah A. Smith

Thursday, November 3, 2016 (continued)

[L33][SYNTAX & MORPHOLOGY] *Verb Phrase Ellipsis Resolution Using Discriminative and Margin-Infused Algorithms*
Kian Kenyon-Dean, Jackie Chi Kit Cheung and Doina Precup

[L34][SYNTAX & MORPHOLOGY] *Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser*
Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer and Noah A. Smith

[L35][SYNTAX & MORPHOLOGY] *LSTM Shift-Reduce CCG Parsing*
Wenduan Xu

[L36][SYNTAX & MORPHOLOGY] *An Evaluation of Parser Robustness for Ungrammatical Sentences*
Homa B. Hashemi and Rebecca Hwa

[L37][SYNTAX & MORPHOLOGY] *Neural Shift-Reduce CCG Semantic Parsing*
Dipendra Kumar Misra and Yoav Artzi

[L38][SYNTAX & MORPHOLOGY] *Syntactic Parsing of Web Queries*
Xiangyan Sun, Haixun Wang, Yanghua Xiao and Zhongyuan Wang

[L39][SUMMARIZATION] *Unsupervised Text Recap Extraction for TV Series*
Hongliang Yu, Shikun Zhang and Louis-Philippe Morency

[L40][TEXT MINING & APPLICATIONS] *On- and Off-Topic Classification and Semantic Annotation of User-Generated Software Requirements*
Markus Dollmann and Michaela Geierhos

[L41][TEXT MINING & APPLICATIONS] *Deceptive Review Spam Detection via Exploiting Task Relatedness and Unlabeled Data*
Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiao-Li Li and Guangxia Li

[L42][TEXT MINING & APPLICATIONS] *Regularizing Text Categorization with Clusters of Words*
Konstantinos Skianis, Francois Rousseau and Michalis Vazirgiannis

[L43][TEXT MINING & APPLICATIONS] *Deep Reinforcement Learning with a Combinatorial Action Space for Predicting Popular Reddit Threads*
Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li and Li Deng

Thursday, November 3, 2016 (continued)

[L44][TEXT MINING & APPLICATIONS] *Non-Literal Text Reuse in Historical Texts: An Approach to Identify Reuse Transformations and its Application to Bible Reuse*
Maria Moritz, Andreas Wiederhold, Barbara Pavlek, Yuri Bizzoni and Marco Büchler

[L45][TEXT MINING & APPLICATIONS] *A Graph Degeneracy-based Approach to Keyword Extraction*
Antoine Tixier, Fragkiskos Malliaros and Michalis Vazirgiannis

[L46][TEXT MINING & APPLICATIONS] *Predicting the Relative Difficulty of Single Sentences With and Without Surrounding Context*
Elliot Schumacher, Maxine Eskenazi, Gwen Frishkoff and Kevyn Collins-Thompson

[L47][TEXT MINING & APPLICATIONS] *A Neural Approach to Automated Essay Scoring*
Kaveh Taghipour and Hwee Tou Ng

[L48][TEXT MINING & APPLICATIONS] *Non-uniform Language Detection in Technical Writing*
Weibo Wang, Abidalrahman Moh'd, Aminul Islam, Axel Soto and Evangelos Milios

[L49][TEXT MINING & APPLICATIONS] *Adapting Grammatical Error Correction Based on the Native Language of Writers with Neural Network Joint Models*
Shamil Chollampatt, Duc Tam Hoang and Hwee Tou Ng

[S01][MACHINE TRANSLATION] *Orthographic Syllable as basic unit for SMT between Related Languages*
Anoop Kunchukuttan and Pushpak Bhattacharyya

[S02][TEXT MINING & APPLICATIONS] *Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge*
Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman and Regina Barzilay

[S03][INFORMATION EXTRACTION] *Supervised Keyphrase Extraction as Positive Unlabeled Learning*
Lucas Sterckx, Cornelia Caragea, Thomas Demeester and Chris Develder

[S04][INFORMATION EXTRACTION] *Learning to Answer Questions from Wikipedia Infoboxes*
Alvaro Morales, Varot Premtoon, Cordelia Avery, Sue Felshin and Boris Katz

[S05][INFORMATION EXTRACTION] *Timeline extraction using distant supervision and joint inference*
Savelie Cornegruta and Andreas Vlachos

Thursday, November 3, 2016 (continued)

[S06][INFORMATION EXTRACTION] *Combining Supervised and Unsupervised Ensembles for Knowledge Base Population*
Nazneen Fatema Rajani and Raymond Mooney

[S07][LANGUAGE & VISION] *Character Sequence Models for Colorful Words*
Kazuya Kawakami, Chris Dyer, Bryan Routledge and Noah A. Smith

[S08][LANGUAGE & VISION] *Analyzing the Behavior of Visual Question Answering Models*
Aishwarya Agrawal, Dhruv Batra and Devi Parikh

[S09][LANGUAGE & VISION] *Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text*
Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney and Kate Saenko

[S10][SEMANTICS] *Representing Verbs with Rich Contexts: an Evaluation on Verb Similarity*
Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache and Chu-Ren Huang

[S11][MACHINE LEARNING] *Speed-Accuracy Tradeoffs in Tagging with Variable-Order CRFs and Structured Sparsity*
Tim Vieira, Ryan Cotterell and Jason Eisner

[S12][MACHINE LEARNING] *Learning Robust Representations of Text*
Yitong Li, Trevor Cohn and Timothy Baldwin

[S13][MACHINE LEARNING] *Modified Dirichlet Distribution: Allowing Negative Parameters to Induce Stronger Sparsity*
Kewei Tu

[S14][MACHINE LEARNING] *Gated Word-Character Recurrent Language Model*
Yasumasa Miyamoto and Kyunghyun Cho

[S15][SYNTAX & MORPHOLOGY] *Unsupervised Word Alignment by Agreement Under ITG Constraint*
Hidetaka Kamigaito, Akihiro Tamura, Hiroya Takamura, Manabu Okumura and Eiichiro Sumita

[S16][SYNTAX & MORPHOLOGY] *Training with Exploration Improves a Greedy Stack LSTM Parser*
Miguel Ballesteros, Yoav Goldberg, Chris Dyer and Noah A. Smith

Thursday, November 3, 2016 (continued)

[S17][SEMANTICS] *Capturing Argument Relationship for Chinese Semantic Role Labeling*
Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui and Tingsong Jiang

[S18][SEMANTICS] *BrainBench: A Brain-Image Test Suite for Distributional Semantic Models*
Haoyan Xu, Brian Murphy and Alona Fyshe

[S19][SEMANTICS] *Evaluating Induced CCG Parsers on Grounded Semantic Parsing*
Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier and Mark Steedman

[S20][SEMANTICS] *Vector-space models for PPDB paraphrase ranking in context*
Marianna Apidianaki

[S21][SENTIMENT ANALYSIS] *Interpreting Neural Networks to Improve Politeness Comprehension*
Malika Aubakirova and Mohit Bansal

[S22][SENTIMENT ANALYSIS] *Does 'well-being' translate on Twitter?*
Laura Smith, Salvatore Giorgi, Rishi Solanki, Johannes Eichstaedt, H. Andrew Schwartz, Muhammad Abdul-Mageed, Anneke Buffone and Lyle Ungar

[S23][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Beyond Canonical Texts: A Computational Analysis of Fanfiction*
Smitha Milli and David Bamman

[S24][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Using Syntactic and Semantic Context to Explore Psychodemographic Differences in Self-reference*
Masoud Rouhizadeh, Lyle Ungar, Anneke Buffone and H. Andrew Schwartz

[S25][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Learning to Identify Metaphors from a Corpus of Proverbs*
Gözde Özbal, Carlo Strapparava, Serra Sinem Tekiroglu and Daniele Pighin

[S26][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *An Embedding Model for Predicting Roll-Call Votes*
Peter Kraft, Hirsh Jain and Alexander M. Rush

[S27][SPOKEN LANGUAGE PROCESSING] *Natural Language Model Re-usability for Scaling to Different Domains*
Young-Bum Kim, Alexandre Rochette and Ruhi Sarikaya

[S28][SPOKEN LANGUAGE PROCESSING] *Leveraging Sentence-level Information with Encoder LSTM for Semantic Slot Filling*
Gakuto Kurata, Bing Xiang, Bowen Zhou and Mo Yu

Thursday, November 3, 2016 (continued)

[S29][SUMMARIZATION] *AMR-to-text generation as a Traveling Salesman Problem*
Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang and Daniel Gildea

[S30][TEXT MINING & APPLICATIONS] *Learning to Capitalize with Character-Level Recurrent Neural Networks: An Empirical Study*
Raymond Hendy Susanto, Hai Leong Chieu and Wei Lu

[S31][TEXT MINING & APPLICATIONS] *The Effects of the Content of FOMC Communications on US Treasury Rates*
Christopher Rohlfs, Sunandan Chakraborty and Lakshminarayanan Subramanian

[S32][TEXT MINING & APPLICATIONS] *Learning to refine text based recommendations*
Youyang Gu, Tao Lei, Regina Barzilay and Tommi Jaakkola

[S33][TEXT MINING & APPLICATIONS] *There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction*
Courtney Napoles, Keisuke Sakaguchi and Joel Tetreault

[S34][SOCIAL MEDIA & COMPUTATIONAL SOCIAL SCIENCE] *Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change*
William L. Hamilton, Jure Leskovec and Dan Jurafsky

Friday, November 4, 2016

07:30–17:30 Registration Day 3

08:00–09:00 ***Morning Coffee***

09:00–10:00 Session P9: Plenary Session: Invited Talk by Andreas Stolcke

09:00–10:00 *You Talking to Me? Speech-based and Multimodal Approaches for Human versus Computer Addressee Detection*
Andreas Stolcke

10:00–10:30 ***Coffee Break***

Friday, November 4, 2016 (continued)

10:30–12:10 Session 7A: Dialogue Systems (Long Papers)

10:30–10:55 *How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation*
Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin and Joelle Pineau

10:55–11:20 *Addressee and Response Selection for Multi-Party Conversation*
Hiroki Ouchi and Yuta Tsuboi

11:20–11:45 *Nonparametric Bayesian Models for Spoken Language Understanding*
Kei Wakabayashi, Johane Takeuchi, Kotaro Funakoshi and Mikio Nakano

11:45–12:10 *Conditional Generation and Snapshot Learning in Neural Dialogue Systems*
Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke and Steve Young

10:30–12:10 Session 7B: Semantic Similarity (Long Papers)

10:30–10:55 *Relations such as Hypernymy: Identifying and Exploiting Hearst Patterns in Distributional Vectors for Lexical Entailment*
Stephen Roller and Katrin Erk

10:55–11:20 *SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity*
Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart and Anna Korhonen

11:20–11:45 *POLY: Mining Relational Paraphrases from Multilingual Sentences*
Adam Grycner and Gerhard Weikum

11:45–12:10 *Exploiting Sentence Similarities for Better Alignments*
Tao Li and Vivek Srikumar

Friday, November 4, 2016 (continued)

10:30–12:10 Session 7C: Dependency Parsing (Long + TACL Papers)

10:30–10:55 *Bi-directional Attention with Agreement for Dependency Parsing*
Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao and Li Deng

10:55–11:20 *[TACL] The Galactic Dependencies Treebanks: Getting More Data by Synthesizing New Languages*
Dingquan Wang and Jason Eisner

11:20–11:45 *[TACL] Easy-First Dependency Parsing with Hierarchical Tree LSTMs*
Eliyahu Kiperwasser and Yoav Goldberg

11:45–12:10 *Anchoring and Agreement in Syntactic Annotations*
Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen and Boris Katz

12:10–13:40 ***Lunch***

13:40–15:25 Session 8A: Short Paper Oral Session I

13:40–13:55 *Tense Manages to Predict Implicative Behavior in Verbs*
Ellie Pavlick and Chris Callison-Burch

13:55–14:10 *Who did What: A Large-Scale Person-Centered Cloze Dataset*
Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel and David McAllester

14:10–14:25 *Building compositional semantics and higher-order inference system for a wide-coverage Japanese CCG parser*
Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao and Daisuke Bekki

14:25–14:40 *Learning to Generate Compositional Color Descriptions*
Will Monroe, Noah D. Goodman and Christopher Potts

14:40–14:55 *A Decomposable Attention Model for Natural Language Inference*
Ankur Parikh, Oscar Täckström, Dipanjan Das and Jakob Uszkoreit

14:55–15:10 *Deep Reinforcement Learning for Mention-Ranking Coreference Models*
Kevin Clark and Christopher D. Manning

Friday, November 4, 2016 (continued)

15:10–15:25 *A Stacking Gated Neural Architecture for Implicit Discourse Relation Classification*
Lianhui Qin, Zhisong Zhang and Hai Zhao

13:40–15:25 Session 8B: Short Paper Oral Session II

13:40–13:55 *Insertion Position Selection Model for Flexible Non-Terminals in Dependency Tree-to-Tree Machine Translation*
Toshiaki Nakazawa, John Richardson and Sadao Kurohashi

13:55–14:10 *Why Neural Translations are the Right Length*
Xing Shi, Kevin Knight and Deniz Yuret

14:10–14:25 *Supervised Attentions for Neural Machine Translation*
Haitao Mi, Zhiguo Wang and Abe Ittycheriah

14:25–14:40 *Learning principled bilingual mappings of word embeddings while preserving monolingual invariance*
Mikel Artetxe, Gorka Labaka and Eneko Agirre

14:40–14:55 *Measuring the behavioral impact of machine translation quality improvements with A/B testing*
Ben Russell and Duncan Gillespie

14:55–15:10 *Creating a Large Benchmark for Open Information Extraction*
Gabriel Stanovsky and Ido Dagan

15:10–15:25 *Bilingually-constrained Synthetic Data for Implicit Discourse Relation Recognition*
Changxing Wu, xiaodong shi, Yidong Chen, Yanzhou Huang and jinsong su

13:40–15:25 Session 8C: Short Paper Oral Session III

13:40–13:55 *Transition-Based Dependency Parsing with Heuristic Backtracking*
Jacob Buckman, Miguel Ballesteros and Chris Dyer

13:55–14:10 *Word Ordering Without Syntax*
Allen Schmaltz, Alexander M. Rush and Stuart Shieber

14:10–14:25 *Morphological Segmentation Inside-Out*
Ryan Cotterell, Arun Kumar and Hinrich Schütze

14:25–14:40 *Parsing as Language Modeling*
Do Kook Choe and Eugene Charniak

14:40–14:55 *Human-in-the-Loop Parsing*
Luheng He, Julian Michael, Mike Lewis and Luke Zettlemoyer

14:55–15:10 *Unsupervised Timeline Generation for Wikipedia History Articles*
Sandro Bauer and Simone Teufel

15:10–15:25 *Encoding Temporal Information for Time-Aware Link Prediction*
Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang and Zhifang Sui

15:25–15:50 *Coffee Break*

Friday, November 4, 2016 (continued)

15:50–17:25 Session P10: Plenary Session: Best Paper

15:50–15:55 *Introduction to Best Papers*
Program Chairs

15:55–16:20 *Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning*
Karthik Narasimhan, Adam Yala and Regina Barzilay

16:20–16:45 *Global Neural CCG Parsing with Optimality Guarantees*
Kenton Lee, Mike Lewis and Luke Zettlemoyer

16:45–17:00 *Learning a Lexicon and Translation Model from Phoneme Lattices*
Oliver Adams, Graham Neubig, Trevor Cohn, Steven Bird, Quoc Truong Do and Satoshi Nakamura

17:00–17:25 *SQuAD: 100,000+ Questions for Machine Comprehension of Text*
Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev and Percy Liang

17:25–17:45 Session P11: Plenary Session: Closing Remarks

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

List of Papers

Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles

James Cross and **Liang Huang**
School of EECS, Oregon State University, Corvallis, OR, USA
{james.henry.cross.iii, liang.huang.sh}@gmail.com

Abstract

Parsing accuracy using efficient greedy transition systems has improved dramatically in recent years thanks to neural networks. Despite striking results in dependency parsing, however, neural models have not surpassed state-of-the-art approaches in constituency parsing. To remedy this, we introduce a new shift-reduce system whose stack contains merely sentence spans, represented by a bare minimum of LSTM features. We also design the first provably optimal dynamic oracle for constituency parsing, which runs in amortized $O(1)$ time, compared to $O(n^3)$ oracles for standard dependency parsing. Training with this oracle, we achieve the best F_1 scores on both English and French of any parser that does not use reranking or external data.

1 Introduction

Parsing is an important problem in natural language processing which has been studied extensively for decades. Between the two basic paradigms of parsing, constituency parsing, the subject of this paper, has in general proved to be the more difficult than dependency parsing, both in terms of accuracy and the run time of parsing algorithms.

There has recently been a huge surge of interest in using neural networks to make parsing decisions, and such models continue to dominate the state of the art in dependency parsing (Andor et al., 2016). In constituency parsing, however, neural approaches are still behind the state-of-the-art (Carreras et al., 2008; Shindo et al., 2012; Thang et al., 2015); see more details in Section 5.

To remedy this, we design a new parsing framework that is more suitable for constituency parsing, and that can be accurately modeled by neural networks. Observing that constituency parsing is primarily focused on sentence spans (rather than individual words, as is dependency parsing), we propose a novel adaptation of the shift-reduce system which reflects this focus. In this system, the stack consists of sentence spans rather than partial trees. It is also factored into two types of parser actions, structural and label actions, which alternate during a parse. The structural actions are a simplified analogue of shift-reduce actions, omitting the directionality of reduce actions, while the label actions directly assign nonterminal symbols to sentence spans.

Our neural model processes the sentence once for each parse with a recurrent network. We represent parser configurations with a very small number of span features (4 for structural actions and 3 for label actions). Extending Wang and Chang (2016), each span is represented as the difference of recurrent output from multiple layers in each direction. No pre-trained embeddings are required.

We also extend the idea of dynamic oracles from dependency to constituency parsing. The latter is significantly more difficult than the former due to F_1 being a combination of precision and recall (Huang, 2008), and yet we propose a simple and extremely efficient oracle (amortized $O(1)$ time). This oracle is proved optimal for F_1 as well as both of its components, precision and recall. Trained with this oracle, our parser achieves what we believe to be the best results for any parser without reranking which was trained only on the Penn Treebank and the French Treebank, despite the fact that it is not only linear-time, but also strictly greedy.

We make the following main contributions:

- A novel factored transition parsing system where the stack elements are sentence spans rather than partial trees (Section 2).
- A neural model where sentence spans are represented as differences of output from a multi-layer bi-directional LSTM (Section 3).
- The first provably optimal dynamic oracle for

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1–11,
Austin, Texas, November 1-5, 2016.

constituency parsing which is also extremely efficient (amortized $O(1)$ time) (Section 4).

- The best F_1 scores of any single-model, closed training set, parser for English and French.

We are also publicly releasing the source code for one implementation of our parser.[1]

2 Parsing System

We present a new transition-based system for constituency parsing whose fundamental unit of computation is the sentence span. It uses a stack in a similar manner to other transition systems, except that the stack contains sentence spans with no requirement that each one correspond to a partial tree structure during a parse.

The parser alternates between two types of actions, structural and label, where the structural actions follow a path to make the stack spans correspond to sentence phrases in a bottom-up manner, while the label actions optionally create tree brackets for the top span on the stack. There are only two structural actions: *shift* is the same as other transition systems, while *combine* merges the top two sentence spans. The latter is analogous to a reduce action, but it does not immediately create a tree structure and is non-directional. Label actions do create a partial tree on top of the stack by assigning one or more non-terminals to the topmost span.

Except for the use of spans, this factored approach is similar to the odd-even parser from Mi and Huang (2015). The fact that stack elements do not have to be tree-structured, however, means that we can create productions with arbitrary arity, and no binarization is required either for training or parsing. This also allows us to remove the directionality inherent in the shift-reduce system, which is at best an imperfect fit for constituency parsing. We do follow the practice in that system of labeling unary chains of non-terminals with a single action, which means our parser uses a fixed number of steps, $(4n-2)$ for a sentence of n words.

Figure 1 shows the formal deductive system for this parser. The stack σ is modeled as a list of strictly increasing integers whose first element is always zero. These numbers are word boundaries which define the spans on the stack. In a slight abuse of notation, however, we sometimes think of it as a list of pairs (i, j), which are the actual sentence spans, i.e., every consecutive pair of indices on the stack, initially empty. We represent stack spans by trapezoids ($_i\!\!\smile\!\!\!\frown_j$) in the figures to emphasize that they may or not have tree stucture.

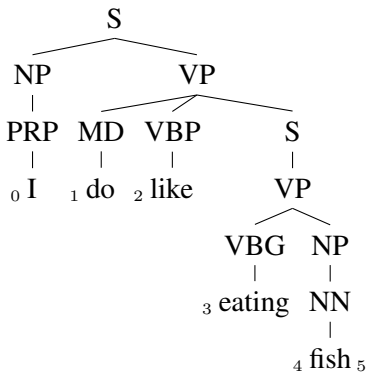
input:	$w_0 \dots w_{n-1}$
axiom:	$\langle 0,\ [0],\ \emptyset \rangle$
goal:	$\langle 2(2n-1),\ [0, n],\ t \rangle$
sh	$\dfrac{\langle z,\ \sigma \,\vert\, j,\ t \rangle}{\langle z+1,\ \sigma \,\vert\, j \,\vert\, j{+}1,\ t \rangle}$ $j < n$, even z
comb	$\dfrac{\langle z,\ \sigma \,\vert\, i \,\vert\, k \,\vert\, j,\ t \rangle}{\langle z+1,\ \sigma \,\vert\, i \,\vert\, j,\ t \rangle}$ even z
label-X	$\dfrac{\langle z,\ \sigma \,\vert\, i \,\vert\, j,\ t \rangle}{\langle z+1,\ \sigma \,\vert\, i \,\vert\, j,\ t \cup \{ {}_iX_j \} \rangle}$ odd z
nolabel	$\dfrac{\langle z,\ \sigma \,\vert\, i \,\vert\, j,\ t \rangle}{\langle z+1,\ \sigma \,\vert\, i \,\vert\, j,\ t \rangle}$ $z < (4n-1)$, odd z

Figure 1: Deductive system for the Structure/Label transition parser. The stack σ is represented as a list of integers where the span defined by each consecutive pair of elements is a sentence segment on the stack. Each X is a nonterminal symbol or an ordered unary chain. The set t contains labeled spans of the form ${}_iX_j$, which at the end of a parse, fully define a parse tree.

The parser alternates between structural actions and label actions according to the parity of the parser step z. In even steps, it takes a structural action, either combining the top two stack spans, which requires at least two spans on the stack, or introducing a new span of unit length, as long as the entire sentence is not already represented on the stack

In odd steps, the parser takes a label action. One possibility is labeling the top span on the stack, (i, j) with either a nonterminal label or an ordered unary chain (since the parser has only one opportunity to label any given span). Taking no action, designated nolabel, is also a possibility. This is essentially a null operation except that it returns the parser to an even step, and this action reflects the decision that (i, j) is not a (complete) labeled phrase in the tree. In the final step, $(4n-2)$, nolabel is not allowed

[1] code: `https://github.com/jhcross/span-parser`



(a) gold parse tree

steps	structural action	label action	stack after	bracket
1–2	sh(I/PRP)	label-NP	$_0\frown_1$	$_0$NP$_1$
3–4	sh(do/MD)	nolabel	$_0\frown_1\frown_2$	
5–6	sh(like/VBP)	nolabel	$_0\frown_1\frown_2\frown_3$	
7–8	comb	nolabel	$_0\frown_1\frown_3$	
9–10	sh(eating/VBG)	nolabel	$_0\frown_1\frown_3\frown_4$	
11–12	sh(fish/NN)	label-NP	$_0\frown_1\frown_3\frown_4\frown_5$	$_4$NP$_5$
13–14	comb	label-S-VP	$_0\frown_1\frown_3\frown_5$	$_3$S$_5$, $_3$VP$_5$
15–16	comb	label-VP	$_0\frown_1\frown_5$	$_1$VP$_5$
17–18	comb	label-S	$_0\frown_5$	$_0$S$_5$

(b) static oracle actions

Figure 2: The running example. It contains one ternary branch and one unary chain (S-VP), and NP-PRP-I and NP-NN-fish are *not* unary chains in our system. Each stack is just a list of numbers but is visualized with spans here.

since the parser must produce a tree.

Figure 2 shows a complete example of applying this parsing system to a very short sentence (*"I do like eating fish"*) that we will use throughout this section and the next. The action in step 2 is label-NP because "I" is a one-word noun phrase (parts of speech are taken as input to our parser, though it could easily be adapted to include POS tagging in label actions). If a single word is not a complete phrase (e.g., "do"), then the action after a shift is nolabel.

The ternary branch in this tree (VP → MD VBP S) is produced by our parser in a straightforward manner: after the phrase "do like" is combined in step 7, no label is assigned in step 8, successfully delaying the creation of a bracket until the verb phrase is fully formed on the stack. Note also that the unary production in the tree is created with a single action, label-S-VP, in step 14.

The static oracle to train this parser simply consists of taking actions to generate the gold tree with a "short-stack" heuristic, meaning combine first whenever combine and shift are both possible.

3 LSTM Span Features

Long short-term memory networks (LSTM) are a type of recurrent neural network model proposed by Hochreiter and Schmidhuber (1997) which are very effective for modeling sequences. They are able to capture and generalize from interactions among their sequential inputs even when separated by a long distance, and thus are a natural fit for analyzing natural language. LSTM models have proved to be a powerful tool for many learning tasks in natural language, such as language modeling (Sundermeyer et al., 2012) and translation (Sutskever et al., 2014).

LSTMs have also been incorporated into parsing in a variety of ways, such as directly encoding an entire sentence (Vinyals et al., 2015), separately modeling the stack, buffer, and action history (Dyer et al., 2015), to encode words based on their character forms (Ballesteros et al., 2015), and as an element in a recursive structure to combine dependency subtrees with their left and right children (Kiperwasser and Goldberg, 2016a).

For our parsing system, however, we need a way to model arbitrary sentence spans in the context of the rest of the sentence. We do this by representing each sentence span as the elementwise difference of the vector outputs of the LSTM outputs at different time steps, which correspond to word boundaries. If the sequential output of the recurrent network for the sentence is $f_0, ..., f_n$ in the forward direction and $b_n, ..., b_0$ in the backward direction then the span (i, j) would be represented as the concatenation of the vector differences $(f_j - f_i)$ and $(b_i - b_j)$.

The spans are represented using output from both backward and forward LSTM components, as can be seen in Figure 3. This is essentially the LSTM-Minus feature representation described by Wang and Chang (2016) extended to the bi-directional case. In initial experiments, we found that there was essentially no difference in performance between using the difference features and concatenating all end-

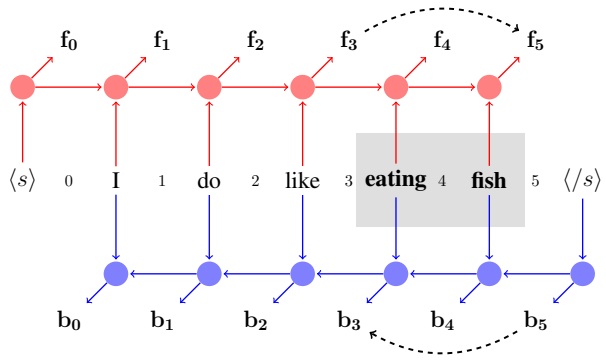


Figure 3: Word spans are modeled by differences in LSTM output. Here the span $_3$ *eating fish* $_5$ is represented by the vector differences $(\mathbf{f_5} - \mathbf{f_3})$ and $(\mathbf{b_3} - \mathbf{b_5})$. The forward difference corresponds to LSTM-Minus (Wang and Chang, 2016).

point vectors, but our approach is almost twice as fast.

This model allows a sentence to be processed once, and then the same recurrent outputs can be used to compute span features throughout the parse. Intuitively, this allows the span differences to learn to represent the sentence spans in the context of the rest of the sentence, not in isolation (especially true for LSTM given the extra hidden recurrent connection, typically described as a "memory cell"). In practice, we use a two-layer bi-directional LSTM, where the input to the second layer combines the forward and backward outputs from the first layer at that time step. For each direction, the components from the first and second layers are concatenated to form the vectors which go into the span features. See Cross and Huang (2016) for more details on this approach.

For the particular case of our transition constituency parser, we use only four span features to determine a structural action, and three to determine a label action, in each case partitioning the sentence exactly. The reason for this is straightforward: when considering a structural action, the top two spans on the stack must be considered to determine whether they should be combined, while for a label action, only the top span on the stack is important, since that is the candidate for labeling. In both cases the remaining sentence prefix and suffix are also included. These features are shown in Table 1.

The input to the recurrent network at each time step consists of vector embeddings for each word

Action	Stack	LSTM Span Features
Structural	$\sigma \mid i \mid k \mid j$	0 i k j n
Label	$\sigma \mid i \mid j$	0 i j n

Table 1: Features used for the parser. No label or tree-structure features are required.

and its part-of-speech tag. Parts of speech are predicted beforehand and taken as input to the parser, as in much recent work in parsing. In our experiments, the embeddings are randomly initialized and learned from scratch together with all other network weights, and we would expect further performance improvement from incorporating embeddings pretrained from a large external corpus.

The network structure after the the span features consists of a separate multilayer perceptron for each type of action (structural and label). For each action we use a single hidden layer with rectified linear (ReLU) activation. The model is trained on a per-action basis using a single correct action for each parser state, with a negative log softmax loss function, as in Chen and Manning (2014).

4 Dynamic Oracle

The baseline method of training our parser is what is known as a static oracle: we simply generate the sequence of actions to correctly parse each training sentence, using a short-stack heuristic (i.e., combine first whenever there is a choice of shift and combine). This method suffers from a well-documeted problem, however, namely that it only "prepares" the model for the situation where no mistakes have been made during parsing, an inevitably incorrect assumption in practice. To alleviate this problem, Goldberg and Nivre (2013) define a dynamic oracle to return the best possible action(s) at any arbitrary configuration.

In this section, we introduce an easy-to-compute optimal dynamic oracle for our constituency parser. We will first define some concepts upon which the dynamic oracle is built and then show how optimal actions can be very efficiently computed using this framework. In broad strokes, in any arbitrary parser configuration c there is a set of brackets $t^*(c)$ from the gold tree which it is still possible to reach. By following dynamic oracle actions, all of those brackets and only those brackets will be predicted.

Even though proving the optimality of our dynamic oracle (Sec. 4.3) is involved, computing the oracle actions is extremely simple (Secs. 4.2) and efficient (Sec. 4.4).

4.1 Preliminaries and Notations

Before describing the computation of our dynamic oracle, we first need to rigorously establish the desired optimality of dynamic oracle. The structure of this framework follows Goldberg et al. (2014).

Definition 1. We denote $c \vdash_\tau c'$ iff. c' is the result of action τ on configuration c, also denoted functionally as $c' = \tau(c)$. We denote $\vdash$ to be the union of $\vdash_\tau$ for all actions τ, and $\vdash^*$ to be the reflexive and transitive closure of $\vdash$.

Definition 2 (descendant/reachable trees)**.** We denote $\mathcal{D}(c)$ to be the set of final descendant trees derivable from c, i.e., $\mathcal{D}(c) = \{t \mid c \vdash^* \langle z,\ \sigma,\ t\rangle\}$. This set is also called "reachable trees" from c.

Definition 3 (F_1)**.** We define the standard F_1 metric of a tree t with respect to gold tree t_G as $\mathrm{F}_1(t) = \frac{2rp}{r+p}$, where $r = \frac{|t \cap t_\mathrm{G}|}{|t_\mathrm{G}|}, p = \frac{|t \cap t_\mathrm{G}|}{|t|}$.

The following two definitions are similar to those for dependency parsing by Goldberg et al. (2014).

Definition 4. We extend the F_1 function to configurations to define the maximum possible F_1 from a given configuration: $\mathrm{F}_1(c) = \max_{t_1 \in \mathcal{D}(c)} \mathrm{F}_1(t_1)$.

Definition 5 (oracle)**.** We can now define the desired dynamic oracle of a configuration c to be the set of actions that retrain the optimal F_1:

$$\mathit{oracle}(c) = \{\tau \mid \mathrm{F}_1(\tau(c)) = \mathrm{F}_1(c)\}.$$

This abstract oracle is implemented by $\mathit{dyna}(\cdot)$ in Sec. 4.2, which we prove to be correct in Sec. 4.3.

Definition 6 (span encompassing)**.** We say span (i, j) is encompassed by span (p, q), notated $(i, j) \preceq (p, q)$, iff. $p \le i < j \le q$.

Definition 7 (strict encompassing)**.** We say span (i, j) is strictly encompassed by span (p, q), notated $(i, j) \prec (p, q)$, iff. $(i, j) \preceq (p, q)$ and $(i, j) \neq (p, q)$. We then extend this relation from spans to brackets, and notate ${}_iX_j \prec {}_pY_q$ iff. $(i, j) \prec (p, q)$.

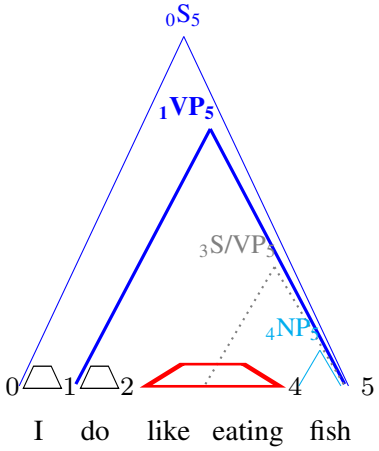


Figure 4: Reachable brackets (w.r.t. gold tree in Fig. 1) for $c = \langle 10,\ [0, 1, 2, 4],\ \{{}_0\mathrm{NP}_1\}\rangle$ which mistakenly combines "like eating". Trapezoids indicate stack spans (the top one in red), and solid triangles denote reachable brackets, with $\mathit{left}(c)$ in blue and $\mathit{right}(c)$ in cyan. The next reachable bracket, $\mathit{next}(c) = {}_1\mathbf{VP}_5$, is in bold. Brackets ${}_3\mathrm{VP}_5$ and ${}_3\mathrm{S}_5$ (in dotted triangle) cross the top span (thus unreachable), and ${}_0\mathrm{NP}_1$ is already recognized (thus not in $\mathit{reach}(c)$ either).

We next define a central concept, *reachable brackets*, which is made up of two parts, the left ones $\mathit{left}(c)$ which encompass (i, j) without crossing any stack spans, and the right ones $\mathit{right}(c)$ which are completely on the queue. See Fig. 4 for examples.

Definition 8 (reachable brackets)**.** For any configuration $c = \langle z,\ \sigma \,|\, i \,|\, j,\ t\rangle$, we define the set of reachable gold brackets (with respect to gold tree t_G) as

$$\mathit{reach}(c) = \mathit{left}(c) \cup \mathit{right}(c)$$

where the left- and right-reachable brackets are

$$\begin{aligned}\mathit{left}(c) &= \{{}_pX_q \in t_\mathrm{G} \mid (i, j) \prec (p, q),\ p \in \sigma \,|\, i\}\\ \mathit{right}(c) &= \{{}_pX_q \in t_\mathrm{G} \mid p \ge j\}\end{aligned}$$

for even z, with the $\prec$ replaced by $\preceq$ for odd z.

Special case (initial): $\mathit{reach}(\langle 0,\ [0],\ \emptyset\rangle) = t_\mathrm{G}$.

The notation $p \in \sigma \mid i$ simply means (p, q) does not "cross" any bracket on the stack. Remember our stack is just a list of span boundaries, so if p coincides with one of them, (p, q)'s left boundary is not crossing and its right boundary q is not crossing either since $q \ge j$ due to $(i, j) \prec (p, q)$.

Also note that $\mathit{reach}(c)$ is strictly disjoint from t, i.e., $\mathit{reach}(c) \cap t = \emptyset$ and $\mathit{reach}(c) \subseteq t_\mathrm{G} - t$. See Figure 6 for an illustration.

Definition 9 (next bracket)**.** For any configuration $c = \langle z,\ \sigma\,|\,i\,|\,j,\ t\rangle$, the next reachable gold bracket (with respect to gold tree t_{G}) is the smallest reachable bracket (strictly) encompassing (i, j):

$$next(c) = \min_{\prec} left(c).$$

4.2 Structural and Label Oracles

For an even-step configuration $c = \langle z,\ \sigma \mid i \mid j,\ t\rangle$, we denote the next reachable gold bracket $next(c)$ to be ${}_pX_q$, and define the dynamic oracle to be:

$$dyna(c) = \begin{cases} \{\mathsf{sh}\} & \text{if } p = i \text{ and } q > j \\ \{\mathsf{comb}\} & \text{if } p < i \text{ and } q = j \\ \{\mathsf{sh}, \mathsf{comb}\} & \text{if } p < i \text{ and } q > j \end{cases} \quad (1)$$

As a special case $dyna(\langle 0,\ [0],\ \emptyset\rangle) = \{\mathsf{sh}\}$.

Figure 5 shows examples of this policy. The key insight is, if you follow this policy, you will *not* miss the next reachable bracket, but if you do not follow it, you certainly will. We formalize this fact below (with proof omitted due to space constraints) which will be used to prove the central results later.

Lemma 1. *For any configuration c, for any $\tau \in dyna(c)$, we have $reach(\tau(c)) = reach(c)$; for any $\tau' \notin dyna(c)$, we have $reach(\tau(c)) \subsetneq reach(c)$.*

The label oracles are much easier than structural ones. For an odd-step configuration $c = \langle z,\ \sigma \mid i \mid j,\ t\rangle$, we simply check if (i, j) is a valid span in the gold tree t_G and if so, label it accordingly, otherwise no label. More formally,

$$dyna(c) = \begin{cases} \{\mathsf{label}\text{-}X\} & \text{if some } {}_iX_j \in t_G \\ \{\mathsf{nolabel}\} & \text{otherwise} \end{cases} \quad (2)$$

4.3 Correctness

To show the optimality of our dynamic oracle, we begin by defining a special tree $t^*(c)$ and show that it is optimal among all trees reachable from configuration c. We then show that following our dynamic oracle (Eqs. 1–2) from c will lead to $t^*(c)$.

Definition 10 ($t^*(c)$)**.** For any configuration $c = \langle z,\ \sigma,\ t\rangle$, we define the optimal tree $t^*(c)$ to include all reachable gold brackets and nothing else. More formally, $t^*(c) = t \cup reach(c)$.

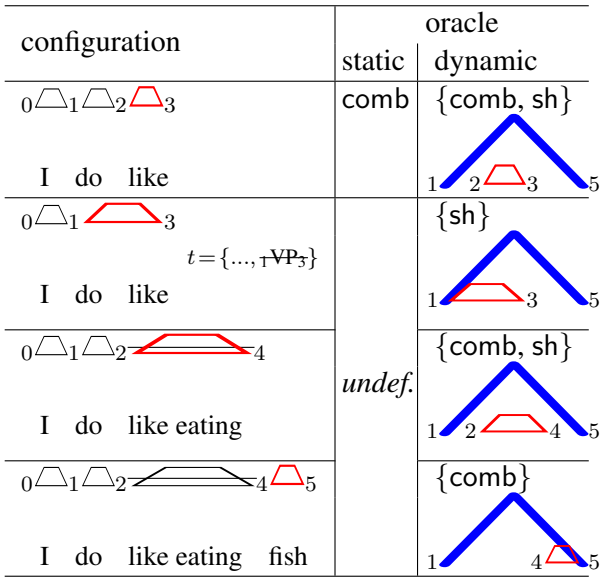


Figure 5: Dynamic oracle with respect to the gold parse in Fig. 2. The last three examples are off the gold path with strike out indicating structural or label mistakes. Trapezoids denote stack spans (top one in red) and the blue triangle denotes the next reachable bracket $next(c)$ which is ${}_1\mathrm{VP}_5$ in all cases.

We can show by induction that $t^*(c)$ is attainable:

Lemma 2. *For any configuration c, the optimal tree is a descendant of c, i.e., $t^*(c) \in \mathcal{D}(c)$.*

The following Theorem shows that $t^*(c)$ is indeed the best possible tree:

Theorem 1 (optimality of t^*)**.** *For any configuration c, $\mathrm{F}_1(t^*(c)) = \mathrm{F}_1(c)$.*

Proof. (SKETCH) Since $t^*(c)$ adds all possible additional gold brackets (the brackets in $reach(c)$), it is not possible to get higher recall. Since it adds no incorrect brackets, it is not possible to get higher pre-

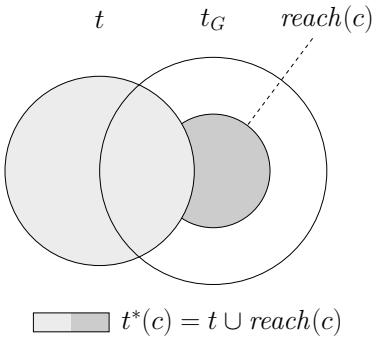


Figure 6: The optimal tree $t^*(c)$ adds all reachable brackets and nothing else. Note that $reach(c)$ and t are disjoint.

cision. Since F_1 is the harmonic mean of precision and recall, it also leads to the best possible F_1. □

Corollary 1. *For any $c = \langle z,\ \sigma,\ t\rangle$, for any $t' \in \mathcal{D}(c)$ and $t' \neq t^*(c)$, we have $\mathrm{F}_1(t') < \mathrm{F}_1(c)$.*

We now need a final lemma about the policy $dyna(\cdot)$ (Eqs. 1–2) before proving the main result.

Lemma 3. *From any $c = \langle z,\ \sigma,\ t\rangle$, for any action $\tau \in dyna(c)$, we have $t^*(\tau(c)) = t^*(c)$. For any action $\tau' \notin dyna(c)$, we have $t^*(\tau'(c)) \neq t^*(c)$.*

Proof. (SKETCH) By case analysis on even/odd z. □

We are now able to state and prove the main theoretical result of this paper (using Lemma 3, Theorem 1 and Corollary 1):

Theorem 2. *The function $dyna(\cdot)$ in Eqs. (1–2) satisfies the requirement of a dynamic oracle (Def. 5):*

$dyna(c) = oracle(c)$ for any configuration c.

4.4 Implementation and Complexity

For any configuration, our dynamic oracle can be computed in **amortized constant** time since there are only $O(n)$ gold brackets and thus bounding $|reach(c)|$ and the choice of $next(c)$. After each action, $next(c)$ either remains unchanged, or in the case of being crossed by a structural action or mislabeled by a label action, needs to be updated. This update is simply tracing the parent link to the next smallest gold bracket repeatedly until the new bracket encompasses span (i, j). Since there are at most $O(n)$ choices of $next(c)$ and there are $O(n)$ steps, the per-step cost is amortized constant time. Thus our dynamic oracle is much faster than the super-linear time oracle for arc-standard dependency parsing in Goldberg et al. (2014).

5 Related Work

Neural networks have been used for constituency parsing in a number of previous instances. For example, Socher et al. (2013) learn a recursive network that combines vectors representing partial trees, Vinyals et al. (2015) adapt a sequence-to-sequence model to produce parse trees, Watanabe and Sumita (2015) use a recursive model applying a shift-reduce system to constituency parsing with beam search, and Dyer et al. (2016) adapt the Stack-LSTM dependency parsing approach to this task. Durrett and Klein (2015) combine both neural and sparse features for a CKY parsing system. Our own previous work (Cross and Huang, 2016) use a recurrent sentence representation in a head-driven transition system which allows for greedy parsing but does not achieve state-of-the-art results.

The concept of "oracles" for constituency parsing (as the tree that is most similar to t_{G} among all possible trees) was first defined and solved by Huang (2008) in bottom-up parsing. In transition-based parsing, the dynamic oracle for shift-reduce dependency parsing costs worst-case $O(n^3)$ time (Goldberg et al., 2014). On the other hand, after the submission of our paper we became aware of a parallel work (Coavoux and Crabbé, 2016) that also proposed a dynamic oracle for their own incremental constituency parser. However, it is not optimal due to dummy non-terminals from binarization.

Network architecture	
Word embeddings	50
Tag embeddings	20
Morphological embeddings†	10
LSTM layers	2
LSTM units	200 / direction
ReLU hidden units	200 / action type
Training settings	
Embedding intialization	random
Training epochs	10
Minibatch size	10 sentences
Dropout (on LSTM output)	$p = 0.5$
ADADELTA parameters	$\rho = 0.99$, $\epsilon = 1 \times 10^{-7}$

Table 2: Hyperparameters. †French only.

6 Experiments

We present experiments on both the Penn English Treebank (Marcus et al., 1993) and the French Treebank (Abeillé et al., 2003). In both cases, all state-action training pairs for a given sentence are used at the same time, greatly increasing training speed since all examples for the same sentence share the same forward and backward pass through the recurrent part of the network. Updates are performed in minibatches of 10 sentences, and we shuffle the training sentences before each epoch. The results we report are trained for 10 epochs.

The only regularization which we employ during training is dropout (Hinton et al., 2012), which is applied with probability 0.5 to the recurrent outputs. It is applied separately to the input to the second LSTM layer for each sentence, and to the input to the ReLU hidden layer (span features) for each state-action pair. We use the ADADELTA method (Zeiler, 2012) to schedule learning rates for all weights. All of these design choices are summarized in Table 2.

In order to account for unknown words during training, we also adopt the strategy described by Kiperwasser and Goldberg (2016b), where words in the training set are replaced with the unknown-word symbol UNK with probability $p_{unk} = \frac{z}{z+f(w)}$ where $f(w)$ is the number of times the word appears in the training corpus. We choose the parameter z so that the training and validation corpora have approximately the same proportion of unknown words. For the Penn Treebank, for example, we used $z = 0.8375$ so that both the validation set and the (rest of the) training set contain approximately 2.76% unknown words. This approach was helpful but not critical, improving F_1 (on dev) by about 0.1 over training without any unknown words.

6.1 Training with Dynamic Oracle

The most straightforward use of dynamic oracles to train a neural network model, where we collect all action examples for a given sentence before updating, is "training with exploration" as proposed by Goldberg and Nivre (2013). This involves parsing each sentence according to the current model and using the oracle to determine correct actions for training. We saw very little improvement on the Penn treebank validation set using this method, however. Based on the parsing accuracy on the training sentences, this appears to be due to the model overfitting the training data early during training, thus negating the benefit of training on erroneous paths.

Accordingly, we also used a method recently proposed by Ballesteros et al. (2016), which specifically addresses this problem. This method introduces stochasticity into the training data parses by randomly taking actions according to the softmax distribution over action scores. This introduces realistic mistakes into the training parses, which we found was also very effective in our case, leading to higher F_1 scores, though it noticeably sacrifices recall in favor of precision.

This technique can also take a parameter α to flatten or sharpen the raw softmax distribution. The results on the Penn treebank development set for various values of α are presented in Table 3. We were surprised that flattening the distribution seemed to be the least effective, as training accuracy (taking into account sampled actions) lagged somewhat behind validation accuracy. Ultimately, the best results were for $\alpha = 1$, which we used for final testing.

Model	LR	LP	F_1
Static Oracle	91.34	91.43	91.38
Dynamic Oracle	91.14	91.61	91.38
+ Explore ($\alpha=0.5$)	90.59	92.18	91.38
+ Explore ($\alpha=1.0$)	91.07	92.22	**91.64**
+ Explore ($\alpha=1.5$)	91.07	92.12	91.59

Table 3: Comparison of performance on PTB development set using different oracle training approaches.

6.2 Penn Treebank

Following the literature, we used the Wall Street Journal portion of the Penn Treebank, with standard splits for training (secs 2–21), development (sec 22), and test sets (sec 23). Because our parsing system seamlessly handles non-binary productions, minimal data preprocessing was required. For the part-of-speech tags which are a required input to our parser, we used the Stanford tagger with 10-way jackknifing.

Table 4 compares test our results on PTB to a range of other leading constituency parsers. Despite being a greedy parser, when trained using dynamic oracles with exploration, it achieves the best F_1 score of any closed-set single-model parser.

6.3 French Treebank

We also report results on the French treebank, with one small change to network structure. Specifically, we also included morphological features for each word as input to the recurrent network, using a small embedding for each such feature, to demonstrate that our parsing model is able to exploit such additional features.

We used the predicted morphological features, part-of-speech tags, and lemmas (used in place of word surface forms) supplied with the SPMRL 2014

Closed Training & Single Model	LR	LP	F_1
Sagae and Lavie (2006)	88.1	87.8	87.9
Petrov and Klein (2007)	90.1	90.3	90.2
Carreras et al. (2008)	90.7	91.4	91.1
Shindo et al. (2012)			91.1
†Socher et al. (2013)			90.4
Zhu et al. (2013)	90.2	90.7	90.4
Mi and Huang (2015)	90.7	90.9	90.8
†Watanabe and Sumita (2015)			90.7
Thang et al. (2015) (A*)	90.9	91.2	91.1
†*Dyer et al. (2016) (discrim.)			89.8
†*Cross and Huang (2016)			90.0
†***static oracle**	90.7	91.4	91.0
†***dynamic/exploration**	90.5	92.1	**91.3**
External/Reranking/Combo			
†Henderson (2004) (rerank)	89.8	90.4	90.1
McClosky et al. (2006)	92.2	92.6	92.4
Zhu et al. (2013) (semi)	91.1	91.5	91.3
Huang (2008) (forest)			91.7
†Vinyals et al. (2015) (WSJ)‡			90.5
†Vinyals et al. (2015) (semi)			92.8
†Durrett and Klein (2015)‡			91.1
†Dyer et al. (2016) (gen. rerank.)			92.4

Table 4: Comparison of performance of different parsers on PTB test set. †Neural. *Greedy. ‡External embeddings.

Parser	LR	LP	F_1
Björkelund et al. (2014)*,‡			82.53
Durrett and Klein (2015)‡			81.25
Coavoux and Crabbé (2016)			80.56
static oracle	83.50	82.87	83.18
dynamic/exploration	81.90	84.77	**83.31**

Table 5: Results on French Treebank. *reranking, ‡external.

data set (Seddah et al., 2014). It is thus possible that results could be improved further using an integrated or more accurate predictor for those features. Our parsing and evaluation also includes predicting POS tags for multi-word expressions as is the standard practice for the French treebank, though our results are similar whether or not this aspect is included.

We compare our parser with other recent work in Table 5. We achieve state-of-the-art results even in comparison to Björkelund et al. (2014), which utilized both external data and reranking in achieving the best results in the SPMRL 2014 shared task.

6.4 Notes on Experiments

For these experiments, we performed very little hyperparameter tuning, due to time and resource constraints. We have every reason to believe that performance could be improved still further with such techniques as random restarts, larger hidden layers, external embeddings, and hyperparameter grid search, as demonstrated by Weiss et al. (2015).

We also note that while our parser is very accurate even with greedy decoding, the model is easily adaptable for beam search, particularly since the parsing system already uses a fixed number of actions. Beam search could also be made considerably more efficient by caching post-hidden-layer feature components for sentence spans, essentially using the precomputation trick described by Chen and Manning (2014), but on a per-sentence basis.

7 Conclusion and Future Work

We have developed a new transition-based constituency parser which is built around sentence spans. It uses a factored system alternating between structural and label actions. We also describe a fast dynamic oracle for this parser which can determine the optimal set of actions with respect to a gold training tree in an arbitrary state. Using an LSTM model and only a few sentence spans as features, we achieve state-of-the-art accuracy on the Penn Treebank for all parsers without reranking, despite using strictly greedy inference.

In the future, we hope to achieve still better results using beam search, which is relatively straightforward given that the parsing system already uses a fixed number of actions. Dynamic programming (Huang and Sagae, 2010) could be especially powerful in this context given the very simple feature representation used by our parser, as noted also by Kiperwasser and Goldberg (2016b).

Acknowledgments

We thank the three anonymous reviewers for comments, Kai Zhao, Lemao Liu, Yoav Goldberg, and Slav Petrov for suggestions, Juneki Hong for proofreading, and Maximin Coavoux for sharing their manuscript. This project was supported in part by NSF IIS-1656051, DARPA FA8750-13-2-0041 (DEFT), and a Google Faculty Research Award.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *Proceedings of ACL*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-lstm parser. *arXiv preprint arXiv:1603.03793*.

Anders Björkelund, Ozlem Cetinoglu, Agnieszka Falenska, Richárd Farkas, Thomas Mueller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morpho-syntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 97–102.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Maximin Coavoux and Benoît Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. *Proceedings of ACL*.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. *Proceedings of ACL*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. *Proceedings of ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *Empirical Methods in Natural Language Processing (EMNLP)*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *Proceedings of HLT-NAACL*.

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

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Trans. of ACL*.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of ACL*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree lstms. *arXiv preprint arXiv:1603.00375*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 691–698. Association for Computational Linguistics.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 440–448. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

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

Le Quang Thang, Hiroshi Noji, and Yusuke Miyao. 2015. Optimal shift-reduce constituent parsing with structured perceptron.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. *Proceedings of ACL-IJCNLP*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*, pages 434–443.

Rule Extraction for Tree-to-Tree Transducers by Cost Minimization

Pascual Martínez-Gómez[1]
pascual.mg@aist.go.jp

Yusuke Miyao[1,2,3]
yusuke@nii.ac.jp

[1]Artificial Intelligence Research Center, AIST
[2]National Institute of Informatics and JST, PRESTO
[3]The Graduate University for Advanced Studies (SOKENDAI)
Tokyo, Japan

Abstract

Tree transducers that model expressive linguistic phenomena often require word-alignments and a heuristic rule extractor to induce their grammars. However, when the corpus of tree/string pairs is small compared to the size of the vocabulary or the complexity of the grammar, word-alignments are unreliable. We propose a general rule extraction algorithm that uses cost functions over tree fragments, and formulate the extraction as a cost minimization problem. As a by-product, we are able to introduce back-off states at which some cost functions generate right-hand-sides of previously unseen left-hand-sides, thus creating transducer rules "on-the-fly". We test the generalization power of our induced tree transducers on a QA task over a large Knowledge Base, obtaining a reasonable syntactic accuracy and effectively overcoming the typical lack of rule coverage.

1 Introduction

Tree transducers are general and solid theoretical models that have been applied to a variety of NLP tasks, such as machine translation (Knight and Graehl, 2005), text summarization (Cohn and Lapata, 2009), question answering (Jones et al., 2012), paraphrasing and textual entailment (Wu, 2005). One strategy to obtain transducer rules is by exhaustive enumeration; however, this method is ineffective when there is a high structural language variability and we wish to have an expressive model. Another strategy is to heuristically extract rules from a corpus of tree/string pairs and word-alignments, as GHKM algorithm does (Galley et al., 2004); however, word-alignments are difficult to estimate when the corpus is small. This would be the case, for example, of machine translation for low-resourced languages where there is often small numbers of parallel sentences, or in Question Answering (QA) tasks where the number of Knowledge Base (KB) identifiers (concepts) is much larger than QA datasets.

Our main contribution is an algorithm that formulates the rule extraction as a cost minimization problem, where the search for the best rules is guided by an ensemble of cost functions over pairs of tree fragments. In GHKM, a tree fragment and a sequence of words are extracted together if they are minimal and their word alignments do not fall outside of their respective boundaries. However, given that alignment violations are not allowed, the quality of the extracted rules degrades as the rate of misaligned words increases. In our framework, we can mimic GHKM by assigning an infinite cost to pairs of tree fragments that violate such conditions on word alignments and by adding a cost regularizer on the size of the tree fragments. Smoother cost functions, however, would permit controlled misalignments, contributing to generalization. Given the generality of these cost functions, we believe that the applicability of tree transducers will be extended.

A by-product of introducing these cost functions is that some of them may act as rule back-offs, where transducer rules are built "on-the-fly" when the transducer is at a predefined back-off state but there is no rule whose left-hand-side (*lhs*) matches the input subtree. These back-off states can be seen as functions that are capable of generating right-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 12–22,
Austin, Texas, November 1-5, 2016.

hand-sides (*rhs*) for unseen input subtrees.

Our rule extraction algorithm and back-off scheme are general, in the sense that they can be applied to any tree transformation task. However, in this paper, we extrinsically evaluate the quality of the extracted rules in a QA task, where the objective is to transform syntactic trees of questions into constituent trees that represent Sparql queries on Freebase, a large Knowledge Base. Implementing all components of a QA system at a sufficient level is out of the scope of this paper; for that reason, in order to evaluate our contribution in isolation, we use the FREE917 corpus released by Cai and Yates (2013), for which an entity and predicate lexicon is available[1]. We show that a tree-to-tree transducer induced using our rule extraction and back-off scheme is accurate and generalizes well, which was not previously achieved with tree transducers in semantic parsing tasks such as QA over large KBs.

2 Related Work

Tree transducers were first proposed by Rounds (1970) and Thatcher (1970), and have been greatly developed recently (Knight and Graehl, 2005). Jones et al. (2012) used tree transducers to semantically parse narrow-domain questions into Prolog queries for GeoQuery (Wong and Mooney, 2006), a small database of 700 geographical facts. Rules were exhaustively enumerated, which was possible given the small size of the database and low variability of questions. Another strategy is that of Li et al. (2013), where they used a variant of GHKM to induce tree transducers that parse into λ-SCFG. Word-to-node alignments could be reliably estimated with the IBM models (Brown et al., 1993) given, again, the small vocabulary and database size of GeoQuery. In such small-scale tasks, our rule extraction and back-off scheme offers no obvious advantage. However, when doing QA over larger and more realistic KBs (and other tasks with similar characteristics), exhaustive enumeration of rules or reliable estimations of alignments are not possible, which prevents the application of tree transducers. Thus, it is on the latter type of tasks where we focus our contribution.

A similar problem has been considered in the tree mapping literature in the form of the *tree-to-tree edit distance*. In that formulation, three edit operations are defined, namely, deleting and inserting single nodes, and replacing the label of a node. These edit operations have a cost associated to them, and the task consists of finding the minimum edit cost and its corresponding edit script[2] that transforms a source into a target tree. The problem was first solved by Tai (1979), and later Zhang and Shasha (1989) proposed a simpler and faster dynamic programming algorithm that operates in polynomial time, and that has inspired multiple variations (Bille, 2005).

However, we need edit operations that involve tree fragments (e.g., noun phrases or parts of verb phrases), rather than single nodes, when searching for the best mappings. We address this problem by searching for non-isomorphic tree mappings, in line with Eisner (2003), except that our rule extraction algorithm is guided by an ensemble of cost functions over *pairs of tree fragments*. This algorithm is capable of extracting rules more robustly than GHKM by permitting misalignments in a controlled manner. Finding a tree mapping solves simultaneously the alignment and the rule extraction problem.

There is a wide array of tree transducers with different expressive capabilities (Knight and Graehl, 2005). We consider *extended*[3] *root-to-frontier*[4] *linear*[5] transducers (Maletti et al., 2009), possibly with *deleting*[6] operations. In this paper, we syntactically parse the natural language question and transform it into a meaning representation, similarly to Ge and Mooney (2005). But instead of using Prolog formulae or λ-SCFG, we use constituent representations of $\lambda-$DCS expressions (Liang, 2013), which is a formal language convenient to represent Sparql queries where variables are eliminated by making existential quantifications implicit (see example in Figure 1).

Another challenge is to construct transducers with sufficient rule coverage, which would require billions of lexical rules that map question phrases to database entities or relations. Even if those rules were available, estimating their rule probabilities would be difficult given the small data sets of ques-

[1] The entity lexicon was released by the authors of FREE917, and the predicate lexicon is ours.

[2] Sequence of edit operations.

[3] *lhs* may have depth larger than 1.

[4] Top-down transformations.

[5] *lhs* variables appear at most once in the *rhs*.

[6] Some variables on the *lhs* may not appear in the *rhs*.

tions paired with their logical representations. We solve the problem by constructing lexical rules "on-the-fly" at the decoding stage, similarly to the candidate generation stage of entity linking systems (Ling et al., 2015). Rule weights are also predicted on-the-fly given rule features and model parameters similar to Cohn and Lapata (2009).

3 Background

Tree transducers apply to general tree transformation problems, but for illustrative purposes, we use the tree pair s and t in Figure 1 (from FREE917) as a running example. s is the syntactic constituent tree of the question "how many teams participate in the uefa", whereas t is a constituent tree of an executable meaning representation in the $\lambda-$DCS formalism:

$$\texttt{count}(\texttt{Team.League.Uefa})$$

Its corresponding lambda expression is

$$\texttt{count}(\lambda x.\exists a.\texttt{Team}(x,a) \wedge \texttt{League}(a,\texttt{Uefa}))$$

which can be converted into a Sparql KB query:

$$\begin{array}{l}\texttt{SELECT COUNT}(?x)\ \texttt{WHERE}\ \{\\ \quad ?a\quad \texttt{Team}\quad\quad ?x\quad\quad .\\ \quad ?a\quad \texttt{League}\quad \texttt{Uefa}\quad .\ \}\end{array}$$

Following the terminology of Graehl and Knight (2004), we define a tree-to-tree transducer as a 5-tuple $(\mathcal{Q}, \Sigma, \Delta, q_{\text{start}}, \mathcal{R})$ where $\mathcal{Q}$ is the set of states, Σ and Δ are the sets of symbols of the input and output languages, q_{start} is the initial state, and $\mathcal{R}$ is the set of rules. For convenience, define T_Σ as the set of trees with symbols in Σ, $T_\Sigma(\mathcal{A})$ the set of trees with symbols in $\Sigma \cup \mathcal{A}$ where symbols in $\mathcal{A}$ only appear in the leaves, $\mathcal{X}$ as the set of variables $\{x_1, \ldots, x_n\}$, and $\mathcal{A}.\mathcal{B}$ for the cross-product of two sets $\mathcal{A}$ and $\mathcal{B}$. A rule $r \in \mathcal{R}$ has the form $q.t_i \xrightarrow{s} t_o$, where $q \in \mathcal{Q}$ is a state, $t_i \in T_\Sigma(\mathcal{X})$ is the left-hand-side (*lhs*) tree pattern (or elementary tree), $t_o \in T_\Delta(\mathcal{Q}.\mathcal{X})$ the right-hand-side (*rhs*), and $s \in \mathbb{R}$ the rule score.

Tree-to-tree transducers apply a sequence of rules to transform a source s into a target t tree. A root-to-frontier transducer starts at the root of the source tree and searches $\mathcal{R}$ for a rule whose i) tree pattern t_i on the *lhs* matches the root of the source tree, and ii) the state q of the rule is the initial state of the transducer. An incipient target tree is created by copying the *rhs* of the rule. Then, the transducer recursively and independently visits the subtrees of the source tree at the *lhs* variable positions of the rule from their new states, and copies the results into the same variable on the target tree.

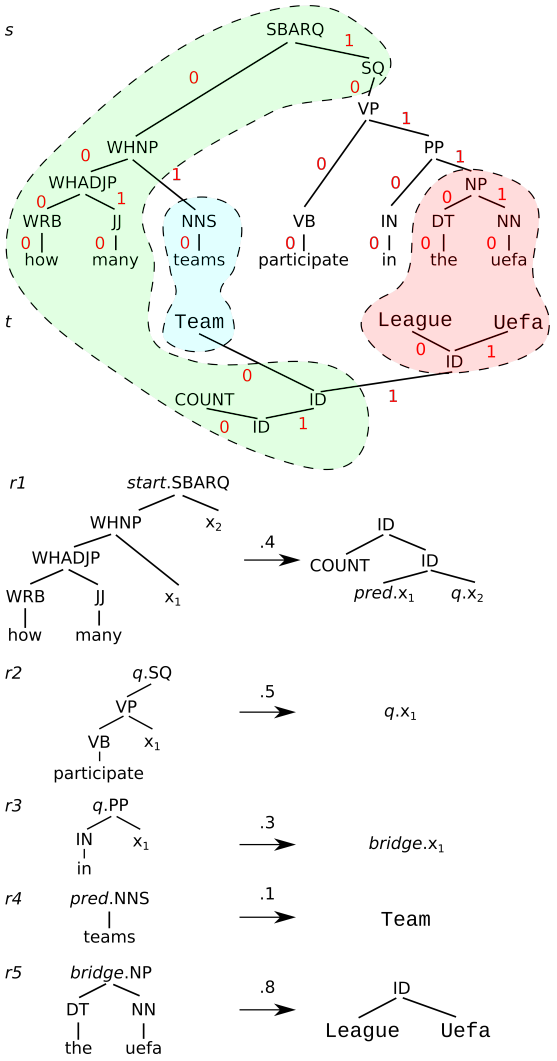


Figure 1: (s) Constituent tree of a question; (t) executable meaning representation; $r1$ - $r5$ are typical transducer rules extracted by our algorithm, where q is a generic state, *pred* and *bridge* are predicate and bridged entity back-off states.

In Figure 1, the sequential application of rules $r1$ to $r5$ is a derivation that transforms the question s into the query t. For example, rule $r1$ consumes a tree fragment of s (e.g. "how", "many", "WRB", etc.) and produces a tree fragment with terminals ("COUNT", x_1, x_2) and non-terminals ("ID") with

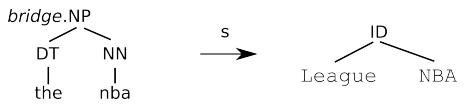
a specific structure. Rules $r2$ and $r3$ only consume but do not produce symbols (other than variables). The *rhs* of rules are target tree fragments that connect to each other at the frontier nodes (those with variables). Rules $r4$ and $r5$ are terminal rules, where $r4$ produces the predicate `Team` and rule $r5$ produces the entity `Uefa` and a disambiguating predicate `League` that has no lexical support on the source side, similarly to the role that *bridging predicates* play in Berant et al. (2013).

Given a corpus of source and target tree pairs, the learning stage aims to obtain rules such as $r1 - r5$ in Figure 1 and their associated probabilities or scores. We discuss our novel approach to rule extraction in Section 5. For the assignment of rule scores, we adopt the latent variable averaged structured perceptron, a discriminative procedure similar to Tsochantaridis et al. (2005) and Cohn and Lapata (2009). Here, we instantiate feature values $\mathbf{f}$ for every rule, and reward the weights $\mathbf{w}$ of rules that participate in a derivation (latent variable) that transforms a training source tree into a meaning representation that retrieves the correct answer.

At decoding stage, rule scores can be predicted as $s = \mathbf{w} \cdot \mathbf{f}$. However, we cannot expect to have extracted all necessary rules at the training stage given the small training data and large-scale KB. For that reason, we propose in Section 4 a novel rule back-off scheme to alleviate coverage problems.

4 Back-off rules

As an illustrative example, consider the question "how many teams participate in the nba", and the rules $r1$ to $r5$ in Figure 1. When the transducer attempts to transform the noun phrase (NP (DT the) (NN nba)), no rule's *lhs* matches it. However, since the transducer is at state *bridge* (as specified by the *rhs* of $r3$), it should be able to produce a list of bridged entities, among which the target subtree (ID `League` `NBA`) will be hopefully included. Thus, the following rule should be created for the occasion:



This mechanism produces rules "on-the-fly", allowing us to compensate low rule coverage by consuming and producing tree fragments that were not necessarily observed in the training data.

Back-off rules are produced when the transducer is at a back-off state $q_b \in \mathcal{Q}_b \subset \mathcal{Q}$, similarly as the back-off mechanisms in finite-state language models where we produce estimates (probabilities) of input structures (sequences) under less conditioning. In our scheme, a back-off state (or function) q_b produces estimates that are target structures $t_2 \in T_\Delta$ with score $s \in \mathbb{R}$, given some information of the source tree fragment $t_1 \in T_\Sigma$. That is, a function $q_b : T_\Sigma \rightarrow \{(T_\Delta, \mathbb{R}), \ldots\}$. In our QA application, we only use the leaves of the input subtree t_1 and use lexicons or entity/predicate linkers to retrieve KB entities, KB relations or a compound of a disambiguating relation and an entity from back-off states *ent*, *pred* and *bridge*, respectively. Other back-off functions would transliterate the leaves of the input tree in machine translation, or produce synonyms/hypernyms in a paraphrasing application.

We associate a score s to these newly created rules, which we learn to predict using the discriminative training procedure suggested by Tsochantaridis et al. (2005), as described in Section 3.

Back-off rules are then constructed on-demand as $q_b.t_1 \xrightarrow{s} t_2$, and the discrete set of rules $\mathcal{R}$ is augmented with them. It remains now to recognize those back-off states when inducing tree transducer grammars, which is covered in Section 5.1.

5 Rule Extraction

Given a pair of trees, our rule extraction algorithm finds a tree mapping that implicitly describes the rules that transform a source into a target tree. In the search of the best mapping, we need to explore the space of edit operations, which are substitutions of source by target tree fragments. We define cost functions for these edit operations, and formulate the tree mapping as a cost minimization problem. Whereas our tree mapping algorithm and back-off scheme are generic and can be used in any tree transformation task, cost functions depend on the application.

5.1 Cost functions

In general, cost functions are defined over edit operations, which are pairs of source and target tree fragments, $\text{cost} : T_\Sigma(\mathcal{X}) \times T_\Delta(\mathcal{Q}.\mathcal{X}) \rightarrow \mathbb{R}^{\geq 0}$, and they are equivalent to feature functions. Some cost

functions are defined over all pairs of tree fragments.

For this QA application, these are:

$$\textit{csize}(t_1, t_2) = |\text{nodes}(t_1)|^2 + |\text{nodes}(t_2)|^2$$

which acts as a tree size regularizer, returning a cost quadratic to the size of the tree fragments, thus encouraging small rules. The cost function *ccount* assigns zero cost if (i) "how" and "many" appear in t_1, and (ii) "`COUNT`" appears in t_2. If only either (i) or (ii), the cost is a positive constant. Similarly, other operators (max, min, argmax, etc.) could be recognized, but this dataset did not require them.

Other cost functions only apply to some pairs of tree fragments. These are the back-off functions described in Section 4, but instead of returning scores for every target tree fragment, they return a cost, e.g. $\textit{cent} : T_\Sigma \times T_\Delta \rightarrow \mathbb{R}^{\geq 0}$. An ensemble will produce up to three different costs for every pair of tree fragments, depending on what back-off functions were triggered. In the case of the entity cost function:

$$\begin{aligned}\gamma_{\textit{ent}}(t_1, t_2) = \lambda_1 \cdot \textit{csize}(t_1, t_2) \\ + \lambda_2 \cdot \textit{ccount}(t_1, t_2) \\ + \lambda_3 \cdot \textit{cent}(t_1, t_2)\end{aligned} \quad (1)$$

where $\lambda_i \in \mathbb{R}^{\geq 0}$ are scaling factors. In the search of the lowest-cost mapping, the labels of the cost functions that are derived from the back-off functions (e.g. $\gamma_{\textit{ent}}$, $\gamma_{\textit{pred}}$) are memorized for the pairs (t_1, t_2) for which they were defined and for which they outputted a cost. These labels are then used as back-off rule state names when constructing rules.

5.2 Tree Mapping: Optimization Problem

Intuitively, the cost of mapping a source node n_s to a target node n_t is equal to the cost of transforming a tree fragment $T_\Sigma(\mathcal{X})$ rooted at node n_s into a tree fragment $T_\Delta(\mathcal{Q}.\mathcal{X})$ rooted at node n_t, plus the sum of costs of mapping the frontier nodes rooted at the variables. In order to formalize our tree mapping, we need a more precise definition of a tree fragment where the locations of variables $\mathcal{X}$ are specified by paths. The notation to specify subtrees is taken from (Graehl and Knight, 2004), and we introduce the $\perp$ operator for convenience.

A path p is a tuple, equivalent to a Gorn address, that uniquely identifies the node of a tree by specifying the sequence of child indices to the node from the root. In the tree s of Figure 1, the path to the VP node is $(1, 0)$, whereas in t, the path to `League` is $(1, 1, 0)$. The path $p = ()$ refers to the root of a tree. We denote by $s \downarrow p$ the subtree of tree s that is rooted at path p and that has no variables. In Figure 1, the left-hand-side (*lhs*) of $r5$ is the subtree $s \downarrow (1, 0, 1, 1)$. In order to introduce variables, we generalize the notion of subtree into a tree pattern $s \downarrow p \perp \{p_1, \ldots, p_n\}$, where n variables replace subtrees $s \downarrow p_i$ at subpaths $p_i \in \{p_1, \ldots, p_n\}$. For example, the *lhs* of $r1$ can be represented with the tree pattern $s \downarrow () \perp \{(0, 1), (1)\}$, and $r2$ with $s \downarrow (1) \perp \{(1, 0, 1)\}$. Note that the order of subpaths $\{p_1, \ldots, p_n\}$ matters. A tree pattern with no subpaths $s \downarrow p \perp \{\}$ is simply a subtree $s \downarrow p$, such as the *lhs* of rules $r4$ and $r5$; a tree pattern with only one subpath equal to its path $s \downarrow p \perp \{p\}$ is a single variable, such as the *rhs* of rules $r2$ and $r3$. Note that in $s \downarrow p \perp \{p_1, \ldots, p_n\}$, all paths p_i to variables are prefixed by p, and that no variables are descendants of any other variable in the same tree pattern. In other words, $\mathbf{p} = \{p_1, \ldots, p_n\}$ are *disjoint subpaths* given p, where $\mathbf{p}$ denotes a list of paths.

We can now formalize the tree mapping algorithm as an optimization problem. Let $\gamma\left(s \downarrow p_s \perp \mathbf{p}, t \downarrow p_t \perp \mathbf{p}'\right)$ be the cost to transform a source into a target tree pattern, as defined in Equation 1. To transform $s \downarrow p_s$ into $t \downarrow p_t$, we need to find the best combination of source subtrees rooted at $\{p_1, \ldots, p_n\}$ that can be transformed at minimum cost to the best combination of target subtrees at $\{p'_1, \ldots, p'_n\}$. The transformation cost of a certain tree pattern $s \downarrow p_s \perp \{p_1, \ldots, p_n\}$ into $t \downarrow p_t \perp \{p'_1, \ldots, p'_n\}$ is equal to the cost of transforming the source tree pattern into the target tree pattern, plus the minimum cost to transform $s \downarrow p_i$ into $t \downarrow p'_i$, for $i \geq 1$. That is:

$$\begin{aligned}&\mathrm{C}\left(s \downarrow p_s, t \downarrow p_t\right) = \\ &\min_{\mathbf{p},\mathbf{p}'}\{\gamma\left(s \downarrow p_s \perp \mathbf{p}, t \downarrow p_t \perp \mathbf{p}'\right) + \\ &\qquad \sum_{i=1}^{|\mathbf{p}|} \mathrm{C}\left(s \downarrow p_i, t \downarrow p'_i\right)\}\end{aligned} \quad (2)$$

subject to $|\mathbf{p}| = |\mathbf{p}'|$, that is, source and target tree patterns having the same number of variables. Then, the cost of transforming the source into the target tree would be given by the expression

$\mathrm{C}\,(s \downarrow (), t \downarrow ())$. Since we are only interested in the pairs of source and target tree patterns that lead to the minimum cost, we keep track of subpaths $\mathbf{p}$ and $\mathbf{p}'$ of tree pattern pairs that minimize the cost.

5.3 Algorithm

5.3.1 Overview

This problem can be solved for small depths of tree patterns and a small number of variables by storing intermediate results in the computation of Eq. 2. However, an exact implementation needs to enumerate all pairs of source and target disjoint subpaths ($\mathbf{p}$ and $\mathbf{p}'$), which has a computational complexity that grows combinatorially with $|\mathbf{p}|$ (variable permutations), and exponentially with the number of descendant nodes of p_s and p_t (powerset of variables).

Instead, we use a beam search algorithm (see Algorithm 1)[7] that constructs source and target disjoint paths ($\mathbf{p}$ and $\mathbf{p}'$) hierarchically (function GENERATEDISJOINT) in a bottom-up order, for any given path pair (p_s, p_t). First, n-best solutions (pairs of disjoint paths) are computed for children; then those partial solutions are combined into their parent using the cross-product. Solutions (with their associated cost) for every pair of paths (p_s, p_t) are stored in a weighted hypergraph, from which we can extract n-best derivations (sequences of rules). In the pseudocode, we use a helper function, $\mathrm{paths}(s \downarrow p_s)$, which denotes the list of subtree paths in bottom-up order: from the leaves up to p_s (including the latter).

5.3.2 Detailed Description

For a certain path pair (p_s, p_t), there are three cases. The first case (line 34-35) considers a pair of empty disjoint subpaths $(\mathbf{p}, \mathbf{p}') = (\{\}, \{\})$, where the cost c of transforming $s \downarrow p_s \perp \{\}$ into $t \downarrow p_t \perp \{\}$ is evaluated and the empty disjoint subpaths are added to the priority queue P, indexed with p_s. Such indexing is useful to retrieve the n-best pairs of disjoint subpaths accumulated at every tree node.

The second case (line 28 to 31) evaluates the cost of transforming single-variable tree patterns: $s \downarrow p_s \perp \{p_c\}$ into $t \downarrow p_t \perp \{p'_c\}$. In this case, variables substitute entire subtrees rooted at paths p_c and p'_c on the source and target tree patterns, respectively. Note that p_c ranges over all node addresses that are descendant of p_s (including p_s), and similarly for p'_c. The pairs of disjoint subpaths $(\mathbf{p}, \mathbf{p}') = (\{p_c\}, \{p'_c\})$ are added into the priority queue, indexed by p_c.

The third case (line 16 to 26) performs the combination of subpaths hierarchically from children to parents, and incrementally across children. For every path $p_c \in \mathrm{paths}(s \downarrow p_s)$, it visits its imme-

[7] https://github.com/pasmargo/t2t-qa

Algorithm 1 Extraction of optimal sequence of rules to transform a source s into a target tree t.

Input: Trees s and t, and ensemble of cost functions γ.
Output: Sequence of optimal rules for $s \Rightarrow^* t$.

```
 1: let H = (V, E) be a hypergraph of solutions with
    V ← {} vertices and E ← {} hyperedges.
 2: for (p_s, p_t) ∈ paths(s) × paths(t) do
 3:   add vertex v = (p_s, p_t) to V
 4:   PP ← GENERATEDISJOINT(s ↓ p_s, t ↓ p_t, γ)
 5:   for (p, p') ∈ PP do
 6:     ▷ Get cost of tree pattern pair.
 7:     c ← γ(s ↓ p_s ⊥ p, t ↓ p_t ⊥ p')
 8:     add edge (p_s, p_t) -c-> (p, p') to E
 9:   end for
10: end for
11: return HYPERGRAPHSEARCH(H)

12: function GENERATEDISJOINT(s ↓ p_s, t ↓ p_t, γ)
13:   P ← {} a priority queue of partial disjoint paths.
14:   for every p_c ∈ paths(s ↓ p_s) do
15:     ▷ Costs when variables combined from children.
16:     for every p_ic immediate child of p_c (if any) do
17:       ▷ Retrieve n-best subpaths p and p' from p_ic.
18:       C ← arg min^n_(p,p'){c | (p_ic, p, p', c) ∈ P}
19:       ▷ Combine subpaths with those accumulated
20:       ▷ from previous siblings and stored at path p_c.
21:       A ← arg min^n_(p,p'){c | (p_c, p, p', c) ∈ P}
22:       for (p, p') ∈ (C ∪ (C.A)) do
23:         c ← γ(s ↓ p_s ⊥ p, t ↓ p_t ⊥ p')
24:         add (p_c, p, p', c) to priority queue P
25:       end for
26:     end for
27:     ▷ Cost of tree patterns with one variable.
28:     for every p'_c ∈ paths(t ↓ p_t) do
29:       c ← γ(s ↓ p_s ⊥ {p_c}, t ↓ p_t ⊥ {p'_c})
30:       add (p_c, {p_c}, {p'_c}, c) to priority queue P
31:     end for
32:   end for
33:   ▷ Cost of tree patterns with no variables.
34:   c ← γ(s ↓ p_s ⊥ {}, t ↓ p_t ⊥ {})
35:   add (p_s, {}, {}, c) to priority queue P
36:   return arg min^n_(p,p'){c | (p_s, p, p', c) ∈ P}
37: end function
```

diate children p_{ic} one by one, and retrieves into C the n-best disjoint subpaths (line 18) that have already been obtained during previous iterations for p_{ic}. Then, we retrieve into A the n-best disjoint subpaths indexed at p_c, which is a list of the best subpaths that were combined from previous immediate children (the list is empty if this is the first immediate child that we visit). The cross-product of disjoint subpaths in C and A, that is $C.A$, is then evaluated and the best combinations are stored in the priority queue indexed at p_c.

As an example of a cross-product between two lists C and A of pairs of disjoint paths, let $C = \{(\mathbf{p_1}, \mathbf{p_1}'), (\mathbf{p_2}, \mathbf{p_2}')\}$ and $A = \{(\mathbf{p_3}, \mathbf{p_3}')\}$. Then the cross-product $C.A$ would be:

$$C.A = \{(\mathbf{p}_1 \cdot \mathbf{p}_3, \mathbf{p}_1' \cdot \mathbf{p}_3'), (\mathbf{p}_2 \cdot \mathbf{p}_3, \mathbf{p}_2' \cdot \mathbf{p}_3')\}$$

where $\mathbf{p}_1 \cdot \mathbf{p}_3 = \{(0,1),(0,2),(0,3),(0,4)\}$ if $\mathbf{p}_1 = \{(0,1),(0,2)\}$ and $\mathbf{p}_3 = \{(0,3),(0,4)\}$. At this stage, subpaths $\mathbf{p}_i$ or $\mathbf{p}_i'$ that are not disjoint are discarded, together with disjoint paths that produce tree patterns with depth larger than a certain user-defined threshold, or whose number of subpaths is larger than the number of variables allowed.

In line 24, the disjoint subpaths of C (in addition to their cross-product $C.A$) are also evaluated and added to the priority queue indexed by p_c, to propagate upwards in the hierarchy of solutions the decision of not combining disjoint subpaths.

Finally, GENERATEDISJOINT returns the n-best pairs of disjoint subpaths of minimum cost ($\mathbf{p}$, $\mathbf{p}'$) that accumulated in the priority queue P for path p_s.

5.3.3 Other Considerations

The n-best source and target pairs of disjoint subpaths are stored at every pair of source and target paths (p_s, p_t) (lines 2-10), forming a hypergraph, as in Figure 2. Then, with a hypergraph search (Huang and Chiang, 2007) we can retrieve at least n-best sequences of rules (derivations) that transform the source into the target tree (line 11).

To maintain diversity of partial disjoint subpaths, we divide P into a matrix of buckets with as many rows and columns as the number of non-variable terminals of the source and target tree patterns, trading memory for more effective search (Koehn, 2015). This operation is implicit in lines 24, 30 and 35.

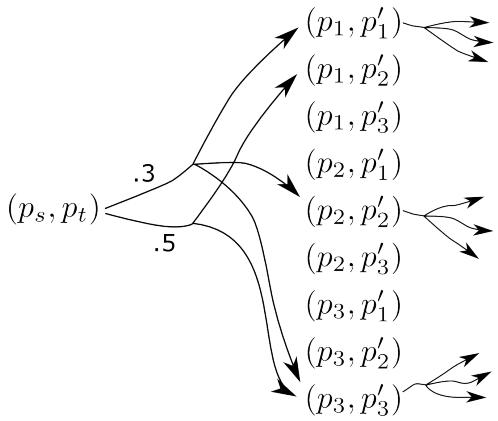


Figure 2: Hypergraph with 2-best pairs of disjoint subpaths for (p_s, p_t). Vertices are pairs of source and target paths. Hyperedges are pairs of tree patterns. The hyperedge with cost .3 denotes the pair $s \downarrow p_s \perp \{p_1, p_2, p_3\}$ and $t \downarrow p_t \perp \{p_1', p_2', p_3'\}$. The one with cost .5, $s \downarrow p_s \perp \{p_1, p_3\}$ and $t \downarrow p_t \perp \{p_2', p_3'\}$.

6 Experiments

6.1 Experiment Settings

Data The training data is a corpus of questions annotated with their logical forms that can be executed on Freebase to obtain a precise answer. For an unseen set of questions, the task is to obtain automatically their logical forms and retrieve the correct answer. Our objective is to evaluate the generalization capabilities of a transducer induced using our rule extraction on an unseen open-domain test set. We parsed questions from FREE917 into source constituent trees using the Stanford caseless model (Klein and Manning, 2003). Target constituent (meaning) representations were obtained by a simple heuristic conversion from the $\lambda-$DCS expressions released by Berant et al. (2013). We evaluate on the same training and testing split as in Berant et al. (2013). Tree pairs (2.9%) for which the gold executable meaning representation did not retrieve valid results were filtered out.

Baselines We compared to two baselines. The first one is SEMPRE (Berant et al., 2013), a state-of-the-art semantic parser that uses a target language grammar to over-generate trees, and a log-linear model to estimate the parameters that guide the decoder towards trees that generate correct answers. For FREE917, SEMPRE uses a manually-created entity lexicon released by (Cai and Yates, 2013), but an automatically generated predicate lexicon. In-

stead, our system and the second baseline use manually created entity *and* predicate lexicons, where the latter was created by selecting all words from every question that relate to the target predicate. For example, for the question "what olympics has egypt participated in", we created an entry that maps the discontinuous phrase "olympics participated in" to the predicate `OlympicsParticipatedIn`.

The second baseline is a tree-to-tree transducer whose rules are extracted using a straightforward adaptation of the GHKM algorithm (Galley et al., 2004) for pairs of trees. Word-to-concept alignments are extracted using three different strategies: i) **ghkm-g** uses the IBM models (Brown et al., 1993) as implemented in GIZA++ (Och and Ney, 2003), ii) **ghkm-m** maps KB concepts (target leaves) to as many source words as present in the entity/predicate lexicons, and iii) **ghkm-c** maps KB concepts as in ii) but only retaining the longest contiguous sequence of source words (or right-most sequence if there is a tie). Bridging predicates are assumed when a KB concept does not align (according to the lexicon) to any source word. Finally, rule state names are set according to the mechanism described in Section 5.

Our *ent*, *pred* and *bridge* cost/back-off functions assign a low cost (or high score) to source and target tree patterns with no variables whose leaves appear in either the entity or the predicate lexicons. Scaling factors λ_i (see Eq. 1) were subjectively tuned on 20 training examples. When used as back-off functions, they generate as many *rhs* as entities or predicates can be retrieved from the lexicons by at least one of the words in the source tree pattern. Bridging predicates are dispreferred by adding an extra constant cost. At back-off, this score function generates a variable predicate, acting as a wildcard in Sparql.

Our system t2t For the rule extraction, we use a beam size of 10, and we output 100 derivations for every tree pair. We do not impose any limit in the depth of *lhs* or *rhs*, or in the number of variables. To increase the coverage of our rules, we produce deleting tree transducers by replacing fully lexicalized branches that are directly attached to the root of a *lhs* with a deleting variable.

For the parameter estimation, we used 3 iterations of the latent variable averaged structured perceptron, where the number of iterations was selected on 20% of held-out training data. To assess the equality between the gold and the decoded tree, we compare their denotations. The features for the discriminative training were the *lhs* and *rhs* roots, the number of variables, deleting variables and leaves, the presence of entities or predicates in the *rhs*, the rule state and children states, and several measures of character overlaps between the leaves of the source and information associated to leaves in target tree patterns.

For decoding, we used standard techniques (Graehl and Knight, 2004) to constrain and prune weighted regular tree grammars given a tree transducer and a source tree, and used the cube-growing algorithm to generate $10,000$ derivations, converted them to Sparql queries, and retained those that were valid (either syntactically correct or that retrieved any result). We compute the accuracy of the system as the percentage of questions for which the 1-best output tree retrieves the correct answer, and the coverage as the percentage for which the correct answer is within the $10,000$ best derivations. The average rule extraction time per tree pair when using beam size 1 was 0.46 seconds (median 0.35, maximum 2.94 seconds). When using beam size 10, the average was 4.7 seconds (median 2.02, maximum 104.4 seconds), which gives us a glimpse of how the beam size influences the computational complexity for the typical tree size of FREE917 questions.

6.2 Results

Results are in Table 1. Note that although we compare our results to those obtained with SEMPRE (Berant et al., 2013), the systems cannot really be compared since Berant et al. (2013) did not have access to a manually created lexicon of predicates. When comparing the average number of entity and predicate rules that the back-off functions generate, we see that the number of predicate rules is much larger, implying a higher ambiguity. Despite this, our base system still produces promising results in terms of accuracy and coverage.

We also carried out several ablation experiments to investigate what are the characteristics of the system that contribute the most to the accuracy: In **no nbest**, we only extract one sequence of rules that transform a source into a target tree. In **no del**, we do not introduce deleting variables. In **beam 1**, we use beam size 1 in rule extraction. In **no size**, no tree size regularizer cost function is used. And in

Systems	Acc.	Cov.	# Preds.	# Ents.	# Rules
SEMPRE	.62	–	–	–	–
ghkm-c	.49	.80	155	14	384
ghkm-m	.48	.77	147	14	399
ghkm-g	.08	.57	102	5	135
t2t	.64	.78	187	19	437
t2t-e	.69	.85	191	20	430
no del	.64	.78	187	19	437
no size	.59	.78	195	19	483
no nbest	.58	.70	93	5	128
beam 1	.53	.65	84	5	112
no back	.00	.01	0	0	175
train-600	.62	.78	187	19	429
train-500	.61	.77	184	19	413
train-400	.62	.75	178	19	390
train-300	.59	.75	177	18	363
train-200	.52	.74	169	17	317
train-100	.52	.67	138	14	317

Table 1: Accuracy and coverage results; average number of predicate rules, entity rules and all rules per input tree.

no back, no rule back-offs are used. As we see, removing the back-off rule capabilities is critical in this setting and makes the QA task unfeasible. We also studied the impact of the size of the training data in the generalization of our system, by training the system in $\{100, 200, \ldots, 600\}$ examples. We found that the accuracy saturates at only 400 training instances, which might be advantageous in tasks where training resources are scarce. Finally, in order to estimate the upper bound in the coverage and accuracy of our approach on FREE917, we also run our pipeline **t2t-e** with a refined version of Cai and Yates (2013)'s entity lexicon, where 65 missing entities are added (7.8% of the total). We can observe a significant increase in the accuracy and coverage of the system, suggesting that the bottleneck may lie in the entity/predicate linking procedures.

7 Future Work

One step further in the generalization of the rule extraction is to remove the necessity of explicitly providing cost functions such as word-to-word hard-alignments or costs between tree fragments. This would allow us to remove the bias introduced by engineered cost functions and to obtain rules that are globally optimal. In this setup, the parameters of the cost functions are to be learned with the objective to maximize the likelihood on the training data or a downstream application performance. However, since rules (or tree mappings) would become hidden variables, this generalized rule extraction may require faster methods to enumerate plausible rules. Another extension would be to make the rule extraction more robust against parsing errors, using pairs of forests instead of pairs of trees, similarly as in Liu et al. (2009).

Regarding the QA application, there are two natural extensions that we want to address, namely to develop general and automatic entity and predicate linking mechanisms for large knowledge bases, and to test our approach in datasets that require higher levels of compositionality such as the QALD challenges (Unger et al., 2015) or those datasets produced by Wang et al. (2015).

8 Conclusion

We proposed to induce tree to tree transducers using a rule extraction algorithm that uses cost functions over pairs of tree fragments (instead of word-alignments), which increases the applicability of these models. Some cost functions may act as rule back-offs, generating new *rhs* given unseen *lhs*, thus producing transducer rules "on-the-fly". The scores of these rules are obtained on demand using a discriminative training procedure that estimates weights for rule features. This strategy was useful to compensate the lack of rule coverage when inducing tree transducers from small tree corpora.

As a proof-of-concept, we tested the tree transducer induced with our algorithm on a QA task over a large KB, a domain in which tree transducers have not been effective before. In this task, lexicon mappings were naturally introduced as cost functions and rule back-offs, without loss of generality. Despite using a manually created lexicon of predicates, we showed a high accuracy and coverage of non-final rules, which are promising results.

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and is also supported by JSPS KAKENHI Grant Number 16K16111. We thank Yoshimasa Tsuruoka, Yo Ehara and the anonymous reviewers for their helpful comments.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1–3):217 – 239.

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

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria, August. Association for Computational Linguistics.

Trevor Anthony Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 205–208, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule. In *HLT-NAACL'2004: Main Proceedings*, pages 273–280.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 105–112, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 488–496, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 1–24. Springer Berlin Heidelberg.

Philipp Koehn. 2015. *Moses Manual*.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended GHKM algorithm for inducing Lambda-SCFG. pages 605–611.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408.

Xiao Ling, Sameer Singh, and Daniel Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566, Suntec, Singapore, August. Association for Computational Linguistics.

Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. 2009. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

William C. Rounds. 1970. Mappings and grammars on trees. *Mathematical systems theory*, 4(3):257–287.

Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, July.

James W. Thatcher. 1970. Generalized sequential machine maps. *Journal of Computer and System Sciences*, 4(4):339 – 367.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, volume 6, pages 1453–1484.

Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. 2015. Question Answering over Linked Data (QALD-5). In Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan, editors, *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum*, volume 1391. Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China, July. Association for Computational Linguistics.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 439–446, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05, pages 25–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

A Neural Network for Coordination Boundary Prediction

Jessica Ficler
Computer Science Department
Bar-Ilan University
Israel
jessica.ficler@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Israel
yoav.goldberg@gmail.com

Abstract

We propose a neural-network based model for coordination boundary prediction. The network is designed to incorporate two signals: the similarity between conjuncts and the observation that replacing the whole coordination phrase with a conjunct tends to produce a coherent sentences. The modeling makes use of several LSTM networks. The model is trained solely on conjunction annotations in a Treebank, without using external resources. We show improvements on predicting coordination boundaries on the PTB compared to two state-of-the-art parsers; as well as improvement over previous coordination boundary prediction systems on the Genia corpus.

1 Introduction

Coordination is a common syntactic phenomena, appearing in 38.8% of the sentences in the Penn Treebank (PTB) (Marcus et al., 1993), and in 60.71% of the sentences in the Genia Treebank (Ohta et al., 2002). However, predicting the correct conjuncts span remain one of the biggest challenges for state-of-the-art syntactic parsers. Both the Berkeley and Zpar phrase-structure parsers (Petrov et al., 2006; Zhang and Clark, 2011) achieve F1 scores of around 69% when evaluated on their ability to recover coordination boundaries on the PTB test set. For example, in:

"He has the government's blessing to [build churches] and [spread Unificationism] in that country."

the conjuncts are incorrectly predicted by both parsers:

Berkeley: *"He has the government's blessing to [build churches] and [spread Unificationism in that country]."*
Zpar: *"He [has the government's blessing to build churches] and [spread Unificationism in that country]."*

In this work we focus on coordination boundary prediction, and suggest a specialized model for this task. We treat it as a ranking task, and learn a scoring function over conjuncts candidates such that the correct candidate pair is scored above all other candidates. The scoring model is a neural network with two LSTM-based components, each modeling a different linguistic principle: (1) conjuncts tend to be similar ("symmetry"); and (2) replacing the coordination phrase with each of the conjuncts usually result in a coherent sentence ("replacement"). The *symmetry* component takes into account the conjuncts' syntactic structures, allowing to capture similarities that occur in different levels of the syntactic structure. The *replacement* component considers the coherence of the sequence that is produced when connecting the participant parts. Both of these signals are syntactic in nature, and are learned solely based on information in the Penn Treebank. Our model substantially outperforms both the Berkeley and Zpar parsers on the coordination prediction task, while using the exact same training corpus. Semantic signals (which are likely to be based on resources external to the treebank) are also relevant for coordination disambiguation (Kawahara and Kurohashi, 2008; Hogan, 2007) and provide complementary information. We plan to incorporate such signals in future work.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 23–32,
Austin, Texas, November 1-5, 2016.

2 Background

Coordination is a very common syntactic construction in which several sentential elements (called conjuncts) are linked. For example, in:

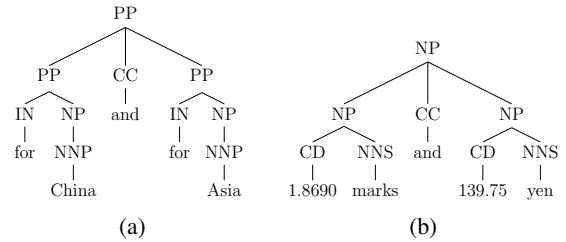
"The Jon Bon Jovi Soul Foundation [was founded in 2006] and$_1$ [exists to combat issues that force (families) and$_2$ (individuals) into economic despair]."

The coordinator *and*$_1$ links the conjuncts surrounded with square brackets and the coordinator *and*$_2$ links the conjuncts surrounded with round brackets.

Coordination between NPs and between VPs are the most common, but other grammatical functions can also be coordinated: *"[relatively active]$_{ADJP}$ but [unfocused]$_{ADJP}$"* ; *"[in]$_{IN}$ and [out]$_{IN}$ the market"*. While coordination mostly occurs between elements with the same syntactic category, cross-category conjunctions are also possible: (*"Alice will visit Earth [tomorrow]$_{NP}$ or [in the next decade]$_{PP}$"*). Less common coordinations involve non-constituent elements *"[equal to] or [higher than]"*, argument clusters (*"Alice visited [4 planets] [in 2014] and [3 more] [since then]"*), and gapping (*"[Bob lives on Earth] and [Alice on Saturn]"*) (Dowty, 1988).

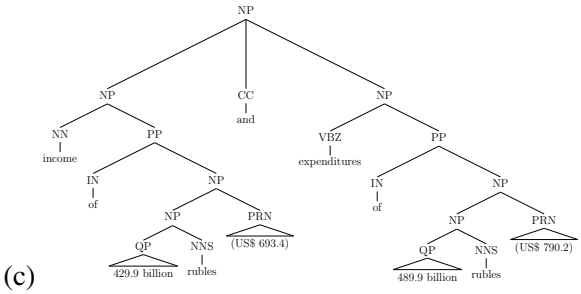
2.1 Symmetry between conjuncts

Coordinated conjuncts tend to be semantically related and have a similar syntactic structure. For example, in (a) and (b) the conjuncts include similar words (China/Asia, marks/yen) and have identical syntactic structures.



(a) (b)

Symmetry holds also in larger conjuncts, such as in:



(c)

Similarity between conjuncts was used as a guiding principle in previous work on coordination disambiguation (Hogan, 2007; Shimbo and Hara, 2007; Hara et al., 2009).

2.2 Replaceability

Replacing a conjunct with the whole coordination phrase usually produce a coherent sentence (Huddleston et al., 2002). For example, in *"Ethan has developed [new products] and [a new strategy]"*, replacement results in: *"Ethan has developed new products"*; and *"Ethan has developed a new strategy"*, both valid sentences. Conjuncts replacement holds also for conjuncts of different syntactic types, e.g.: *"inactivation of tumor-suppressor genes, [alone] or [in combination], appears crucial to the development of such scourges as cancer."*.

While both symmetry and replacebility are strong characteristics of coordination, neither principle holds universally. Coordination between syntactically dissimilar conjuncts is possible ("tomorrow and for the entirety of the next decade"), and the replacement principle fails in cases of ellipsis, gapping and others (*"The bank employs [8,000 people in Spain] and [2,000 abroad]"*).

2.3 Coordination in the PTB

Coordination annotation in the Penn Treebank (Marcus et al., 1993) is inconsistent (Hogan, 2007) and lacks internal structure for NPs with nominal modifiers (Bies et al., 1995). In addition, conjuncts in the PTB are not explicitly marked. These deficiencies led previous works on coordination disambiguation (Shimbo and Hara, 2007; Hara et al., 2009; Hanamoto et al., 2012) to use the Genia treebank of biomedical text (Ohta et al., 2002) which explicitly marks coordination phrases. However, using the Genia corpus is not ideal since it is in a specialized

domain and much smaller than the PTB. In this work we rely on a version of the PTB released by Ficler and Goldberg (2016) in which the above deficiencies are manually resolved. In particular, coordinating elements, coordination phrases and conjunct boundaries are explicitly marked with specialized function labels.

2.4 Neural Networks and Notation

We use $w_{1:n}$ to indicate a list of vectors $w_1, w_2, \ldots w_n$ and $w_{n:1}$ to indicate the reversed list. We use $\circ$ for vector concatenation. When a discrete symbol w is used as a neural network's input, the corresponding embedding vector is assumed.

A multi-layer perceptron (MLP) is a non linear classifier. In this work we take MLP to mean a classifier with a single hidden layer: $MLP(x) = V \cdot g(Wx + b)$ where x is the network's input, g is an activation function such as ReLU or Sigmoid, and W, V and b are trainable parameters. Recurrent Neural Networks (RNNs) (Elman, 1990) allow the representation of arbitrary sized sequences. In this work we use LSTMs (Hochreiter and Schmidhuber, 1997), a variant of RNN that was proven effective in many NLP tasks. LSTM$(w_{1:n})$ is the outcome vector resulting from feeding the sequence $w_{1:n}$ into the LSTM in order. A bi-directional LSTM (biLSTM) takes into account both the past $w_{1:i}$ and the future $w_{i:n}$ when representing the element in position i:

$$biLSTM(w_{1:n}, i) = LSTM_F(w_{1:i}) \circ LSTM_B(w_{n:i})$$

where $LSTM_F$ reads the sequence in its regular order and $LSTM_B$ reads it in reverse.

3 Task Definition and Architecture

Given a coordination word in a sentence, the coordination prediction task aims to returns the two conjuncts that are connected by it, or NONE if the word does not function as a coordinating conjunction of a relevant type.[1] Figure 1 provides an example.

Our system works in three phases: first, we determine if the coordinating word is indeed part of a conjunction of a desired type. We then extract a ranked list of candidate conjuncts, where a candidate is a pair of spans of the form $((i, j), (l, m))$. The candidates are then scored and the highest scoring pair is returned. Section 4 describes the scoring model, which is the main contribution of this work. The coordination classification and candidate extraction components are described in Section 5.

Sentence:
And$_1$ the city decided to treat its guests more like royalty or$_2$ rock stars than factory owners.
Expected output:
and$_1$: NONE
or$_2$: (11-11) royalty ; (12-13) rock stars

Sentence:
The president is expected to visit Minnesota, New York and$_1$ North Dakota by the end of the year.
Expected output:
and$_1$: (9-10) New York ; (12-13) North Dakota

Figure 1: The coordination prediction task.

4 Candidate Conjunctions Scoring

Our scoring model takes into account two signals, symmetry between conjuncts and the possibility of replacing the whole coordination phrase with its participating conjuncts.

4.1 The Symmetry Component

As noted in Section 2.1, many conjuncts spans have similar syntactic structure. However, while the similarity is clear to human readers, it is often not easy to formally define, such as in:

"about/IN half/NN its/PRP$ revenue/NN
and/CC
more/JJR than/IN half/NN its/PRP$ profit/NN"

If we could score the amount of similarity between two spans, we could use that to identify correct coordination structures. However, we do not know the similarity function. We approach this by training the similarity function in a data-dependent manner. Specifically, we train an encoder that encodes spans into vectors such that vectors of similar spans will have a small Euclidean distance between them. This architecture is similar to Siamese Networks, which are used for learning similarity functions in vision tasks (Chopra et al., 2005).

[1] We consider *and*, *or*, *but*, *nor* as coordination words. In case of more than two coordinated elements (conjuncts), we focus on the two conjuncts which are closest to the coordinator.

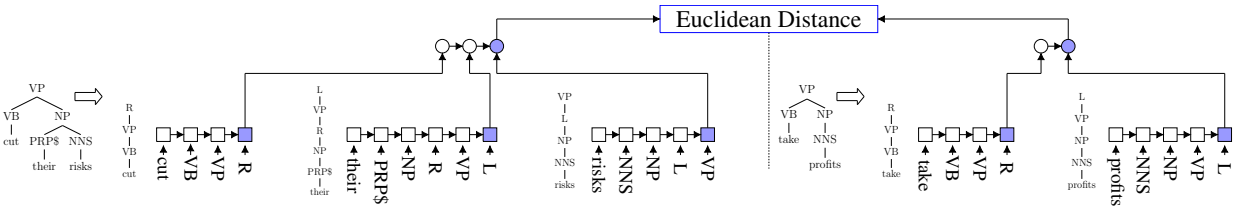


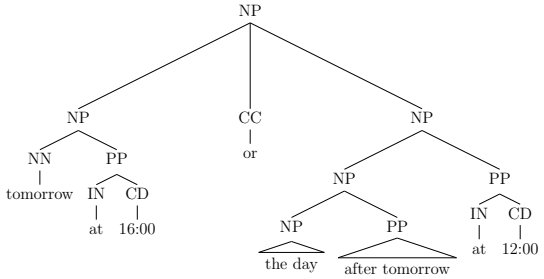
Figure 2: Illustration of the symmetry scoring component that takes into account the conjuncts syntactic structures. Each conjunct tree is decomposed into paths that are fed into the path-LSTMs (squares). The resulting vectors are fed into the symmetry LSTM function (circles). The outcome vectors (blue circles) are then fed into the euclidean distance function.

Given two spans of lengths k and m with corresponding vector sequences $u_{1:k}$ and $v_{1:m}$ we encode each sequences using an LSTM, and take the euclidean distance between the resulting representations:

$$Sym(u_{1:k}, v_{1:m}) = ||LSTM(u_{1:k}) - LSTM(v_{1:m})||$$

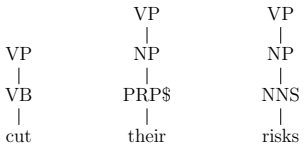
The network is trained such that the distance is minimized for compatible spans and large for incompatible ones in order to learn that vectors that represent correct conjuncts are closer than vectors that do not represent conjuncts.

What are the elements in the sequences to be compared? One choice is to take the vectors u_i to correspond to embeddings of the ith POS in the span. This approach works reasonably well, but does not consider the conjuncts' syntactic structure, which may be useful as symmetry often occurs on a higher level than POS tags. For example, in:

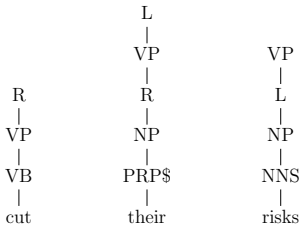


the similarity is more substantial in the third level of the tree than in the POS level.

A way to allow the model access to higher levels of syntactic symmetry is to represent each word as the projection of the grammatical functions from the word to the root.[2] For example, the projections for the first conjunct in Figure 2 are:



This decomposition captures the syntactic context of each word, but does not uniquely determine the structure of the tree. To remedy this, we add to the paths special symbols, R and L, which marks the lowest common ancestors with the right and left words respectively. These are added to the path above the corresponding nodes. For example consider the following paths which corresponds to the above syntactic structure:



The lowest common ancestor of *"their"* and *"risks"* is NP. Thus, R is added after NP in the path of *"their"* and L is added after NP in the path of *"risks"*. Similarly, L and R are added after the VP in the *"their"* and *"cut"* paths.

The path for each word is encoded using an LSTM receiving vector embeddings of the elements in the path from the word to the root. We then use the resulting encodings instead of the POS-tag embeddings as input to the LSTMs in the similarity function. Figure 2 depicts the complete process for the spans "cut their risks" and "take profits".

Using syntactic projections requires the syntactic structures of the conjuncts. This is obtained by running the Berkeley parser over the sentence and taking the subtree with the highest probability from the

[2] Similar in spirit to the spines used in Carreras et al. (2008) and Shen et al. (2003).

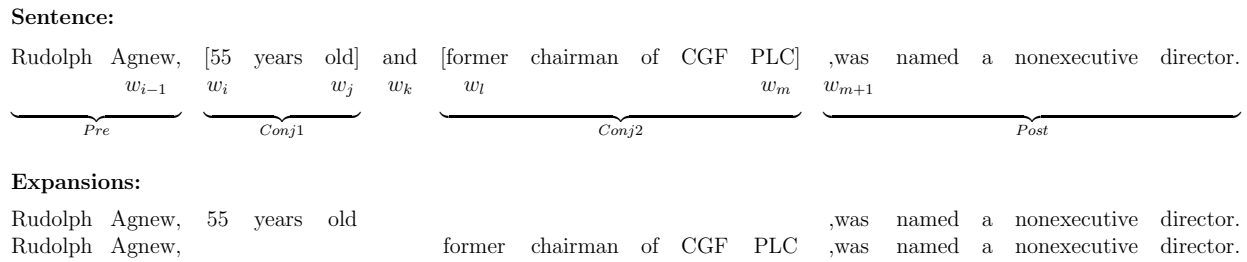


Figure 3: The correct conjuncts spans of the coordinator *and* in the sentence and the outcome expansions.

corresponding cell in the CKY chart.[3] In both approaches, the POS embeddings are initialized with vectors that are pre-trained by running word2vec (Mikolov et al., 2013) on the POS sequences in PTB training set.

4.2 The Replacement Component

The replacement component is based on the observation that, in many cases, the coordination phrase can be replaced with either one of its conjuncts while still preserving a grammatical and semantically coherent sentence (Section 2.2)

When attempting such a replacement on incorrect conjuncts, the resulting sentence is likely to be either syntactically or semantically incorrect. For example, in the following erroneous analysis: *"Rudolph Agnew, [55 years old] and [former chairman] of Consolidated Gold Fields PLC"* replacing the conjunction with the first conjunct results in the semantically incoherent sequence *"Rudolph Agnew, 55 years old of Consolidated Golden Fields, PLC"*.[4]

Our goal is to distinguish replacements resulting from correct conjuncts from those resulting from erroneous ones. To this end, we focus on the *connection points*. A connection point in a resulting sentence is the point where the sentence splits into two sequences that were not connected in the original sentence. For example, consider the sentence in Figure 3. It has four parts, marked as *Pre*, *Conj1*, *Conj2* and *Post*. Replacing the coordination phrase *Conj1 and Conj2* with *Conj2* results in a connection point between *Pre* and *Conj2*. Likewise, replacing the coordination phrase with *Conj1* results in connection point between *Conj1* and *Post*.

In order to model the validity of the connection points, we represent each connection point as the concatenation of a forward and reverse LSTMs centered around that point. Specifically, for the spans in Figure 3 the two connection points are represented as:

$LSTM_F$(Rudolph,...,old)$\circ LSTM_B$(director,...,was,,)
and
$LSTM_F$(Rudolph,Agnew,,)$\circ LSTM_B$(director,...,former)

Formally, assuming words $w_{1:n}$ in a sentence with coordination at position k and conjuncts $w_{i:j}$ and $w_{l:m}$,[5] the connection points are between $w_{1:j}$ and $w_{m+1:n}$; and between $w_{1:i-1}$ and $w_{l:n}$. The two connection points representations are then concatenated, resulting in a *replacement vector*:

$$\textsc{Repl}(w_{1:n}, i, j, l, m) = \textsc{ConPoint}(w_{1:n}, i-1, l) \circ \textsc{ConPoint}(w_{1:n}, j, m+1)$$

where:

$$\textsc{ConPoint}(w_{1:n}, i, j) = LSTM_F(w_{1:i}) \circ LSTM_B(w_{n:j})$$

We use two variants of the replacement vectors, corresponding to two levels of representation. The first variant is based on the sentence's words, while the second is based on its POS-tags.

4.3 Parser based Features

In addition to the symmetry and replacement signals, we also incorporate some scores that are derived from the Berkeley parser. As detailed in Section 5, a list of conjuncts candidates are extracted

[3] The parser's CKY chart did not include a tree for 10% of the candidate spans, which have inside probability 0 and outside probability > 0. For those, we obtained the syntactic structure by running the parser on the span words only.

[4] While uncommon, incorrect conjuncts may also result in valid sentences, e.g. *"He paid $ 7 for cold [drinks] and [pizza] that just came out of the oven."*

[5] Usually $j = k - 1$ and $l = k + 1$, but in some cases punctuation symbols may interfere.

from the CKY chart of the parser. The candidates are then sorted in descending order according to the multiplication of inside and outside scores of the candidate's spans:[6] $I_{(i,j)} \times O_{(i,j)} \times I_{(l,m)} \times O_{(l,m)}$. Each candidate $\{(i,j),(l,m)\}$ is assigned two numerical features based on this ranking: its position in the ranking, and the ratio between its score and the score of the adjacent higher-ranked candidate. We add an additional binary feature indicating whether the candidate spans are in the 1-best tree predicted by the parser. These three features are denoted as $Feats(i,j,l,m)$.

4.4 Final Scoring and Training

Finally, the score of a candidate $\{(i,j),(l,m)\}$ in a sentence with words $w_{1:n}$ and POS tags $p_{1:n}$ is computed as:

$$\begin{aligned}&\textsc{Score}(w_{1:n}, p_{1:n}, \{(i,j),(l,m)\}) = \\ &\quad MLP(\\ &\qquad Sym(v^{Path}_{i:j}, v^{Path}_{l:m}) \\ &\qquad \circ\, Repl(w_{1:n}, i, j, l, m) \\ &\qquad \circ\, Repl(p_{1:n}, i, j, l, m) \\ &\qquad \circ\, Feats(i, j, l, m)\ \ )\end{aligned}$$

where $v^{Path}_{i:j}$ and $v^{Path}_{l:m}$ are the vectors resulting from the path LSTMs, and Sym, $Repl$ and $Feats$ are the networks defined in Sections 4.1 – 4.3 above. The network is trained jointly, attempting to minimize a pairwise ranking loss function, where the loss for each training case is given by:

$$loss = max(0, 1 - (\hat{y} - y_g))$$

where $\hat{y}$ is the highest scoring candidate and y_g is the correct candidate. The model is trained on all the coordination cases in Section 2–21 in the PTB.

5 Candidates Extraction and Supporting Classifiers

Candidates Extraction We extract candidate spans based on the inside-outside probabilities assigned by the Berkeley parser. Specifically, to obtain candidates for conjunct span we collect spans that are marked with COORD, are adjacent to the coordinating word, and have non-zero inside or outside probabilities. We then take as candidates all possible pairs of collected spans. On the PTB dev set, this method produces 6.25 candidates for each coordinating word on average and includes the correct candidates for 94% of the coordinations.

Coordination Classification We decide whether a coordination word w_k in a sentence $w_{1:n}$ functions as a coordinator by feeding the biLSTM vector centered around w_k into a logistic classifier:

$$\sigma(v \cdot \text{biLSTM}(w_{1:n}, k) + b).$$

The training examples are all the coordination words (marked with CC) in the PTB training set. The model achieves 99.46 F1 on development set and 99.19 F1 on test set.

NP coordinations amount to about half of the coordination cases in the PTB, and previous work is often evaluated specifically on NP coordination. When evaluating on NP coordination, we depart from the unrealistic scenario used in most previous work where the type of coordination is assumed to be known a-priori, and train a specialized model for predicting the coordination type. For a coordination candidate $\{(i,j),(l,m)\}$ with a coordinator w_k, we predict if it is NP coordination or not by feeding a logistic classifier with a biLSTM vector centered around the coordinator and constrained to the candidate spans:

$$\sigma(v \cdot \text{biLSTM}(w_{i:m}, k) + b).$$

The training examples are coordinations in the PTB training set, where where a coordinator is considered of type NP if its head is labeled with NP or NX. Evaluating on gold coordinations results in F1 scores of 95.06 (dev) and 93.89 (test).

6 Experiments

We evaluate our models on their ability to identify conjunction boundaries in the extended Penn Treebank (Ficler and Goldberg, 2016) and Genia Treebank (Ohta et al., 2002)[7].

When evaluating on the PTB, we compare to the conjunction boundary predictions of the generative

[6] Inside-Outside probabilities (Goodman, 1998) represent the probability of a span with a given non-terminal symbol. The inside probability $I_{(N,i,j)}$ is the probability of generating words $w_i, w_{i+1}, ..., w_j$ given that the root is the non-terminal N. The outside probability $O_{(N,i,j)}$ is the probability of generating words $w_1, w_2, ..., w_{i-1}$, the non-terminal N and the words $w_{j+1}, w_{j+2}, ..., w_n$ with the root S.

[7] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

	Dev			Test		
	P	R	F	P	R	F
Berkeley	70.14	70.72	70.42	68.52	69.33	68.92
Zpar	72.21	72.72	**72.46**	68.24	69.42	68.82
Ours	72.34	72.25	72.29	72.81	72.61	**72.7**

Table 1: Coordination prediction on PTB (All coordinations).

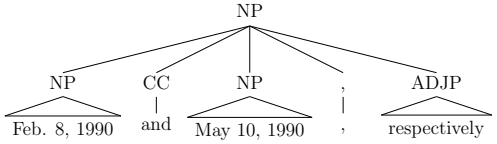
	Dev			Test		
	P	R	F	P	R	F
Berkeley	67.53	70.93	69.18	69.51	72.61	71.02
Zpar	69.14	72.31	70.68	69.81	72.92	71.33
Ours	75.17	74.82	**74.99**	76.91	75.31	**76.1**

Table 2: Coordination prediction on PTB (NP coordinations).

Berkeley parser (Petrov et al., 2006) and the discriminative Zpar parser (Zhang and Clark, 2011). When evaluating on the Genia treebank, we compare to the results of the discriminative coordination-prediction model of Hara et al. (2009).[8]

6.1 Evaluation on PTB

Baseline Our baseline is the performance of the Berkeley and Zpar parsers on the task presented in Section 3, namely: for a given coordinating word, determine the two spans that are being conjoined by it, and return NONE if the coordinator is not conjoining spans or conjoins spans that are not of the expected type. We convert predicted trees to conjunction predictions by taking the two phrases that are immediately adjacent to the coordinator on both sides (ignoring phrases that contain solely punctuation). For example, in the following Zpar-predicted parse tree the conjunct prediction is (*"Feb. 8, 1990"*,*"May 10, 1990"*).



Cases in which the coordination word is the left-most or right-most non-punctuation element in its phrase (e.g. `(PRN (P -)(CC and)(S it's been painful)(P -)))` are considered as no-coordination ("None").

[8] Another relevant model in the literature is (Hanamoto et al., 2012), however the results are not directly comparable as they use a slightly different definition of conjuncts, and evaluate on a subset of the Genia treebank, containing only trees that were properly converted to an HPSG formalism.

We consider two setups. In the first we are interested in all occurrences of coordination, and in the second we focus on NP coordination. The second scenario requires typed coordinations. We take the type of a parser-predicted coordination to be the type of the phrase immediately dominating the coordination word.

Evaluation Metrics We measure precision and recall compared to the gold-annotated coordination spans in the extended PTB, where an example is considered correct if both conjunct boundaries match exactly. When focusing on NPs coordinations, the type of the phrase above the CC level must match as well, and phrases of type NP/NX are considered as NP coordination.

Results Tables (1) and (2) summarize the results. The Berkeley and Zpar parsers perform similarly on the coordination prediction task. Our proposed model outperforms both parsers, with a test-set F_1 score of 72.7 (3.78 F_1 points gain over the better parser) when considering all coordinations, and test-set F_1 score of 76.1 (4.77 F_1 points gain) when considering NP coordination.

6.2 Evaluation on Genia

To compare our model to previous work, we evaluate also on the Genia treebank (Beta), a collection of constituency trees for 4529 sentences from Medline abstracts. The Genia treebank coordination annotation explicitly marks coordination phrases with a special function label (COOD), making the corpus an appealing resource for previous work on coordination boundary prediction (Shimbo and Hara, 2007; Hara et al., 2009; Hanamoto et al., 2012). Following Hara et al. (2009), we evaluate the models' ability to predict the span of the entire coordination phrase, disregarding the individual conjuncts. For example, in *"My plan is to visit Seychelles, ko Samui and Sardinia by the end of the year"* the goal is to recover *"Seychelles, ko Samui and Sardinia"*. This is a recall measure. We follow the exact protocol of Hara et al. (2009) and train and evaluate the model on 3598 coordination phrases in Genia Treebank Beta and report the micro-averaged results of a five-fold cross validation run.[9] As shown by Hara

[9] We thank Kazuo Hara for providing us with the exact details of their splits.

Sym	Correct:	Retail sales volume was [down 0.5% from the previous three months] and [up 1.2% from a year earlier].
	Incorrect:	Everyone was concerned about the [general narrowness of the rally] and [failure of the OTC market] to get into plus territory.
Rep_w	Correct:	The newsletter said [she is 44 years old] and [she studied at the University of Puerto Rico School of Medicine].
	Incorrect:	But Robert Showalter said no special [bulletins] or [emergency meetings of the investors' clubs] are planned .
Rep_p	Correct:	[On the Big Board floor] and [on trading desks], traders yelped their approval.
	Incorrect:	It suddenly burst upward 7.5 as Goldman, Sachs & Co. [stepped in] and [bought almost] every share offer, traders said.

Figure 4: Correct in incorrect predictions by the individual components.

COOD	#	Our Model	Hara et al.
Overall	3598	**64.14**	61.5
NP	2317	**65.08**	64.2
VP	465	**71.82**	54.2
ADJP	321	74.76	**80.4**
S	188	17.02	**22.9**
PP	167	56.28	**59.9**
UCP	60	**51.66**	36.7
SBAR	56	**91.07**	51.8
ADVP	21	80.95	**85.7**
Others	3	33.33	**66.7**

Table 3: Recall on the Beta version of Genia corpus. Numbers for Hara et al. are taken from their paper.

	All types			NPs		
	P	R	F	P	R	F
Sym	67.13	67.06	67.09	69.69	72.08	70.86
Rep_p	69.26	69.18	69.21	69.73	71.16	70.43
Rep_w	56.97	56.9	56.93	59.78	64.3	61.95
Feats	70.92	70.83	70.87	72.23	73.22	72.72
Joint	72.34	72.25	**72.29**	75.17	74.82	**74.99**

Table 4: Performance of the individual components on PTB section 22 (dev). Sym: Symmetry. Rep_p: POS replacement. Rep_w: Word replacement. Feats: features extracted from Berkeley parser. Joint: the complete model.

et al. (2009), syntactic parsers do not perform well on the Genia treebank. Thus, in our symmetry component we opted to not rely on predicted tree structures, and instead use the simpler option of representing each conjunct by its sequence of POS tags. To handle coordination phrases with more than two conjuncts, we extract candidates which includes up to 7 spans and integrate the first and the last span in the model features. Like Hara et al., we use gold POS.

Results Table 3 summarizes the results. Our proposed model achieves Recall score of 64.14 (2.64 Recall points gain over Hara et al.) and significantly improves the score of several coordination types.

6.3 Technical Details

The neural networks (candidate scoring model and supporting classifiers) are implemented using the pyCNN package.[10].

In the supporting models we use words embedding of size 50 and the Sigmoid activation function. The LSTMs have a dimension of 50 as well. The models are trained using SGD for 10 iterations over the train-set, where samples are randomly shuffled before each iteration. We choose the model with the highest F1 score on the development set.

All the LSTMs in the candidate scoring model have a dimension of 50. The input vectors for the symmetry LSTM is of size 50 as well. The MLP in the candidate scoring model uses the Relu activation function, and the model is trained using the Adam optimizer. The words and POS embeddings are shared between the symmetry and replacment components. The syntactic label embeddings are for the path-encoding LSTM, We perform grid search with 5 different seeds and the following: [1] MLP hidden layer size (100, 200, 400); [2] input embeddings size for words, POS and syntactic labels (100, 300). We train for 20 iterations over the train set, randomly shuffling the examples before each iteration. We choose the model that achieves the highest F1 score on the dev set.

[10]https://github.com/clab/cnn/tree/master/pycnn

7 Analysis

Our model combines four signals: symmetry, word-level replacement, POS-level replacement and features from Berkeley parser. Table 4 shows the PTB dev-set performance of each sub-model in isolation. On their own, each of the components' signals is relatively weak, seldom outperforming the parsers. However, they provide complementary information, as evident by the strong performance of the joint model. Figure 4 lists correct and incorrect predictions by each of the components, indicating that the individual models are indeed capturing the patterns they were designed to capture – though these patterns do not always lead to correct predictions.

8 Related Work

The similarity property between conjuncts was explored in several previous works on coordination disambiguation. Hogan (2007) incorporated this principle in a generative parsing model by changing the generative process of coordinated NPs to condition on properties of the first conjunct when generating the second one. Shimbo and Hara (2007) proposed a discriminative sequence alignment model to detect similar conjuncts. They focused on disambiguation of non-nested coordination based on the learned edit distance between two conjuncts. Their work was extended by Hara et al. (2009) to handle nested coordinations as well. The discriminative edit distance model in these works is similar in spirit to our symmetry component, but is restricted to sequences of POS-tags, and makes use of a sequence alignment algorithm. We compare our results to Hara et al.'s in Section 6.2. Hanamoto et al. (2012) extended the previous method with dual decomposition and HPSG parsing. In contrast to these symmetry-directed efforts, Kawahara et al. (2008) focuses on the dependency relations that surround the conjuncts. This kind of semantic information provides an additional signal which is complementary to the syntactic signals explored in our work. Our neural-network based model easily supports incorporation of additional signals, and we plan to explore such semantic signals in future work.

9 Conclusions

We presented an neural-network based model for resolving conjuncts boundaries. Our model is based on the observation that (a) conjuncts tend to be similar and (b) that replacing the coordination phrase with a conjunct results in a coherent sentence. Our models rely on syntactic information and do not incorporate resources external to the training treebanks, yet improve over state-of-the-art parsers on the coordination boundary prediction task.

Acknowledgments

This work was supported by The Israeli Science Foundation (grant number 1555/15) as well as the German Research Foundation via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

David Dowty. 1988. Type raising, functional composition, and non-constituent conjunction. In *Categorial grammars and natural language structures*, pages 153–197. Springer.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jessica Ficler and Yoav Goldberg. 2016. Coordination annotation extension in the penn tree bank. *Association for Computational Linguistics*.

Joshua Goodman. 1998. Parsing inside-out. *arXiv preprint cmp-lg/9805007*.

Atsushi Hanamoto, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Coordination structure analysis using dual decomposition. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 430–438. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 967–975. Association for Computational Linguistics.

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

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. Association for Computational Linguistics.

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, page 1275.

Daisuke Kawahara and Sadao Kurohashi. 2008. Coordination disambiguation without any similarities. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 425–432. Association for Computational Linguistics.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, pages 82–86. Morgan Kaufmann Publishers Inc.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Libin Shen, Anoop Sarkar, and Aravind K Joshi. 2003. Using ltag based features in parse reranking. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 89–96. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *EMNLP-CoNLL*, pages 610–619.

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

Using Left-corner Parsing to Encode Universal Structural Constraints in Grammar Induction

Hiroshi Noji
Graduate School of Information Science
Nara Institute of Science and Technology
noji@is.naist.jp

Yusuke Miyao
National Institute of Informatics
yusuke@nii.ac.jp

Mark Johnson
Department of Computing
Macquarie University
mark.johnson@mq.edu.au

Abstract

Center-embedding is difficult to process and is known as a rare syntactic construction across languages. In this paper we describe a method to incorporate this assumption into the grammar induction tasks by restricting the search space of a model to trees with limited center-embedding. The key idea is the tabulation of *left-corner* parsing, which captures the degree of center-embedding of a parse via its stack depth. We apply the technique to learning of famous generative model, the dependency model with valence (Klein and Manning, 2004). Cross-linguistic experiments on Universal Dependencies show that often our method boosts the performance from the baseline, and competes with the current state-of-the-art model in a number of languages.

1 Introduction

Human languages in the world are divergent, but they also exhibit many striking similarities (Greenberg, 1963; Hawkins, 2014). At the level of syntax, one attractive hypothesis for such regularities is that any grammars of languages have evolved under the pressures, or biases, to avoid structures that are difficult to process. For example it is known that many languages have a preference for shorter dependencies (Gildea and Temperley, 2010; Futrell et al., 2015), which originates from the difficulty in processing longer dependencies (Gibson, 2000).

Such syntactic regularities can also be useful in applications, in particular in unsupervised (Klein and Manning, 2004; Mareček and Žabokrtský, 2012; Bisk and Hockenmaier, 2013) or weakly-supervised (Garrette et al., 2015) grammar induction tasks, where the models try to recover the syntactic structure of language without access to the syntactically annotated data, e.g., from raw or part-of-speech tagged text only. In these settings, finding better syntactic regularities universal across languages is essential, as they work as a small cue to the correct linguistic structures. A preference exploited in many previous works is favoring shorter dependencies, which has been encoded in various ways, e.g., initialization of EM (Klein and Manning, 2004), or model parameters (Smith and Eisner, 2006), and this has been the key to success of learning (Gimpel and Smith, 2012).

In this paper, we explore the utility for another universal syntactic bias that has not yet been exploited in grammar induction: a bias against center-embedding. Center-embedding is a syntactic construction on which a clause is embedded into another one. An example is "*The reporter [who the senator [who Mary met] attacked] ignored the president.*", where "*who Mary met*" is embedded in a larger relative clause. These constructions are known to cause memory overflow (Miller and Chomsky, 1963; Gibson, 2000), and also are rarely observed cross-linguistically (Karlsson, 2007; Noji and Miyao, 2014). Our learning method exploits this universal property of language. Intuitively during learning our models explore the restricted search space, which excludes linguistically implausible trees, i.e., those with deeper levels of center-embedding.

We describe how these constraints can be imposed in EM with the inside-outside algorithm. The central

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 33–43,
Austin, Texas, November 1-5, 2016.

SHIFT	$\sigma^{d-1} \overset{a}{\mapsto} \sigma^{d-1}\|A^d$	$A \to a \in P$
SCAN	$\sigma^{d-1}\|B/A^d \overset{a}{\mapsto} \sigma^{d-1}\|B^d$	$A \to a \in P$
PRED	$\sigma^{d-1}\|A^d \overset{\varepsilon}{\mapsto} \sigma^{d-1}\|B/C^d$	$B \to A\,C \in P$
COMP	$\sigma^{d-1}\|D/B^d\|A^{d+1} \overset{\varepsilon}{\mapsto} \sigma^{d-1}\|D/C^d$	$B \to A\,C \in P$

Figure 1: A set of transitions in left-corner parsing. The rules on the right side are the side conditions, in which P is the set of rules of a given CFG.

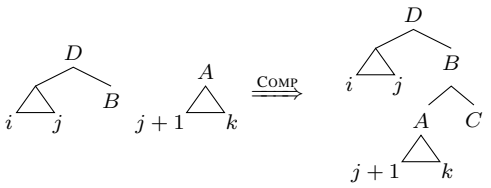


Figure 2: COMP combines two subtrees on the top of the stack. i, j, k are indices of spans.

idea is to tabulate *left-corner* parsing, on which its stack depth captures the degree of center-embedding of a partial parse. Each chart item keeps the current stack depth and we discard all items where the depth exceeds some threshold. The technique is general and can be applicable to any model on PCFG; in this paper, specifically, we describe how to apply the idea on the dependency model with valence (DMV) (Klein and Manning, 2004), a famous generative model for dependency grammar induction.

We focus our evaluation on grammar induction from part-of-speech tagged text, comparing the effect of several biases including the one against longer dependencies. Our main empirical finding is that though two biases, avoiding center-embedding and favoring shorter dependencies, are conceptually similar (both favor simpler grammars), often they capture different aspects of syntax, leading to different grammars. In particular our bias cooperates well with additional small syntactic cue such as the one that the sentence root tends to be a verb or a noun, with which our models compete with the strong baseline relying on a larger number of hand crafted rules on POS tags (Naseem et al., 2010).

Our contributions are: the idea to utilize left-corner parsing for a tool to constrain the models of syntax (Section 3), the formulation of this idea for DMV (Section 4), and cross-linguistic experiments across 25 languages to evaluate the universality of the proposed approach (Sections 5 and 6).

2 Left-corner parsing

We first describe (arc-eager) left-corner (LC) parsing as a push-down automaton (PDA), and then reformulate it as a grammar transform. In previous work this algorithm has been called *right-corner* parsing (e.g., Schuler et al. (2010)); we avoid this term and instead treat it as a variant of LC parsing following more recent studies, e.g., van Schijndel and Schuler (2013). The central motivation for this technique is to detect center-embedding in a parse efficiently. We describe this mechanism after providing the algorithm itself. We then give historical notes on LC parsing at the end of this section.

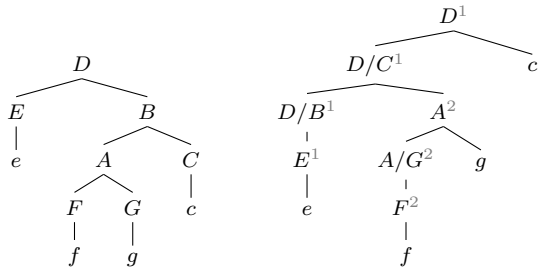
PDA Let us assume a CFG is given, and it is in CNF. We formulate LC parsing as a set of transitions between configurations, each of which is a pair of the stack and the input position (next input symbol). In Figure 1 a transition $\sigma_1 \overset{a}{\mapsto} \sigma_2$ means that the stack is changed from σ_1 to σ_2 by reading the next input symbol a. We use a vertical bar to signify the append operation, e.g., $\sigma = \sigma'|\sigma_1$ denotes σ_1 is the topmost symbol of σ. Each stack symbol is either a nonterminal, or a pair of nonterminals, e.g., A/B, which represents a subtree rooted at A and is awaiting symbol B. We also decorate each symbol with depth; for example, $\sigma^{d-1}|A^d$ means the current stack depth is d, and the depth of the topmost symbol in σ is $d-1$. The bottom symbol on the stack is always the empty symbol ε^0 with depth 0. Parsing begins with ε^0. Given the start symbol of CFG S, it finishes when S^1 is found on the stack.

The key transition here is COMP (Figure 2).[1] Basically the algorithm builds a tree by expanding the hypothesis from left to right. In COMP, a subtree rooted at A is combined with the second top subtree (D/B) on the stack. This can be done by first *predicting* that A's parent symbol is B and its sibling is C; then it unifies two different Bs to combine them. PRED is simpler, and it just predicts the parent and sibling symbols of A. The input symbols are read by SHIFT and SCAN: SHIFT addes a new element on the stack while SCAN fills in the predicted sibling symbol. For an example, Figure 3 shows how

[1] van Schijndel and Schuler (2013) employ different transition names, e.g., L- and L+; we avoid them as they are less informative.

Step	Transition	Stack	Next input symbol
0		ε	e
1	SHIFT	E^1	f
2	PRED	D/B^1	f
3	SHIFT	$D/B^1\ F^2$	g
4	PRED	$D/B^1\ A/G^2$	g
5	SCAN	$D/B^1\ A^2$	c
6	COMP	D/C^1	c
7	SCAN	D^1	

Figure 3: Sequence of transitions in LC PDA to parse the tree in Figure 4(a).



(a) (b)

Figure 4: An example of LC transform: (a) the original parse; and (b) the transformed parse.

this PDA works for parsing a tree in Figure 4(a).

Grammar transform The algorithm above can be reformulated as a grammar transform, which becomes the starting point for our application to grammar induction. This can be done by extracting the operated top symbols on the stack in each transition:

SHIFT: $A^d \rightarrow a \quad (A \rightarrow a \in P)$;
SCAN: $B^d \rightarrow B/A^d\ a \quad (A \rightarrow a \in P)$;
PRED: $B/C^d \rightarrow A^d \quad (B \rightarrow A\ C \in P)$;
COMP: $D/C^d \rightarrow D/B^d\ A^{d+1} \quad (B \rightarrow A\ C \in P)$.

where a rule on the right side is a condition given the set of rules P in the CFG.

Figure 4 shows an example of this transform. The essential point is that each CFG rule in the transformed parse (b) corresponds to a transition in the original algorithm (Figure 1). For example a rule $D/C^1 \rightarrow D/B^1\ A^2$ in the parse indicates that the stack configuration $D/B^1|A^2$ occurs during parsing (just corresponding to the step 5 in Figure 3) and COMP is then applied. This can also be seen as an instantiation of Figure 2.

Stack depth and center-embedding We use the term *center-embedding* to distinguish just the tree structures, i.e., ignoring symbols. That is, the tree in Figure 4(a) is the minimal, one degree of center-embedded tree, where the constituent rooted at A is embedded into a larger constituent rooted at D. Multiple, or degree ≥ 2 of center-embedding occurs if this constituent is also embedded into another larger constituent.

Note that it is only COMP that consumes the top *two* symbols on the stack. This means that a larger stack depth occurs only when COMP is needed. Furthermore, from Figure 2 COMP always induces a subtree involving new center-embedding, and this is the underlying mechanism that the stack depth of the algorithm captures the degree of center-embedding.

One thing to note is that to precisely associate the stack depth and the degree of center-embedding the depth calculation in COMP should be revised as:

$$\text{COMP: } D/C^d \rightarrow D/B^d\ A^{d'} \quad (B \rightarrow A\ C \in P)$$
$$d' = \begin{cases} d & (\textsc{SpanLen}(A) = 1) \\ d+1 & (\text{otherwise}), \end{cases} \quad (1)$$

where $\textsc{SpanLen}(A)$ calculates the span length of the constituent rooted at A, which is 2 in Figure 4(b). This modification is necessary since COMP for a single token occurs for building purely right-branching structures.[2] Formally, then, given a tree with degree λ of center-embedding the largest stack depth d^* during parsing this tree is: $d^* = \lambda + 1$.

Schuler et al. (2010) found that on English treebanks larger stack depth such as 3 or 4 rarely occurs while Noji and Miyao (2014) validated the language universality of this observation through cross-linguistic experiments. These suggest we may utilize LC parsing as a tool for exploiting universal syntactic biases as we discuss in Section 3.

Historical notes Rosenkrantz and Lewis (1970) first presented the idea of LC parsing as a grammar transform. This is *arc-standard*, and has no relevance to center-embedding; Resnik (1992) and Johnson (1998) formulated an *arc-eager* variant by extending this algorithm. The presented algorithm here is the same as Schuler et al. (2010), and is slightly different from Johnson (1998). The difference is in the start and end conditions: while

[2] Schuler et al. (2010) skip this subtlety by only concerning stack depth after PRED or COMP. We do not take this approach since ours allows a flexible extension described in Section 3.

our parser begins with an empty symbol, Johnson's parser begins with the predicted start symbol, and finishes with an empty symbol.

3 Learning with structural constraints

Now we discuss how to utilize LC parsing for grammar induction in general. An important observation in the above transform is that if we perform chart parsing, e.g., CKY, we can detect center-embedded trees efficiently in a chart. For example, by setting a threshold of stack depth δ, we can eliminate any parses involving center-embedding up to degree $\delta-1$. Note that in a probabilistic setting, each weight of a transformed rule comes from the corresponding underlying CFG rule (i.e., the condition).

For learning, our goal is to estimate θ of a generative model $p(z, x|\theta)$ for parse z and its yields (words) x. We take an EM-based simplest approach, and multiply the original model by a constraint factor $f(z, x) \in [0, 1]$ to obtain a new model:

$$p'(z, x|\theta) \propto p(z, x|\theta) f(z, x), \tag{2}$$

and then optimize θ based on $p'(z, x|\theta)$. This is essentially the same approach as Smith and Eisner (2006). As shown in Smith (2006), when training with EM we can increase the likelihood of $p'(z, x|\theta)$ by just using the expected counts from an E-step on the unnormalized distribution $p(z, x|\theta) f(z, x)$.

We investigate the following constraints in our experiments:

$$f(z, x) = \begin{cases} 0 & (d^*_z > \delta) \\ 1 & (\text{otherwise}), \end{cases} \tag{3}$$

where d^*_z is the largest stack depth for z in LC parsing and δ is the threshold. This is a hard constraint, and can easily be achieved by removing all chart items (of LC transformed grammar) on which the depth of the symbol exceeds δ. For example, when $\delta = 1$ the model only explores trees without center-embedding, i.e., right- or left-linear trees.

Length-based constraints By $\delta = 2$, the model is allowed to explore trees with one degree of center-embedding. Besides these simple ones, we also investigate relaxing $\delta = 1$ that results in an intermediate between $\delta = 1$ and 2. Specifically, we relax the depth calculation in COMP (Eq. 1) as follows:

$$d' = \begin{cases} d & (\textsc{SpanLen}(A) \leq \xi) \\ d + 1 & (\text{otherwise}), \end{cases} \tag{4}$$

where $\xi \geq 1$ controls the minimal length of a span regarded as *embedded* into another one. For example, when $\xi = 2$, the parse in Figure 4(a) is *not* regarded as center-embedded because the span length of the constituent reduced by COMP (i.e., A) is 2.

This modification is motivated with our observation that in many cases center-embedded constructions arise due to embedding of small chunks, rather than clauses. An example is "*... prepared [the cat 's] dinner*", where "*the cat 's*" is center-embedded in our definition. For this sentence, by relaxing the condition with, e.g., $\xi = 3$, we can suppress the increase of stack depth. We treat ξ as a hyperparameter in our experiments, and in practice, we find that this relaxed constraint leads to higher performance.

4 Dependency grammar induction

In this section we discuss how we can formulate the dependency model with valence (DMV) (Klein and Manning, 2004), a famous generative model for *dependency* grammar induction, on LC parsing. Though as we will see, applying LC parsing for a dependency model is a little involved compared to simple PCFG models, dependency models have been the central for the grammar induction tasks, and we consider it is most appropriate for assessing effectiveness of our approach.

DMV is a head-outward generative model of a dependency tree, controlled by two types of multinomial distributions. For *stop* $\in \{\textsc{stop}, \neg\textsc{stop}\}$, $\theta_{\textsc{s}}(\textit{stop}|h, \textit{dir}, \textit{adj})$ is a Bernoulli random variable to decide whether or not to attach further dependents in *dir* $\in \{\leftarrow, \rightarrow\}$ direction. The adjacency *adj* $\in \{\textsc{true}, \textsc{false}\}$ is the key factor to distinguish the distributions of the first and the other dependents, which is TRUE if h has no dependent yet in *dir* direction. Another type of parameter is $\theta_{\textsc{a}}(a|h, \textit{dir})$, a probability that h takes a as a dependent in *dir* direction.

For this particular model, we take the following approach to formulate it in LC parsing: 1) converting a dependency tree into a binary CFG parse; 2) applying LC transform on it; and 3) encoding DMV

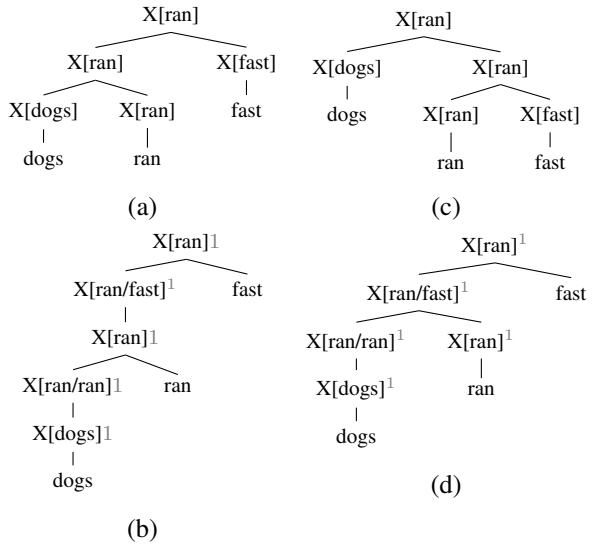


Figure 5: Two CFG parses for "dogs ran fast" and the results of LC transform ((a) → (b); (c) → (d)). X[a/b] is an abbreviation for X[a]/X[b].

parameters into each CFG rule of the transformed grammar.[3] Below we discuss a problem for (1) and (2), and then consider parameterization.[4]

Spurious ambiguity The central issue for applying LC parsing is the *spurious ambiguity* in dependency grammars. That is, there are more than one (binary) CFG parses corresponding to a given dependency tree. This is problematic mainly for two reasons: 1) we cannot specify the degree of center-embedding in a dependency tree uniquely; and 2) this one-to-many mapping prevents the inside-outside algorithm to work correctly (Eisner, 2000).

As a concrete example, Figures 5(a) and 5(c) show two CFG parses corresponding to the dependency tree *dogs*↶*ran*↷*fast*. We approach this problem by first providing a grammar transform, which generates all valid LC transformed parses (e.g., Figures 5(b) and 5(d)) and then restricting the grammar

[3]Another approach might be just applying the technique in Section 3 to some PCFG that encodes DMV, e.g., Headden III et al. (2009). The problem with this approach, in particular with split-head grammars (Johnson, 2007), is that the calculated stack depth no longer reflects the degree of center-embedding in the original parse correctly. As we discuss later, instead, we can speed up inference by applying head-splitting *after* obtaining the LC transformed grammar.

[4]Technical details including the chart algorithm for split-head grammars can be found in the Ph.D. thesis of the first author (Noji, 2016).

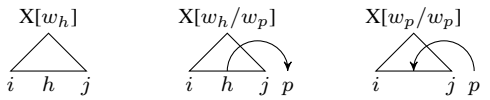


Figure 6: The senses of the symbols as a chart item. X[w_h/w_p] *predicts* the next dependent outside of the span while X[w_p/w_p] predicts the head.

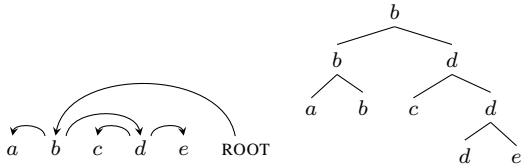


Figure 7: Implicit binarization of the restricted grammar. For each token, if its parent is in the right side (e.g., b), it attaches all left children first. The behavior is opposed when the parent is in its left (e.g., d). A dummy root token is placed at the end.

for generating particular parses only.

Naive method Let us begin with the grammar below, which suffers from the spurious ambiguity:

SHIFT: $\mathrm{X}[w_h]^d \to w_h$
SCAN: $\mathrm{X}[w_h]^d \to \mathrm{X}[w_h/w_p]^d\ w_p$
L-PRED: $\mathrm{X}[w_p/w_p]^d \to \mathrm{X}[w_h]^d\ (w_h \curvearrowleft w_p)$;
R-PRED: $\mathrm{X}[w_h/w_p]^d \to \mathrm{X}[w_h]^d\ (w_h \curvearrowright w_p)$;
L-COMP: $\mathrm{X}[w_h/w_p]^d \to \mathrm{X}[w_h/w_p]^d \mathrm{X}[w_a]^{d'}\ (w_a \curvearrowleft w_p)$;
R-COMP: $\mathrm{X}[w_h/w_a]^d \to \mathrm{X}[w_h/w_p]^d \mathrm{X}[w_p]^{d'}\ (w_p \curvearrowright w_a)$.

Here X[a/b] denotes X[a]/X[b] while w_h denotes the h-th word in the sentence w. We can interpret these rules as the operations on chart items (Figure 6). Note that only PRED and COMP create new dependency arcs and we divide them depending on the direction of the created arcs (L and R). d' is calculated by Eq. 4. Note also that for L-COMP and R-COMP h might equal p; X[ran/fast]^1 → X[ran/ran]^1 X[ran]^2 in Figure 5(d) is such a case for R-COMP.

Removing spurious ambiguity We can show that by restricting conditions for some rules, the spurious ambiguity can be eliminated (the proof is omitted).

1. Prohibit R-COMP when $h = p$;
2. Assume the span of $\mathrm{X}[w_p]^{d'}$ is (i, j) ($i \le p \le j$). Then allow R-COMP only when $i = p$.

Intuitively, these conditions constraint the order that each word collects its left and right children. For

example, by the condition 1, this grammar is prohibited to generate the parse of Figure 5(d).

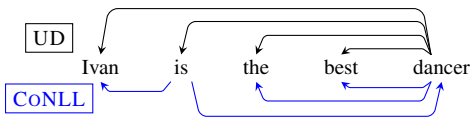
Binarization Note that two CFG parses in Figures 5(a) and 5(c) differ in how we *binarize* a given dependency tree. This observation indicates that our restricted grammar *implicitly* binarizes a dependency tree, and the incurred stack depth (or the degree of center-embedding) is determined based on the structure of the binarized tree. Specifically, we can show that the presented grammar performs *optimal* binarization; i.e., it *minimizes* the incurred stack depth. Figure 7 shows an example, which is not regarded as center-embedded in our procedure. In summary, our method detects center-embedding for a dependency tree, but the degree is determined based on the structure of the binarized CFG parse.

Parameterization We can encode DMV parameters into each rule. A new arc is introduced by one of {L/R}-{PRED/COMP}, and the stop probabilities can be assigned appropriately in each rule by calculating the valence from indices in the rule. For example, after L-PRED, w_h does not take any right dependents so $\theta_{\mathrm{S}}(stop|w_h, \rightarrow, h = j)$, where j is the right span index of $\mathrm{X}[w_h]$, is multiplied.

Improvement Though we omit the details, we can improve the time complexity of the above grammar from $O(n^6)$ to $O(n^4)$ applying the technique similar to Eisner and Satta (1999) without changing the binarization mechanism mentioned above. We implemented this improved grammar.

5 Experimental setup

A sound evaluation metric in grammar induction is known as an open problem (Schwartz et al., 2011; Bisk and Hockenmaier, 2013), which essentially arises from the ambiguity in the notion of head. For example, Universal dependencies (UD) is the recent standard in annotation and prefers content words to be heads, but as shown below this is very different from the conventional style, e.g., the one in CoNLL shared tasks (Johansson and Nugues, 2007):



The problem is that both trees are *correct* under some linguistic theories but the standard metric, unlabeled attachment score (UAS), only takes into account the annotation of the current gold data.

Our goal in this experiment is to assess the effect of our structural constraints. To this end, we try to eliminate such arbitrariness in our evaluation as much as possible in the following way:

- We experiment on UD, in which every treebank follows the consistent UD style annotation.
- We restrict the model to explore only trees that follow the UD style annotation during learning[5], by prohibiting every function word[6] in a sentence to have any dependents.
- We calculate UAS in a standard way.

We use UD of version 1.2. Some treebanks are very small, so we select the top 25 largest languages. The input to the model is coarse universal POS tags. Punctuations are stripped off. All models are trained on sentences of length ≤ 15 and tested on ≤ 40.

Initialization Much previous work of dependency grammar induction relies on the technique called harmonic initialization, which also biases the model towards shorter dependencies (Klein and Manning, 2004). Since our focus is to see the effect of structural constraints, we do not try this and initialize models uniformly. However, we add a baseline model with this initialization in our comparison to see the relative strength of our approach.

Models For the baseline, we employ a variant of DMV with *features* (Berg-Kirkpatrick et al., 2010), which is simple yet known to boost the performance well. The feature templates are almost the same; the only change is that we add backoff features for STOP probabilities that ignore both direction and adjacency, which we found slightly improves the performance in a preliminary experiment. We set the regularization parameter to 10 though in practice we found the model is less sensitive to this value. We run 100 iterations of EM for each setting. The dif-

[5] We remove the restriction at test time though we found it does not affect the performance.

[6] A word with one of the following POS tags: ADP, AUX, CONJ, DET, PART, and SCONJ.

ference of each model is then the type of constraints imposed during the E-step[7], or initialization:

- Baseline (FUNC): Function word constraints;
- HARM: FUNC with harmonic initialization;
- DEP: FUNC + stack depth constraints (Eq. 3);
- LEN: FUNC + soft dependency length bias, which we describe below.

For DEP, we use $\delta = 1.\xi$ to denote the relaxed maximum depth allowing span length up to ξ (Eq. 4).

LEN is the previously explored structural bias (Smith and Eisner, 2006), which penalizes longer dependencies by modifying each attachment score:

$$\theta'_{\text{A}}(a|h, dir) = \theta_{\text{A}}(a|h, dir) \cdot e^{-\gamma \cdot (|h-a|-1)}, \quad (5)$$

where γ (≥ 0) determines the strength of the bias and $|h - a|$ is (string) distance between h and a.

Note that DEP and LEN are closely related; generally center-embedded constructions are accompanied by longer dependencies so LEN also penalizes center-embedding implicitly. However, the opposite is not true and there exist many constructions with longer dependencies without center-embedding. By comparing these two settings, we discuss the worth of focusing on constraining center-embedding relative to the simpler bias on dependency length.

Finally we also add the system of Naseem et al. (2010) in our comparison. This system encodes many manually crafted rules between POS tags with the posterior regularization technique. For example, the model is encouraged to find NOUN → ADJ relationship. Our systems cannot access to these core grammatical rules so it is our strongest baseline.[8]

Constraining root word We also see the effects of the constraints when a small amount of grammatical rule is provided. In particular, we restrict the candidate root words of the sentence to a noun or a verb; similar rules have been encoded in past work such as Gimpel and Smith (2012) and the CCG induction system of Bisk and Hockenmaier (2013).

[7]We again remove the restrictions at decoding as we observed that the effects are very small.

[8]We encode the customized rules that follow UD scheme. The following 13 rules are used: ROOT → VERB, ROOT → NOUN, VERB → NOUN, VERB → ADV, VERB → VERB, VERB → AUX, NOUN → ADJ, NOUN → DET, NOUN → NUM, NOUN → NOUN, NOUN → CONJ, NOUN → ADP, ADJ → ADV.

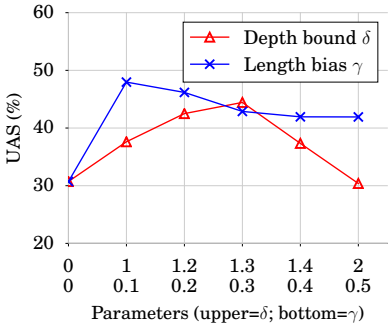


Figure 8: UAS for various settings on (UD) WSJ.

Hyperparameters Selecting hyperparameters in multilingual grammar induction is difficult; some works tune values for each language based on the development set (Smith and Eisner, 2006; Bisk et al., 2015), but this violates the assumption of unsupervised learning. We instead follow many works (Mareček and Žabokrtský, 2012; Naseem et al., 2010) and select the values with the English data. For this, we use the WSJ data, which we obtain in UD style from the Stanford CoreNLP (ver. 3.6.0).[9]

6 Experiments

WSJ Figure 8 shows the result on WSJ. Both DEP and LEN have one parameter: the maximum depth δ, and γ (Eq. 5), and the figure shows the sensitivity on them. Note that x-axis = 0 represents FUNC.

For LEN, we can see the optimal parameter γ is 0.1, and degrades the performance when increasing the value; i.e., the small bias is the best. For DEP, we find the best setting is 1.3, i.e., allowing embedded constituents of length 3 or less ($\xi = 3$ in Eq. 4). We can see that allowing depth 2 degrades the performance, indicating that depth 2 allows too many trees and does not reduce the search space effectively.[10]

Multilingual results Table 1 shows the main multilingual results. When we see "No root constraint" block, we notice that our DEP boosts the performance in many languages (e.g., Bulgarian, French,

[9]Note that the English data in UD is Web Treebank (Silveira et al., 2014), not the standard WSJ Penn treebank.

[10]We see the same effects when training with longer sentences (e.g., length ≤ 20). This is probably because a looser constraint does nothing for shorter sentences. In other words, the model can restrict the search space only for longer sentences, which are relatively small in the data.

	No root constraint				+ root constraint				
	FUNC	DEP	LEN	HARM	FUNC	DEP	LEN	HARM	N10
A-Greek	35.9	31.6	34.7	**37.8**	37.9	**45.0**	34.4	37.7	40.1
Arabic	48.6	38.7	**49.8**	42.8	45.9	44.3	**49.6**	31.4	37.8
Basque	41.7	**46.1**	45.0	24.9	42.5	44.8	**44.8**	25.3	50.1
Bulgarian	45.6	**69.0**	64.8	66.4	69.1	**71.1**	61.9	68.0	58.6
Croatian	40.8	32.2	**50.7**	47.8	40.7	42.2	47.6	**47.7**	41.0
Czech	56.0	**62.0**	52.7	53.7	47.2	**62.2**	56.0	52.2	52.0
Danish	42.5	42.7	42.3	**47.2**	42.6	42.8	42.3	**46.6**	42.8
Dutch	25.7	26.6	**28.0**	26.2	25.7	27.5	**28.7**	26.4	40.6
English	37.2	39.8	**52.1**	37.5	37.5	**40.0**	38.4	38.2	51.4
Estonian	68.5	67.4	68.0	**68.6**	68.0	67.8	65.1	**68.5**	67.3
Finnish	26.2	24.5	**27.9**	25.7	25.7	27.3	**27.9**	20.5	44.6
French	36.7	**48.0**	36.8	36.5	36.5	**54.6**	36.3	36.7	53.3
German	44.6	**48.0**	46.3	43.6	43.9	**50.4**	47.9	43.9	53.5
Hebrew	58.4	54.4	58.5	**59.1**	55.4	**59.7**	59.4	59.0	56.9
Hindi	54.7	52.6	16.0	**55.8**	**55.8**	52.6	48.8	55.7	55.8
Indonesian	36.0	**52.9**	45.6	40.1	30.4	**53.1**	40.5	40.0	51.1
Italian	63.8	67.8	**68.4**	65.0	63.1	65.7	**68.8**	62.9	56.3
Japanese	46.8	44.5	**73.8**	47.9	47.6	46.7	**72.3**	47.9	51.3
Latin-ITT	42.3	**43.8**	42.1	41.0	42.4	**43.7**	38.4	41.6	38.4
Norwegian	44.7	45.3	45.1	**51.9**	44.8	45.4	45.2	**45.7**	55.4
Persian	**44.9**	39.0	37.3	36.6	44.1	**46.6**	37.2	43.6	55.2
Portuguese	48.4	61.1	**61.6**	55.9	49.2	61.1	**61.4**	44.6	47.1
Slovenian	**65.6**	61.0	50.1	62.7	**65.1**	60.7	49.4	63.6	53.1
Spanish	52.2	54.6	**62.5**	49.1	44.4	53.8	**60.0**	48.4	55.3
Swedish	42.7	48.1	**51.4**	48.1	43.1	42.8	42.7	**47.6**	46.7
Avg	46.0	48.1	**48.5**	46.9	45.9	**50.1**	48.2	45.8	50.2

Table 1: Attachment scores on UD with or without root POS constraints. A-Greek = Ancient Greek. N10 = Naseem et al. (2010) with modified rules.

Indonesian, and Portuguese), though LEN performs equally well and in average, LEN performs slightly better. Harmonic initialization does not work well.

We then move on to the settings with the constraint on root tags. Interestingly, in these settings DEP performs the best. The model competes with Naseem et al.'s system in average, and outperforms it in many languages, e.g., Bulgarian, Czech, etc. LEN, on the other hand, decreases the average score.

Analysis Why does DEP perform well in particular with the restriction on root candidates? To shed light on this, we inspected the output parses of English with no root constraints, and found that the types of errors are very different across constraints.

Figure 9 shows a typical example of the difference. One difference between trees is in the constructions of phrase "On ... pictures". LEN predicts that "On the next two" comprises a constituent, which modifies "pictures" while DEP predicts that "the ... pictures" comprises a constituent, which is correct, although the head of the determiner is incorrectly predicted. On the other hand, LEN works well to find more primitive dependency arcs between POS tags, such as arcs from verbs to nouns, which are often incorrectly recognized by DEP.

These observations may partially answer the

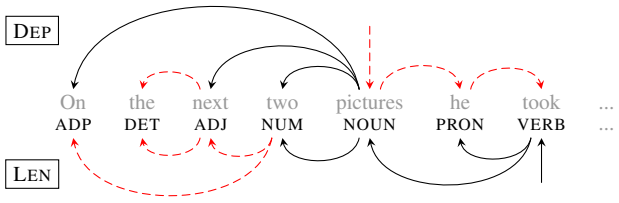


Figure 9: A comparison of output parses by DEP and LEN (with no root constraints). Dashed arcs are misclassified ones.

	Prec.	Recall	F1
FUNC (English)	11.6	18.4	14.1
DEP (English)	**22.4**	**37.1**	**27.9**
LEN (English)	21.6	31.0	25.5
FUNC (Avg.)	22.5	30.0	25.6
DEP (Avg.)	**27.8**	**34.5**	**30.5**
LEN (Avg.)	24.0	33.7	27.9
FUNC + ROOT (Avg.)	22.0	29.4	25.0
DEP + ROOT (Avg.)	**28.1**	**35.2**	**31.0**
LEN + ROOT (Avg.)	21.8	31.2	25.6

Table 2: Unlabeled bracket scores in various settings. Avg. is the average score across languages.

question above. The main source of improvements by DEP is detections of constituents, but this constraint itself does not help to resolve some core dependency relationships, e.g., arcs from verbs to nouns. The constraint on root POS tags is thus orthogonal to this approach, and it may help to find such core dependencies. On the other hand, the dependency length bias is the most effective to find basic dependency relationships between POS tags while the resulting tree may involve implausible constituents. Thus the effect of the length bias seems somewhat overlapped with the root POS constraints, which may be the reason why they do not well collaborate with each other.

Bracket scores We verify the above intuition quantitatively. To this end, we convert both the predicted and gold dependency trees into the unlabeled bracket structures, and then compare them on the standard PARSEVAL metrics. This bracket tree is not binarized; for example, we extract (X a b (X c d)) from the tree $a \curvearrowleft b \curvearrowright c \curvearrowleft d$. Table 2 shows the results, and we can see that DEP always performs the best, showing that DEP leads to the models that find better constituent structures. Of particular note

	UAS	F1
DEP	48.1	**30.5**
LEN	48.5	27.9
DEP+LEN	**49.2**	27.0

Table 3: Average scores of DEP, LEN, and the combination.

is in Enlgish the bracket and dependency scores are only loosely correlated. In Table 1, UASs for FUNC, DEP, and LEN are 37.2, 39.8, and 52.1, respectively, though F1 of DEP is substantially higher. This suggests that DEP often finds more linguistically plausible structures even when the improvement in UAS is modest. We conjecture that this performance change between constraints essentially arise due to the nature of DEP, which eliminates center-embedding, i.e., implausible *constituent* structures, rather than dependency arcs.

Combining DEP and LEN These results suggest DEP and LEN capture different aspects of syntax. To furuther understand this difference, we now evaluate the models with *both* constraints. Table 3 shows the average scores across languages (without root constraints). Interestingly, the combination (DEP+LEN) performs the best in UAS while the worst in bracket F1. This indicates the ability of DEP to find good constituent boundaries is diminished by combining LEN. We feel the results are expected observing that center-embedded constructions are a special case of longer dependency constructions. In other words, LEN is a stronger constraint than DEP in that the structures penalized by DEP are only a subset of structures penalized by LEN. Thus when LEN and DEP are combined LEN overwhelms, and the advantage of DEP is weakened. This also suggests not penalizing all longer dependencies is important for learning accurate grammars. The improvement of UAS suggests there are also collaborative effects in some aspect.

7 Conclusion

We have shown that a syntactic constraint that eliminates center-embedding is helpful in dependency grammar induction. In particular, we found that our method facilitates to find linguistically correct constituent structures, and given an additional cue on dependency, the models compete with the system relying on a significant amount of prior linguistic knowledge. Future work includes applying our DEP constraint into other PCFG-based grammar induction tasks beyond dependency grammars. In particular, it would be fruitful to apply our idea into constituent structure induction for which, to our knowledge, there has been no successful PCFG-based learning algorithm. As discussed in de Marcken (1999) one reason for the failures of previous work is the lack of necessary syntactic biases, and our approach could be useful to alleviate this issue. Finally, though we have focused on unsupervised learning for simplicity, we believe our syntactic bias also leads to better learning in more practical scenarios, e.g., weakly supervised learning (Garrette et al., 2015).

Acknowledgements

We would like to thank John Pate for the help in preliminary work, as well as Taylor Berg-Kirkpatric for sharing his code. We are also grateful to Edson Miyamoto and Makoto Kanazawa for the valuable feedbacks. The first author was supported by JSPS KAKENHI Gran-in-Aid for JSPS Fellows (Grant Numbers 15J07986), and MOU Grant in National Institute of Informatics.

References

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

Yonatan Bisk and Julia Hockenmaier. 2013. An hdp model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876, Beijing,China, July.

C. de Marcken. 1999. On the unsupervised induction of phrase-structure grammars. In Susan Armstrong,

Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 191–208. Springer Netherlands.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 457–464, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Eisner. 2000. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341.

Dan Garrette, Chris Dyer, Jason Baldridge, and Noah Smith. 2015. Weakly-supervised grammar-informed bayesian ccg parser learning.

E. Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Kevin Gimpel and Noah A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581, Montréal, Canada, June. Association for Computational Linguistics.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Human Language*, pages 73–113. MIT Press, Cambridge, Mass.

John A Hawkins. 2014. *Cross-linguistic variatoin and efficiency*. Oxford University Press, jan.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May.

Mark Johnson. 1998. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In Christian Boitet and Pete Whitelock, editors, *COLING-ACL*, pages 619–623. Morgan Kaufmann Publishers / ACL.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic, June. Association for Computational Linguistics.

Fred Karlsson. 2007. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July.

David Mareček and Zdeněk Žabokrtský. 2012. Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Jeju Island, Korea, July. Association for Computational Linguistics.

George A. Miller and Noam Chomsky. 1963. Finitary models of language users. In D. Luce, editor, *Handbook of Mathematical Psychology*, pages 2–419. John Wiley & Sons.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA, October. Association for Computational Linguistics.

Hiroshi Noji and Yusuke Miyao. 2014. Left-corner transitions on dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2140–2150, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Hiroshi Noji. 2016. *Left-corner Methods for Syntactic Modeling with Universal Structural Constraints*. Ph.D. thesis, Graduate University for Advanced Studies, Tokyo, Japan, March.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *COLING*, pages 191–197.

D.J. Rosenkrantz and P.M. Lewis. 1970. Deterministic left corner parsing. In *Switching and Automata Theory, 1970., IEEE Conference Record of 11th Annual Symposium on*, pages 139–152, Oct.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 663–672, Portland, Oregon, USA, June. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pages 569–576, Sydney, July.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD, October.

Marten van Schijndel and William Schuler. 2013. An analysis of frequency- and memory-based processing costs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 95–105, Atlanta, Georgia, June. Association for Computational Linguistics.

Distinguishing Past, On-going, and Future Events: The EventStatus Corpus

Ruihong Huang
Texas A&M University
huangrh@cse.tamu.edu

Ignacio Cases
Stanford University
cases@stanford.edu

Dan Jurafsky
Stanford University
jurafsky@stanford.edu

Cleo Condoravdi
Stanford University
cleoc@stanford.edu

Ellen Riloff
University of Utah
riloff@cs.utah.edu

Abstract

Determining whether a major societal event has already happened, is still on-going, or may occur in the future is crucial for event prediction, timeline generation, and news summarization. We introduce a new task and a new corpus, *EventStatus*, which has 4500 English and Spanish articles about civil unrest events labeled as PAST, ON-GOING, or FUTURE. We show that the temporal status of these events is difficult to classify because local tense and aspect cues are often lacking, time expressions are insufficient, and the linguistic contexts have rich semantic compositionality. We explore two approaches for event status classification: (1) a feature-based SVM classifier augmented with a novel induced lexicon of *future-oriented* verbs, such as "threatened" and "planned", and (2) a convolutional neural net. Both types of classifiers improve event status recognition over a state-of-the-art TempEval model, and our analysis offers linguistic insights into the semantic compositionality challenges for this new task.

1 Introduction

When a major societal event is mentioned in the news (e.g., civil unrest, terrorism, natural disaster), it is important to understand whether the event has already happened (PAST), is currently happening (ON-GOING), or may happen in the future (FUTURE). We introduce a new task and corpus for studying the temporal/aspectual properties of major events. The **EventStatus corpus** consists of 4500 English and Spanish news articles about *civil unrest events*, such as protests, demonstrations, marches, and strikes, in which each event is annotated as PAST, ON-GOING, or FUTURE (sublabeled as PLANNED, ALERT or POSSIBLE). This task bridges event extraction research and temporal research in the tradition of TIMEBANK (Pustejovsky et al., 2003) and TempEval (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013). Previous corpora have begun this association: TIMEBANK, for example, includes temporal relations linking events with Document Creation Times (DCT). But the EventStatus task and corpus offers several new research directions.

First, major societal events are often discussed before they happen, or while they are still happening, because they have the potential to impact a large number of people. News outlets frequently report on impending natural disasters (e.g., hurricanes), anticipated disease outbreaks (e.g., Zika virus), threats of terrorism, and plans or warnings of potential civil unrest (e.g., strikes and protests). Traditional event extraction research has focused primarily on recognizing events that have already happened. Furthermore, the linguistic contexts of on-going and future events involve complex compositionality, and features like explicit time expressions are less useful. Our results demonstrate that a state-of-the-art TempEval system has difficulty identifying on-going and future events, mislabeling examples like these:

(1) *The metro workers' strike in Bucharest has entered the fifth day.* (On-Going)
(2) *BBC unions demand more talks amid threat of new strikes.* (Future)
(3) *Pro-reform groups have called for nationwide protests on polling day.* (Future)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 44–54,
Austin, Texas, November 1-5, 2016.

Second, we intentionally created the EventStatus corpus to concentrate on one particular event frame (class of events): civil unrest. In contrast, previous temporally annotated corpora focus on a wide variety of events. Focusing on one frame (semantic depth instead of breadth) makes this corpus analogous to domain-specific event extraction data sets, and therefore appropriate for evaluating rich tasks like event extraction and temporal question answering, which require more knowledge about event frames and schemata than might be represented in large broad corpora like TIMEBANK (UzZaman et al., 2012; Llorens et al., 2015).

Third, the EventStatus corpus focuses on specific instances of high-level events, in contrast to the low-level and often non-specific or generic events that dominate other temporal datasets.[1] Mentions of specific events are much more likely to be realized in non-finite form (as nouns or infinitives, such as "the strike" or "to protest") than randomly selected event keywords. In breadth-based corpora like the EventCorefBank (ECB) corpus (Bejan and Harabagiu, 2008), 34% of the events have non-finite realization; in TIMEBANK, 45% of the events have non-finite realization. By contrast, in a frame-based corpus like ACE2005 (ACE, 2005), 59% of the events have non-finite forms. In the EventStatus corpus, 80% of the events have non-finite forms. Whether this is due to differences in labeling or to intrinsic properties of these events, the result is that they are much harder to label because tense and aspect are less available than for events realized as finite verbs.

Fourth, the EventStatus data set is multilingual: we collected data from both English and Spanish texts, allowing us to compare events representing the same event frame across two languages that are known to differ in their typological properties for describing events (Talmy, 1985).

Using the new EventStatus corpus, we investigate two approaches for recognizing the temporal status of events. We create a SVM classifier that incorporates features drawn from prior TempEval work (Bethard, 2013; Chambers et al., 2014; Llorens et al., 2010) as well as a new automatically induced lexicon of 411 English and 348 Spanish "future-oriented" matrix verbs—verbs like "threaten" and "fear" whose complement clause or nominal direct object argument is likely to describe a future event. We show that the SVM outperforms a state-of-the-art TempEval system and that the induced lexicon further improves performance for both English and Spanish. We also introduce a Convolutional Neural Network (CNN) to detect the temporal status of events. Our analysis shows that it successfully models semantic compositionality for some challenging temporal contexts. The CNN model again improves performance in both English and Spanish, providing strong initial results for this new task and corpus.

2 The EventStatus Corpus

For major societal events, it can be very important to know whether the event has ended or if it is still in progress (e.g., are people still rioting in the streets?). And sometimes events are anticipated before they actually happen, such as labor strikes, marches and parades, social demonstrations, political events (e.g., debates and elections), and acts of war. The EventStatus corpus represents the *temporal status* of an event as one of five categories:

Past: An event that has started and has ended. There should be no reason to believe that it may still be in progress.
On-going: An event that has started and is still in progress or likely to resume[2] in the immediate future. There should be no reason to believe that it has ended.
Future Planned: An event that has not yet started, but a person or group has planned for or explicitly committed to an instance of the event in the future. There should be near certainty it will happen.
Future Alert: An event that has not yet started, but a person or group has been threatening, warning, or advocating for a future instance of the event.
Future Possible: An event that has not yet started, but the context suggests that its occurrence is a live possibility (e.g., it is anticipated, feared, hinted at, or is mentioned conditionally).

The three subtypes of future events are important

[1] For example in TIMEBANK almost half the annotated events (3720 of 7935) are hypothetical or generic, i.e., PERCEPTION, REPORTING, ASPECTUAL, I_ACTION, STATE or I_STATE rather than the specific OCCURRENCE.

[2] For example, demonstrators have gone home for the day but are expected to return in the morning.

	Past
[EN]	Today's *demonstration* ended without violence.
	An estimated 2,000 people *protested* against the government in Peru.
[SP]	Terminó la *manifestación* de los kurdos en la UNESCO de París.
	On-going
[EN]	Negotiations continue with no end in sight for the 2 week old *strike*.
	Yesterday's rallies have caused police to fear more today.
[SP]	Pacifistas latinoamericanos no cesan sus *protestas* contra guerra en Irak.
	Future Planned
[EN]	77 percent of German steelworkers voted to *strike* to raise their wages.
	Peace groups have already started organizing mass *protests* in Sydney.
[SP]	Miedo en la City en víspera de masivas *protestas* que la toman por blanco.
	Future Alert
[EN]	Farmers have threatened to hold *demonstrations* on Monday.
	Nurses are warning they intend to walkout if conditions don't improve.
[SP]	Indigenas hondureños amenazan con declararse en *huelga* de hambre.
	Future Possible
[EN]	Residents fear *riots* if the policeman who killed the boy is acquitted.
	The military is preparing for possible protests at the G8 summit.
[SP]	Policía Militar analiza la posibilidad de decretar una *huelga* nacional.

Table 1: Examples of event status categories for civil unrest events, showing two examples in English [EN] and one in Spanish [SP].

in marking not just temporal status but also what we might call predictive status. Events very likely to occur are distinguished from events whose occurrence depends on other contingencies (Future Planned vs. Alert/Possible). Warnings or mentions of a potential event by a likely actor are further distinguished from events whose occurrence is more open-ended (Future Alert vs. Possible). The status of future events is not due just to lexical semantics or local context but also other qualifiers in the sentence (e.g. "may"), the larger discourse context, and world knowledge. The annotation guidelines are formulated with that in mind. The categories for future events are not incompatible with one another but are meant to be informationally ordered (e.g. "future alert" implies "future possible"). Annotators are instructed to go for the strongest implication supported by the overall context. Table 1 presents examples of each category in news reports about civil unrest events, with the event keywords in *italics*.

2.1 EventStatus Annotations

The EventStatus dataset consists of English and Spanish news articles. We manually identified 6 English words[3] and 13 Spanish words[4] and phrases associated with civil unrest events, and added their morphological variants. We then randomly selected 2954 and 1491[5] news stories from the English Gigaword 5th Ed. (Parker et al., 2011) and Spanish Gigaword 3rd Ed. (Mendon et al., 2011) corpora, respectively, that contain at least one civil unrest phrase. Events of a specific type are very sparsely distributed in a large corpus like the Gigaword, so we used keyword matching just as a first pass to identify candidate event mentions.

[3]The English keywords are "protest", "strike", "march", "rally", "riot" and "occupy". These correspond to the most frequent words in the relevant frame in the Media Frames corpus (Card et al., 2015). Because "march" most commonly refers to the month, we removed the word itself and only kept its other morphological variations.

[4]Spanish keywords: "marchar", "protestar", "amotinar(se)", "manifestar(se)", "huelga", "manifestación", "disturbio", "motín", "ocupar * la calle", "tomar * la calle", "salir * las calles", "lanzarse a las calles", "cacerolas vacías", "cacerolazo", "cacerolada". Asterisks could be replaced by up to 4 words. The last three terms are common expressions for protest marches in many countries of Latin America and Spain.

[5]46 (out of 3000) and 9 (out of 1500) stories were removed due to keyword errors.

	Past	Ongoing	Future (Plan,Alert,Possible)	Multiple	Not Event
EN	1735	583	292 (197,48,47)	28	186
SP	1545	739	360 (279,61,30)	21	72

Table 2: Counts of Temporal Status Labels in EventStatus.

Because many keyword instances don't refer to a specific event, primarily due to lexical ambiguity and generic descriptions (e.g., *"Protests are often facilitated by ..."*), we used a two-stage annotation process. First, we extracted sentences containing at least one key phrase, and had three human annotators judge whether the sentence describes a specific civil unrest event. Next, for each sentence that mentions a specific event, the annotators assigned an event status to every civil unrest key phrase in that sentence. In both annotation phases, we asked the annotators to consider the context of the entire article.

In the first annotation phase, the average pairwise inter-annotator agreement (Cohen's κ) among the annotators was $\kappa = 0.84$ on the English data and 0.70 on the Spanish data. We then assigned the majority label among the three annotators to each sentence. In the English data, of the 5085 sentences with at least one key phrase, 2492 (49%) were judged to be about a specific civil unrest event. In the Spanish data, 3249 sentences contained at least one key phrase and 2466 (76%) described a specific event.

In the second phase, the annotators assigned one of the five temporal status categories listed in Section 2 to each event keyword in a relevant sentence. In addition, we provided a *Not Event* label.[6] Occasionally, a single instance of a keyword can refer to multiple events (e.g., *"Both last week's and today's protests..."*), so we permitted multiple labels to be assigned to an event phrase. However this happened for only 28 cases in English and 21 cases in Spanish.

The average pairwise inter-annotator agreement among the three human annotators for the temporal status labels was κ=.78 for English and κ=.80 for Spanish. We used the majority label among the three annotators as the gold status. In total, 2907 English and 2807 Spanish event phrases exist in the relevant sentences and were annotated. However there were 83 English cases (≈2.9%) and 70 Spanish cases (≈2.5%) where the labels among the three annotators were all different, so we discarded these cases. Table 2 shows the final distribution of labels in the EventStatus corpus. The EventStatus corpus[7] is available through the LDC.

[6] A sentence can contain multiple keyword instances. So even in a relevant sentence, some instances may not refer to a specific event.

2.2 Linguistic Properties of Event Mentions

Next, we investigated the linguistic properties of the event status categories, lumping together the 3 future subcategories. Table 3 shows the distribution of syntactic forms of the event mentions in two commonly used event datasets, ACE2005 (ACE, 2005) and EventCorefBank (Bejan and Harabagiu, 2008), and our new EventStatus corpus. In the introduction, we mentioned the high frequency of non-finite event expressions; Table 3 provides the evidence: non-finite forms (nouns and infinitives) constitute 59% in ACE2005, 34% in EventCorefBank, and a very high 80% of the events in the EventStatus dataset. The distribution is even more skewed for future events, which are 95% (English) and 96% (Spanish) realized by non-finite surface forms.

	Finite Verbs	Nouns	Inf. Verbs	Other
ACE Dataset				
	2201 (41)	2566 (48)	352 (7)	243 (5)
ECB Dataset				
	1151 (66)	488 (28)	77 (4)	25 (1)
EventStatus, English Section				
PA	331 (19)	1295 (75)	103 (6)	6 (0)
OG	58 (10)	476 (82)	29 (5)	20 (3)
FU	15 (5)	245 (84)	32 (11)	0 (0)
EventStatus, Spanish Section				
PA	315 (20)	1145 (74)	84 (5)	1 (0)
OG	41 (6)	685 (93)	12 (2)	1 (0)
FU	14 (4)	309 (86)	36 (10)	1 (0)

Table 3: Number and % (in parentheses) of event mentions by syntactic form. PA = Past; OG = On-going; FU = Future

2.3 Future Oriented Verbs

We observed that many future event mentions are preceded by a set of lexical (non-aux) verbs that we call *future oriented verbs*, such as "threatened" in (4) and "fear" in (5). These verbs project the events in the lower clause into the future.

[7] http://faculty.cse.tamu.edu/huangrh/EventStatus_corpus.html

(4) *They threatened to protest if Kmart does not acknowledge their request for a meeting.*

(5) *People fear renewed rioting during the coming days.*

Categories of future oriented verbs include mental activity ("anticipate", "expect"), affective ("fear", "worry"), planning ("plan", "prepare", "schedule"), threatening ("threaten", "advocate", "warn"), and inchoative verbs ("start", "initiate", and "launch"). We found that these categories correlate with the predictive status of the events they embed. We drew on these insights to induce a lexicon of future oriented verbs.

We harvested matrix verbs whose complement unambiguously describes a future event using two heuristics. One heuristic looks for examples with a tense conflict between the matrix verb and its complement: a matrix verb in the past tense (like "planned" below) whose complement event is an infinitive verb or deverbal noun modified by a future time expression (like "tomorrow" or "next week"), hence in the future (e.g., "strike" below): [8]

(6) *The union planned to strike next week.*

Future events are often marked by conditional clauses, so the second heuristic considers an event to be future if it was post-modified by a conditional clause (beginning with "if" or "unless"):

(7) *The union threatened to strike if their appeal was rejected.*

Finally, to increase precision, we only harvested a verb as future-oriented if it functioned as a matrix both in sentences with an embedded future time expression and in sentences with a conditional clause.

Future Oriented Verb Categories: We ran the algorithm on the English and Spanish Gigaword corpora (Parker et al., 2011; Mendon et al., 2011), obtaining 411 English verbs and 348 Spanish verbs. To better understand the structure of the learned lexicon, we mapped each English verb to Framenet (Baker et al., 1998); 86% (355) of the English verbs occurred in Framenet, in 306 unique frames. We clustered these into 102 frames[9] and grouped the Spanish verbs following English Framenet, identifying 67 categories. (Some learned verbs, such as "poise" , "slate" , "compel" and "hesitate", had a clear future orientation but didn't exist in Framenet.) Table 4 shows examples of learned verbs for English and their categories.

Commitment: threaten, vow, promise, pledge, commit, declare, claim, volunteer, anticipate
Coming_to_be: enter, emerge, plunge, kick, mount reach, edge, soar, promote, increase, climb, double
Purpose: plan, intend, project, aim, object, target
Permitting: allow, permit, approve, subpoena
Experiencer_subj: fear, scare, hate
Waiting: expect, wait
Scheduling: arrange, schedule
Deciding: decide, opt, elect, pick, select, settle
Request: ask, urge, order, encourage, demand, appeal, request, summon, implore, advise, invite
Evoking: raise, press, back, recall, pressure, force, rush, pull, drag, respond

Table 4: Examples from Future Oriented Verb Lexicon

In the next sections we propose two classifiers, an SVM classifier using standard TempEval features plus our new future-oriented lexicon, and a Convolutional Neural Net, as a pilot exploration of what features and architecture work well for the EventStatus task. For these studies we combine the Future Planned, Future Alert and Future Possible categories into a single Future event status because we first wanted to establish how well classifiers can detect the primary temporal distinctions between Past vs. Ongoing vs. Future. The future subcategories are, of course, relatively smaller and we expect that the most effective approach will be to design a classifier that sits on top of the primary classifier to further subcategorize the Future instances. We leave the task of subcategorizing future events for later work.

[8] For English, we extract events linked by the "xcomp" dependency using the Stanford dependency parser (Marneffe et al., 2006), with a future time expression attached to the second event with the "tmod" relation. For Spanish, we consider two events related if they are at most 5 words apart, and the second event is modified by a time expression, at most 5 words apart.

[9] By merging frames that share frame elements (e.g., "Purpose" and "Project" share the frame element "plan")

3 SVM Event Status Model

Our first classifier is a linear SVM classifier.[10] We trained three binary classifiers (one per class) using one-vs.-rest, and label an event mention with the class that assigned the highest score to the mention. We used features inspired by prior TempEval work and by the previous analysis, including words, tense and aspect features, time expressions, and the new future-oriented verb lexicon. We also experimented with other features used by TempEval systems (including bigrams, POS tags, and two-hop dependency features), but they did not improve performance.[11]

Bag-Of-Words Features: For bag-of-words unigram features we used a window size of 7 (7 left and 7 right) for the English data and 6 for the Spanish data; this size was optimized on the tuning sets.

Tense, Aspect and Time Expressions: Because these features are known to be the most important for relating events to document creation time (Bethard, 2013; Llorens et al., 2010), we used TIPSem (Llorens et al., 2010) to generate the tense and aspect of events and find time expressions in both languages. TIPSem infers the tense and aspect of nominal and infinitival event mentions using heuristics without relying on syntactic dependencies. For the English data set, we also generated syntactic dependencies using Stanford CoreNLP (Marneffe et al., 2006) and applied several rules to create additional tense and aspect features based on the governing words of event mentions[12]. Time indication features are created by comparing document creation time to time expressions linked to an event mention detected by TIPSem. If TIPSem detects no linked time expressions for an event mention, we take the nearest time expression in the same sentence.

Governing Words: Governing words have been useful in prior work. Our version of the feature pairs the governing word of an event mention with the dependency relation in between. We used Stanford CoreNLP (Marneffe et al., 2006) to generate dependencies for the English data. For the Spanish data, we used Stanford CoreNLP to generate Part-of-Speech tags[13] and then applied the MaltParser (Nivre et al., 2004) to generate dependencies.

4 Convolutional Neural Network Model

Convolutional neural networks (CNNs) have been shown to be effective in modeling natural language semantics (Collobert et al., 2011). We were especially keen to find out whether the convolution operations of CNNs can model the semantic compositionality needed to detect temporal-aspectual status. For our experiments, we trained a simple CNN with one convolution layer followed by one max pooling layer (Kim, 2014; Collobert et al., 2011),

The convolution layer has 300 hidden units. In each unit, the same affine transformation is applied to every consecutive 5 words (a filter instance) in the input sequence of words. A different affine transformation is applied to each hidden unit. After each affine transformation, a Rectified Linear Units (ReLU) (Nair and Hinton, 2010) non-linearity is applied. For each hidden unit, the max pooling layer selects the maximum value from the pool of real values generated from each filter instance.

After the max pooling layer, a *softmax* classifier predicts probabilites for each of the three classes, Past, Ongoing and Future. To alleviate overfitting of the CNN model, we applied dropout (Hinton et al., 2012) on the convolution layer and the following pooling layer with a keeping rate of 0.5.

Our experiments used the 300-dimension English word2vec embeddings[14] trained on 100 billion words of Google News. We trained our own 300-dimension Spanish embeddings, running word2vec (Mikolov et al., 2013) over both Spanish Gigaword (Mendon et al., 2011)— tokenized using Stanford CoreNLP SpanishTokenizer (Manning et al., 2014)— and the pre-tokenized Spanish Wikipedia dump (Al-Rfou et al., 2013). The vectors were then tuned during backpropagation for our specific task.

[10] Trained using LIBSVM (Chang and Lin, 2011) with linear kernels (polynomial kernels yielded worse performance).

[11] Previous TempEval work reported that those additional features were useful when computing temporal relations between two events but not when relating an event to the Document Creation Time, for which tense, aspect, and time expression features were the most useful (Llorens et al., 2010; Bethard, 2013).

[12] We did not imitate this procedure for Spanish because the quality of our generated Spanish dependencies is poor.

[13] Stanford CoreNLP has no support for generating syntactic dependencies for Spanish.

[14] docs.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM.

Row	Method	PA	OG	FU	Macro	Micro
1	**TIPSem**	26/80/39	8/32/13	4/23/7	13/45/20	20/68/31
2	**TIPSem with transitivity**	75/76/75	14/22/17	4/21/7	31/40/35	55/67/61
3	**SVM with all features**	91/81/86	33/47/39	45/58/51	56/62/59	75/75/75
4	**SVM with BOW features only**	88/80/84	37/46/41	40/53/45	55/60/57	72/72/72
5	**+Tense/Aspect/Time**	89/81/85	40/50/44	42/52/46	57/61/59	73/73/73
6	**+Governing Word**	90/81/85	43/56/48	42/55/47	58/64/61	75/75/75
7	**+Future Oriented Lexicon**	90/82/86	44/56/49	48/62/54	61/66/63	76/76/76
8	**Convolutional Neural Net**	91/83/**87**	46/57/**51**	49/67/**57**	62/69/**65**	77/77/**77**

Table 6: Experimental Results on English Data. Each cell shows Recall/Precision/F-score.

Row	Method	PA	OG	FU	Macro	Micro
1	**TIPSem**	19/84/31	14/38/20	4/53/8	12/58/20	16/65/25
2	**TIPSem with transitivity**	69/70/70	40/35/37	12/62/20	40/56/47	54/59/56
3	**SVM with all features**	84/77/80	48/51/49	42/57/48	58/62/60	69/69/69
4	**SVM with BOW features only**	82/75/78	53/56/54	34/52/41	56/61/59	68/68/68
5	**+Tense/Aspect/Time**	82/77/79	55/57/56	45/61/52	61/65/63	70/70/70
6	**+Governing Word**	83/75/79	51/56/53	42/58/49	59/63/61	69/69/69
7	**+Future Oriented Lexicon**	82/77/79	55/57/56	47/63/**54**	61/65/63	70/70/70
8	**Convolutional Neural Net**	84/80/**82**	60/58/**59**	44/59/50	62/66/**64**	72/72/**72**

Table 7: Experimental Results on Spanish Data. Each cell shows Recall/Precision/F-score.

	PA	OG	FU
English	1385 (68%)	427 (21%)	233 (11%)
Spanish	1251 (59%)	589 (28%)	280 (13%)

Table 5: Label Distributions in the Test Set

5 Evaluations

For all subsequent evaluations, we use gold event mentions. We randomly sampled around 20% of the annotated documents as the parameter tuning set and used the rest as the test set. Rather than training once on a distinct training set, all our experiment results are based on 10-fold cross validation on the test set, (1191 Spanish documents, 2364 English documents; see Table 5 for the distribution of event mentions).

5.1 Comparing with a TempEval System

We begin with a baseline: applying a TempEval system to classify each event. Most of our features are already drawn from TempEval, but our goal was to see if an off-the-shelf system could be directly applied to our task. We chose TIPSem (Llorens et al., 2010), a CRF system trained on TimeBank that uses linguistic features, has achieved top performance in TempEval competitions for both English and Spanish (Verhagen et al., 2010), and can compute the relation of each event with the Document Creation Time. We applied TIPSem to our test set, mapping the DCT relations to our three event status classes[15].

Row 1 of Tables 6 and 7 shows TIPSem results. The columns show results for each category separately, as well as macro-average and micro-average results across the three categories. Each cell shows the Recall/Precision/F-score numbers. Since TIPSem linked relatively few event mentions to the DCT, we next leveraged the transitivity of temporal relations (UzZaman et al., 2012; Llorens et al., 2015), linking an event to a DCT if the temporal relation between another event in the same sentence and the DCT is transferable. For instance, if event A is AFTER its DCT, and event B is AFTER event A, then event B is also AFTER the DCT.[16] Row 2 shows the results of TIPSem with temporal transitivity.

Even augmented by transitivity, TIPSem fails to detect many Ongoing (**OG**) and Future (**FU**) events; most mislabeled OG and FU events were nominal. Confusion matrices (Table 8) show that most of the

[15]We used the obvious mappings from TIPSem relations: "BEFORE" to "PA", "AFTER" to "FU" , and "INCLUDES" (for English) and "OVERLAP" (for Spanish) to "OG".

[16]Some transitivity rules are ambiguous: if event A is AFTER DCT, event B INCLUDES event A, event B can be AFTER or INCLUDES DCT. We ran experiments and chose rules that improved performance the most for TipSem.

missed OG events were labeled as Past (PA) while FU events were commonly mislabeled as both PA and OG. Below are some examples of OG and FU events mislabeled as PA:

(8) Jego said Sunday on arriving in Guadeloupe that he would stay as long as it took to bring an end to the strike organised by the Collective against Extreme Exploitation (LKP). (OG)

(9) A massive protest planned for Kathmandu on Tuesday has been re-baptised a victory parade. (FU)

	Predicted (EN)			Predicted (SP)		
	PA	**OG**	**FU**	**PA**	**OG**	**FU**
Gold **PA**	718	96	15	653	231	6
Gold **OG**	156	35	11	196	160	10
Gold **FU**	72	30	7	78	72	26

Table 8: Confusion Matrices for TIPSem (with transitivity).

SVM Results Next, we compare TIPSem's results with our SVM classifier. An issue is that TIPSem identifies only 72% and 78% of the gold event mentions, for English and Spanish respectively[17]. To have a fair comparison, we applied the SVM to only the event mentions that TipSem recognized. Row 3 shows these results for the SVM classifier using its full feature set. The SVM outperforms TipSem on all three categories, for both languages, with the largest improvements on Future events.

Next, we ran ablation experiments with the SVM to evaluate the impact of different subsets of its features. For these experiments, we applied the SVM to all gold event mentions, thus Rows 1-3 of Tables 6 and 7 report on fewer event mentions than rows 4-8. Row 4 shows results using only bag-of-words features[18]. Row 5 shows results when additionally including the tense, aspect, and time features provided by TIPSem (Llorens et al., 2010). Unsurprisingly, in both languages[19] these features improve over just bag-of-word features.

Row 6 further adds governing word features. These improve English performance, especially for On-Going events. For Spanish, governing word features slightly decrease performance, likely due to the poor quality of the Spanish dependencies.

Row 7 adds the future oriented lexicon features[20]. For both English and Spanish, the future oriented lexicon increased overall performance, and (as expected) especially for Future events.

CNN Results Row 8 shows the results using CNN models. For English and Spanish, the same window (7 words for English, 6 words for Spanish) was used to compute bag-of-word features for SVMs as for training the CNN models. For English, the CNN model further increased recall and precision across all three classes. The CNN improved Spanish performance on both Past and On-going events, but the SVM outperformed the CNN for Future events when the future oriented lexicon features were included.

6 Analysis

To better understand whether the CNN model's strong performance was related to handling compositionality, we examined some English examples that were correctly recognized by the CNN model but mislabeled by the SVM classifier with bag-of-words features. The examples below (event mentions are in *italics*) suggest that the CNN may be capturing the compositional impact of local cues like "possibility" or "since":

(10) Raising the possibility of a *strike* on New Year's Eve, the president of New York City's largest union is calling for a 30 percent raise over three years. (FU)

(11) The lockout was announced in the wake of a go-slow and partial *strike* by the union since July 12 after management turned down its demand. (OG)

We also conducted an error analysis by randomly sampling and then analyzing 50 of the 473 errors by the CNN model. Many cases (26/50) are ambiguous from the sentence alone, requiring discourse information. The first example below is caused by the well-known "double access" ambiguity of the complement of a communication verb (Smith, 1978; Abusch, 1997; Giorgi, 2010).

(12) Chavez also said he discussed the *strike* with UN Secretary General Kofi Annan and told him the strike organizers were "terrorists." (OG)

[17] We were not able to decouple TipSem's event recognition component and force it to process all event mentions.

[18] Replacing each word feature with a word2vec embedding resulted in slightly worse performance.

[19] We always obtain even recall and precision for the micro average metric because we only apply classifiers to event mentions that refer to a civil unrest event.

[20] For Spanish, we removed the governing word features because of the poor quality of the Spanish dependencies.

(13) Students and teachers *protest* over education budget (PA)

In 9/50 cases, the contexts that imply temporal status are complex and fall out of our ± 7 word range, e.g.,:

(14) Protesters on Saturday also *occupied* two gymnastics halls near Gorleben which are to be used as accommodation for police. They were later forcibly dispersed by policemen. (PA)

The remaining 15/50 cases contain enough local cues to be solvable by humans, but both the CNN and SVM models nonetheless failed:

(15) Eastern leaders have grown weary of the *protest* movement led mostly by Aymara. (OG)

7 Related Work

Our work overlaps with two communities of tasks and corpora: the task of classifying temporal order between event mentions and Document Creation Time (DCT) in TempEval (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013), and the task of extracting events, associated with corpora such as ACE2005 (ACE, 2005) and the EventCorefBank (ECB) (Bejan and Harabagiu, 2008). By studying the events in a particular frame (civil unrest), but focusing on their temporal status, our work has the potential to draw these communities together. Most event extraction work (Freitag, 1998; Appelt et al., 1993; Ciravegna, 2001; Chieu and Ng, 2002; Riloff and Jones, 1999; Roth and Yih, 2001; Zelenko et al., 2003; Bunescu and Mooney, 2007) has focused on extracting event slots or frames for past events and assigning dates. The TempEval task of linking events to DCT has not focused on events that tend to have non-finite realizations, nor has it focused on subtypes of future events. Our work, including the corpus and the future-oriented verb lexicon, has the potential to benefit related tasks like generating event timelines from news articles (Allan et al., 2000; Yan et al., 2011) or social media sources (Li and Cardie, 2014; Ritter et al., 2012), or exploring the psychological implications of future oriented language (Nie et al., 2015; Schwartz et al., 2015).

8 Conclusions

We have proposed a new task of recognizing the past, on-going, or future temporal status of major events, introducing a new resource for studying events in two languages. Besides its importance for studying time and aspectuality, the EventStatus dataset offers a rich resource for any future investigation of information extraction from major societal events.

The strong performance of the convolutional net system suggests the power of latent representations to model temporal compositionality, and points to extensions of our work using deeper and more powerful networks.

Finally, our investigation of the role of context and semantic composition in conveying temporal information also has implications for our understanding of temporality and aspectuality and their linguistic expression. Many of the errors made by our CNN system are complex ambiguities, like the double access readings, that cannot be solved without information from the wider discourse context. Our work can thus also be seen as a call for the further use of rich discourse information in the computational study of temporal processing.

9 Acknowledgments

We want to thank the Stanford NLP group and especially Danqi Chen for valuable inputs, and Michael Zeleznik for helping us refine the categories and for masterfully orchestrating the annotation efforts. We also thank Luke Zettlemoyer and all our reviewers for providing useful comments. This work was partially supported by the National Science Foundation via NSF Award IIS-1514268, by the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040, and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, NSF, IARPA, DoI/NBC, or the U.S. Government.

References

Dorit Abusch. 1997. Sequence of tense and temporal *de re*. *Linguistics & Philosophy*, 20:1–50.

ACE. 2005. NIST ACE evaluation website. In *http://www.nist.gov/speech/tests/ace/2005*.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

J. Allan, V. Lavrenko, D. Malin, and R. Swan. 2000. Detections, Bounds, and Timelines: Umass and TDT-3. In *Proceedings of Topic Detection and Tracking Workshop*.

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. FASTUS: a Finite-state Processor for Information Extraction from Real-world Text. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

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

C. Bejan and S. Harabagiu. 2008. A Linguistic Resource for Discovering Event Structures and Resolving Event Coreference. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*.

S. Bethard. 2013. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Proceedings of Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

R. Bunescu and R. Mooney. 2007. Learning to Extract Relations from the Web using Minimal Supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Nathanael Chambers, Bill McDowell, Taylor Cassidy, and Steve Bethard. 2014. Dense event ordering with a multi-pass architecture. In *Transactions of the Association for Computational Linguistics (TACL)*.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

H.L. Chieu and H.T. Ng. 2002. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In *Proceedings of the 18th National Conference on Artificial Intelligence*.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, and P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. In *Journal of Machine Learning Research*.

Dayne Freitag. 1998. Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*.

Alessandra Giorgi. 2010. *About the speaker: towards a syntax of indexicality*. Oxford University Press, Oxford.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. In *arXiv preprint arXiv:1207.0580*.

Y. Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of 2014 the Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*.

J. Li and C. Cardie. 2014. Timeline Generation: Tracking Individuals on Twitter. In *Proceedings of the 23rd International Conference on World Wide Web*.

H. Llorens, E. Saquete, and B. Navarro. 2010. TIPSem (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

H. Llorens, N. Chambers, N. UzZaman, Mostafazadeh N., J. Allen, and J. Pustejovsky. 2015. Semeval-2015 Task 5: QA TempEval - Evaluating Temporal Information Understanding with Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 55–60.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth Conference on Language Resources and Evaluation (LREC-2006)*.

Angelo Mendon ca, Daniel Jaquette, David Graff, and Denise DiPersio. 2011. Spanish Gigaword Third Edition. In *Linguistic Data Consortium*.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*.

V. Nair and G. E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of 27th International Conference on Machine Learning*.

A. Nie, J. Shepard, J. Choi, B. Copley, and P. Wolff. 2015. Computational Exploration of the Linguistic Structures of Future-Oriented Expression: Classification and Categorization. In *Proceedings of the NAACL Student Research Workshop (NAACL-SRW'15)*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-Based Dependency Parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL)*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword. In *Linguistic Data Consortium*.

J. Pustejovsky, P. Hanks, R. Saur, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics*.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

A. Ritter, Mausam, O. Etzioni, and S. Clark. 2012. Open Domain Event Extraction from Twitter. In *The 18th ACM SIGKDD Knowledge Discovery and Data Mining Conference*.

D. Roth and W. Yih. 2001. Relational Learning via Propositional Algorithms: An Information Extraction Case Study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1257–1263, Seattle, WA, August.

A. Schwartz, G. Park, M. Sap, E. Weingarten, J. Eichstaedt, M. Kern, J. Berger, M. Seligman, and L. Ungar. 2015. Extracting Human Temporal Orientation in Facebook Language. In *Proceedings of the The 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*.

Carlota Smith. 1978. The syntax and interpretation of temporal expressions in English. *Linguistics & Philosophy*, 2:43–99.

Leonard Talmy. 1985. Lexicalization patterns: Semantic structure in lexical forms. In Timothy Shopen, editor, *Language Typology and Syntactic Description, Volume 3*. Cambridge University Press.

N. UzZaman, H. Llorens, and J. Allen. 2012. Evaluating Temporal Information Understanding with Temporal Question Answering. In *Proceedings of IEEE International Conference on Semantic Computing*.

N. UzZaman, H. Llorens, J. Allen, L. Derczynski, M. Verhagen, and J. Pustejovsky. 2013. SemEval-2013 task 1: TempEval-3 evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. 2007. SemEval-2007 task 15: TempEval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*.

M. Verhagen, R. Sauri, T. Caselli, and J. Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

R. Yan, L. Kong, C. Huang, X. Wan, X. Li, and Y. Zhang. 2011. Timeline Generation through Evolutionary Trans-temporal Summarization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3.

Nested Propositions in Open Information Extraction

Nikita Bhutani
Department of EECS
University of Michigan
Ann Arbor
nbhutani@umich.edu

H. V. Jagadish
Department of EECS
University of Michigan
Ann Arbor
jag@umich.edu

Dragomir Radev
Department of EECS
University of Michigan
Ann Arbor
radev@umich.edu

Abstract

The challenges of Machine Reading and Knowledge Extraction at a web scale require a system capable of extracting diverse information from large, heterogeneous corpora. The Open Information Extraction (OIE) paradigm aims at extracting assertions from large corpora without requiring a vocabulary or relation-specific training data. Most systems built on this paradigm extract binary relations from arbitrary sentences, ignoring the context under which the assertions are correct and complete. They lack the expressiveness needed to properly represent and extract complex assertions commonly found in the text. To address the lack of representation power, we propose NESTIE, which uses a nested representation to extract higher-order relations, and complex, interdependent assertions. Nesting the extracted propositions allows NESTIE to more accurately reflect the meaning of the original sentence. Our experimental study on real-world datasets suggests that NESTIE obtains comparable precision with better minimality and informativeness than existing approaches. NESTIE produces 1.7-1.8 times more minimal extractions and achieves 1.1-1.2 times higher informativeness than CLAUSIE.

1 Introduction

Syntactic analyses produced by syntactic parsers are a long way from representing the full meaning of the sentences parsed. In particular, they cannot support questions like "*Who did what to whom?*", "*Where did what happen?*". Owing to the large, heterogeneous corpora available at web scale, traditional approaches to information extraction (Brin, 1998; Agichtein and Gravano, 2000) fail to scale to the millions of relations found on the web. As a response, the paradigm of Open Information Extraction (OIE) (Banko et al., 2007) has seen a rise in interest as it eliminates the need for domain knowledge or relation-specific annotated data. OIE systems use a collection of patterns over the surface form or dependency tree of a sentence to extract propositions of the form `(arg1,rel,arg2)`.

However, state-of-the-art OIE systems, REVERB (Fader et al., 2011) and OLLIE (Schmitz et al., 2012) focus on extracting binary assertions and suffer from three key drawbacks. First, lack of expressivity of representation leads to significant information loss for higher-order relations and complex assertions. This results in incomplete, uninformative and incoherent prepositions. Consider Example 1 in Figure 1. Important contextual information is either ignored or is subsumed in over-specified argument and relation phrases. It is not possible to fix such nuances by post-processing the propositions. This affects downstream applications like Question Answering (Fader et al., 2014) which rely on correctness and completeness of the propositions.

Second, natural language frequently includes relations presented in a non-canonical form that cannot be captured by a small set of extraction patterns that only extract relation mediated by verbs or a subset of verbal patterns. Consider Example 2 in Figure 1 that asserts, "*Rozsa Hill is the third hill near the river*", "*Rozsa Hill is Rose Hill*" and "*Rozsa Hill lies north of Castle Hill*". A verb-mediated pattern would extract

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 55–64,
Austin, Texas, November 1-5, 2016.

1. After giving 5,000 people a second chance at life, doctors are celebrating the 25th anniversary of Britain's first heart transplant.	
R:	P1: (doctors, are celebrating the 25th anniversary of, Britain 's first heart transplant)
O:	P1: (doctors, are celebrating, the 25th anniversary of Britain's first heart transplant)
N:	P1: (doctors, are celebrating, the 25th anniversary of Britain's first heart transplant) P2: (doctors, giving, second chance at life) P3: (P1, after, P2)
2. Rozsa (Rose) Hill , the third hill near the river, lies north of Castle Hill.	
R:	P1: (the third hill, lies north of, Castle Hill)
O:	P1: (the third hill, lies north of, Castle Hill)
N:	P1: (Rozsa, lies, north of Castle Hill) P2: (Rozsa Hill, is, third hill near the river) P3: (Rozsa Hill, is, Rose)
3. "A senior official in Iraq said the body, which was found by U.S. military police, appeared to have been thrown from a vehicle."	
R:	P1: (Iraq, said, the body) P2: (the body, was found by, U.S. military police)
O:	P1: (A senior official in Iraq, said, the body which was found by U.S. military police)
N:	P1: (body, appeared to have been thrown, ∅) P2: (P1, from, vehicle) P3: (A senior official in Iraq, said, P2) P4: (U.S. military police, found, body)

Figure 1: Example propositions from OIE systems: REVERB (R), OLLIE (O) and NESTIE(N).

a triple, `(the third hill, lies north of, Castle Hill)` that is less informative than a triple, `(Rozsa, lies, north of Castle Hill)` which is not mediated by a verb in the original sentence. Furthermore, these propositions are not complete. Specifically, queries of the form '*What is the other name of Rozsa Hill?*", "*Where is Rozsa Hill located?*", "*Which is the third hill near the river?*" will either return no answer or return an uninformative answer with these propositions. Since information is encoded at various granularity levels, there is a need for a representation rich enough to express such complex relations and sentence constructions.

Third, OIE systems tend to extract propositions with long argument phrases that are not minimal and are difficult to disambiguate or aggregate for downstream applications. For instance, the argument phrase, *body which was found by U.S. military police*, is less likely to be useful than the argument phrase, *body* in Example 3 in Figure 1.

In this paper we present NESTIE, which overcomes these limitations by 1) expanding the proposition representation to nested expressions so additional contextual information can be captured, 2) expanding the syntactic scope of relation phrases to allow relations mediated by other syntactic entities like nouns, adjectives and nominal modifiers. NESTIE bootstraps a small set of extraction patterns that cover simple sentences and learns broad-coverage relation-independent patterns. We believe that it is possible to adapt OIE systems that extract verb-based relations to process assertions denoting events with many arguments, and learn other non-clausal relations found in the text. With weakly-supervised learning techniques, patterns encoding these relations can be learned from a limited amount of data containing sentence equivalence pairs.

This article is organized as follows. We provide background on OIE in Sec. 2 followed by an overview of our proposed solution in Sec. 3. We then discuss how the extraction patterns for nested representations are learned in Sec. 4. In Sec. 5, we compare NESTIE against alternative methods on two datasets: Wikipedia and News. In Sec. 6, we discuss related work on pattern-based information extraction.

2 Background

The key goal of OIE is to obtain a shallow semantic representation of the text in the form of tuples consisting of argument phrases and a phrase that expresses the relation between the arguments. The phrases are identified automatically using domain-independent syntactic and lexical constraints. Some OIE systems are:

TextRunner (Yates et al., 2007) **WOE** (Wu and Weld, 2010): They use a sequence-labeling graphical model on extractions labeled automatically using heuristics or distant supervision. Consequently, long-range dependencies, holistic and lexical aspects of relations tend to get ignored.

ReVerb (Fader et al., 2011): Trained with shallow syntactic features, REVERB uses a logistic regression classifier to extract relations that begin with a

verb and occur between argument phrases.

Ollie (Schmitz et al., 2012): Bootstrapping from REVERB extractions, OLLIE learns syntactic and lexical dependency parse-tree patterns for extraction. Some patterns reduce higher order relations to ReVerb-style relation phrases. Also, representation is extended optionally to capture contextual information about conditional truth and attribution for extractions.

ClausIE (Del Corro and Gemulla, 2013): Using linguistic knowledge and a small set of domain-independent lexica, CLAUSIE identifies and classifies clauses into clause types, and then generates extractions based on the clause type. It relies on a predefined set of rules on how to extract assertions instead of learning extraction patterns. Also, it doesn't capture the relations between the clauses.

There has been some work in open-domain information extraction to extract higher-order relations. KRAKEN (Akbik and Löser, 2012) uses a predefined set of rules based on dependency parse to identify fact phrases and argument heads within fact phrases. But unlike alternative approaches, it doesn't canonicalize the fact phrases. There is another body of work in natural language understanding that shares tasks with OIE. AMR parsing (Banarescu et al.,), semantic role labeling (SRL) (Toutanova et al., 2008; Punyakanok et al., 2008) and frame-semantic parsing (Das et al., 2014). In these tasks, verbs or nouns are analyzed to identify their arguments. The verb or noun is then mapped to a semantic frame and roles of each argument in the frame are identified. These techniques have gained interest with the advent of hand-constructed semantic resources like PropBank and FrameNet (Kingsbury and Palmer, 2002; Baker et al., 1998). Generally, the verb/noun and the semantically labeled arguments correspond to OIE propositions and, therefore, the two tasks are considered similar. Systems like SRL-IE (Christensen et al., 2010) explore if these techniques can be used for OIE. However, while OIE aims to identify the relation/predicate between a pair of arguments, frame-based techniques aim to identify arguments and their roles with respect to a predicate. Hence, the frames won't correspond to propositions when both the arguments cannot be identified for a binary relation or when the correct argument is buried in long argument phrases.

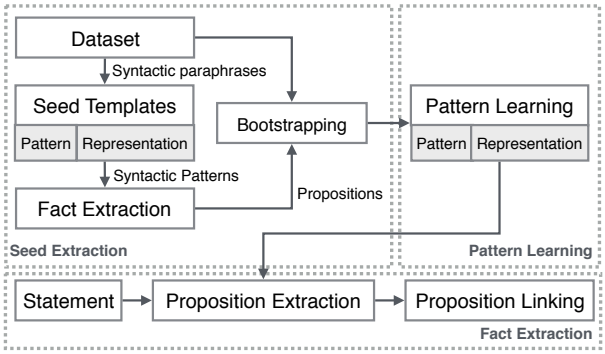


Figure 2: System Architecture of NESTIE.

3 Task Definition and NestIE Overview

Task: We focus on the task of OIE, where the system takes a natural language statement and extracts the supported assertions. This is achieved by using an extractor that uses nested representations to extract propositions and a linker that connects extracted propositions to capture context.

Proposition-based Extractor: We propose a framework to extend open-domain binary-relation extractors to extract n-ary and complex relations. As not all assertions can be expressed as `(arg1,rel,arg2)`, we learn syntactic patterns for relations that are expressed as nested templates like, `(arg1,rel,(arg2,rel2,arg3))`, `((arg1,rel,arg2),rel2,arg3)`.

Proposition Linking: In practice, it is infeasible to enumerate simple syntactic pattern templates that capture the entire meaning of a sentence. Also, increasing the complexity of templates would lead to sparsity issues while bootstrapping. We assume that there is a finite set of inter-proposition relations that can be captured using a small set of rules which take into account the structural properties of the propositions and syntactic dependencies between the relation phrases of the propositions.

System Evaluation: To compare NESTIE to other alternative methods, we conduct an experimental study on two real-world datasets: Wikipedia and News. Propositions from each system are evaluated for correctness, minimality, and informativeness.

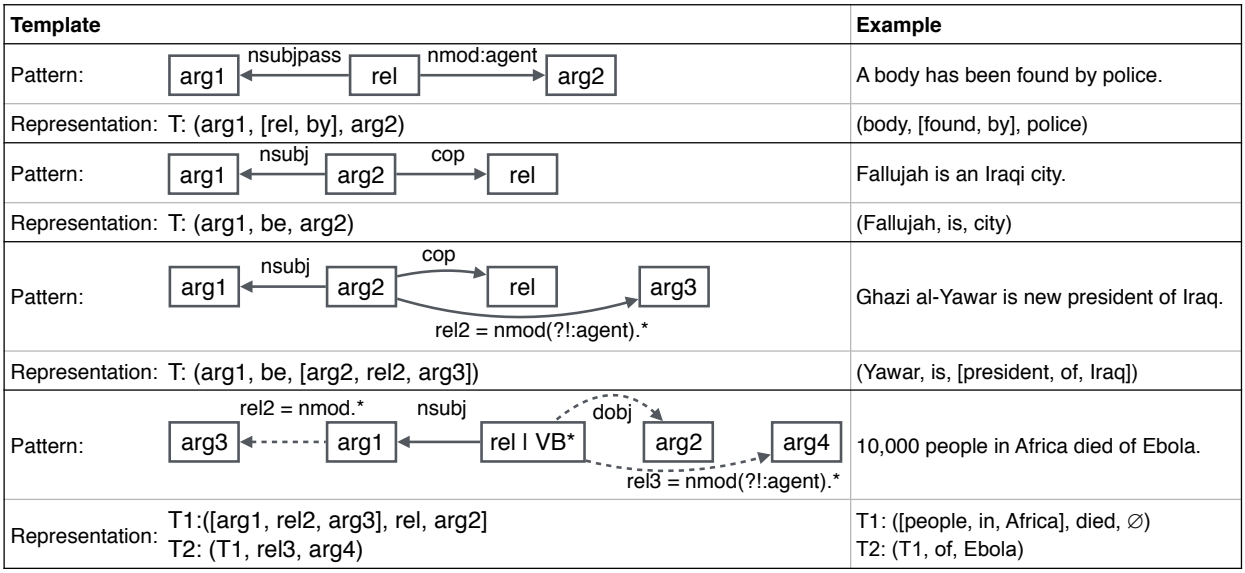


Figure 3: Seed templates and corresponding representation.

4 Proposition Extraction

Figure 2 illustrates the system architecture of NESTIE. First, a set of high-precision seed templates is used to extract propositions. A template maps a dependency parse-tree pattern to a triple representation such as `(arg1,rel,arg2)` for binary relations, or a nested triple representation such as `((arg1,rel,arg2),rel2,arg3)` for n-ary relations. Furthermore, an argument is allowed to be a sequence of words, "`arg2 rel2 arg3`" to capture its nominal modifiers. Then, using a RTE dataset that contains syntactic paraphrases, NESTIE learns equivalent parse-tree patterns for each template in the seed set. These patterns are used to extract propositions which are then linked.

4.1 Constructing Seed Set

We use a set of 13 hand-written templates. Each template maps an extraction pattern for a simple sentence to corresponding representation. A subset of these templates is shown in Figure 3. To create a seed set of propositions, we use the RTE dataset which is comprised of statements and their entailed hypotheses. We observed that most of the hypotheses were syntactic variants of the facts in their corresponding statements. These hypotheses were also short with a single, independent clause. These shared sentence constructions could be captured with a small set of templates. We iteratively create templates until at least one proposition could be extracted for each hypothesis. The propositions from the hypotheses form the set for bootstrapping.

For each seed proposition extracted from a hypothesis, the statement entailing the hypothesis contains all the content words of the proposition and expresses the same information as the proposition. However, there is a closed class of words, such as prepositions, a subset of adverbs, determiners, verbs etc. that does not modify the underlying meaning of the hypothesis or the statement and can be considered auxiliary. These were ignored while constructing the seed set.

Example 1 Consider a statement-hypothesis pair,
Statement: *Paul Bremer, the top U.S. civilian administrator in Iraq, and Iraq's new president, Ghazi al-Yawar, visited the northern Iraqi city of Kirkuk.*
Hypothesis: *Ghazi al-Yawar is the president of Iraq.*
The hypothesis is entailed in the statement. The seed templates extract propositions from the hypothesis: `(al-Yawar,is,president`, `(al-Yawar,is,president of Iraq)`, and `(al-Yawar,is president of,Iraq)`.

Bootstrapping is a popular technique to generate positive training data for information extraction (Collins and Singer, 1999; Hoffmann et al., 2011). We extend the bootstrapping techniques employed

in OLLIE and RENOUN, for n-ary and complex relations. First, instead of learning dependency parse-tree patterns connecting the heads of the argument phrases and the relation phrase connecting them, we learn the dependency parse-tree patterns connecting the heads of all argument and relation phrases in the template. This allows greater coverage of context for the propositions and prevents the arguments/relations from being over-specified and/or uninformative. Second, some of the relations in the representation are derived from the type of dependency, e.g. type of nominal modifier. As these relations are implicit, and might not be present in the paraphrase, they are ignored for learning. Intuitively, with such constraints, paraphrases "*Mary gave John a car*" and "*Mary gave a car to John*" can map to the same representation.

4.2 Extraction Pattern Learning

The biggest challenge in information extraction is the multitude of ways in which information can be expressed. Since it is not possible to enumerate all the different syntactic variations of an assertion, there is a need to learn general patterns that encode the various ways of expressing the assertion. In particular, we learn the various syntactic patterns that can encode the same information as the seed patterns and hence can be mapped to same representation.

NESTIE tries to learn the different ways in which the content words of a seed proposition from a hypothesis can be expressed in the statement that entails this hypothesis. We use the Stanford dependency parser (De Marneffe et al., 2006) to parse the statement and identify the path connecting the content words in the parse tree. If such a path exists, we retain the syntactic constraints on the nodes and edges in the path and ignore the surface forms of the nodes in the path. This helps generalize the learned patterns to unseen relations and arguments. NESTIE could learn 183 templates from the 13 seed templates. Figure 4 shows a subset of these patterns.

Example 2 Consider dependency parse-subtree of the statement and hypothesis from Example 1,
Statement: $Iraq \xrightarrow{\text{poss}} president \xrightarrow{\text{appos}} al-Yawar$
Hypothesis: $al-Yawar \xleftarrow{\text{nsubj}} president \xrightarrow{\text{of}} Iraq$
A seed extraction pattern maps the parse-tree of the hypothesis to the representation, `(arg1, be, arg2)`, returning proposition, `(al-Yawar,is,president of Iraq)`. With bootstrapping, the syntactic pattern from the statement is mapped to the same representation.

4.3 Pattern Matching

Once the extraction patterns are learned, we use these patterns to extract propositions from new unseen sentences. We first parse a new sentence and match the patterns against the parse tree. As the patterns only capture the heads of the arguments and relations, we expand the extracted propositions to increase the coverage of context of the arguments as in the original sentence.

Example 3 In the statement from Example 1, the extraction patterns capture the dependency path connecting the head words: `Iraq`, `administrator` and `Paul Bremer`. However, to capture the contextual information, further qualification of the argument node, `administrator`, is required.

Following this observation, we expand the arguments on `nmod`, `amod`, `compound`, `nummod`, `det`, `neg` edges. We expand the relations on `advmod`, `neg`, `aux`, `auxpass`, `cop`, `nmod` edges. Only the dependency edges not captured in the pattern are considered for expansion. Also, the order of words from the original sentence is retained in the argument phrases.

4.4 Proposition Linking

NESTIE uses a nested representation to capture the context of extracted propositions. The context could include condition, attribution, belief, order, reason and more. Since it is not possible to generate or learn patterns that can express these complex assertions as a whole, NESTIE links the various propositions from the previous step to generate nested propositions that are complete and closer in meaning to the original statement.

The proposition linking module is based on the assumption that the inter-proposition relation can be inferred from the dependency parse of the sentence from which propositions were extracted. Some of the rules employed to link the propositions are:

- The relation of proposition P1 has a relationship to the relation of proposition P2.

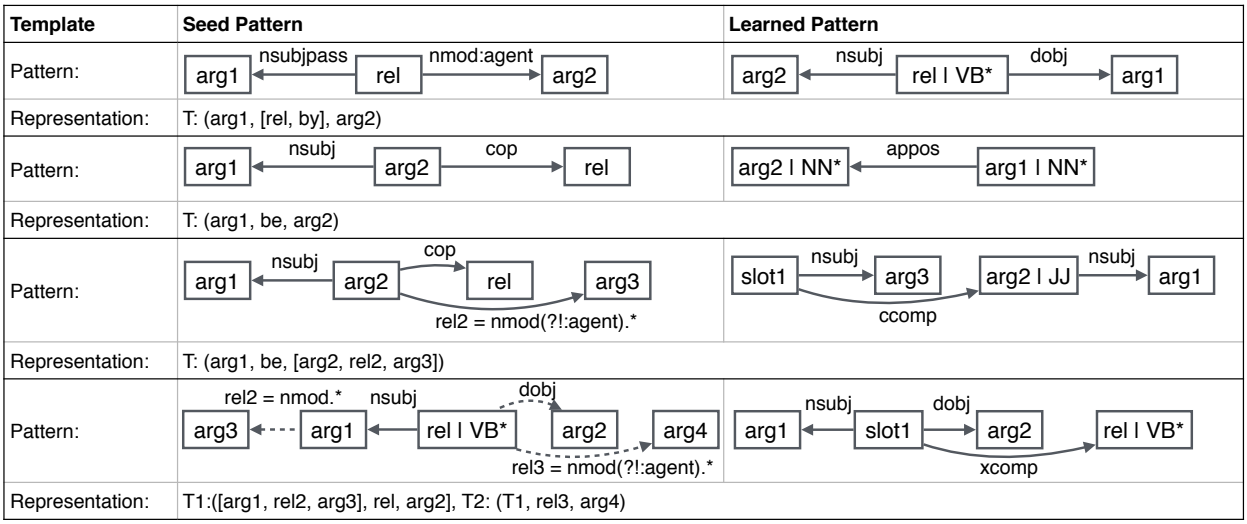


Figure 4: Syntactic Patterns learned using bootstrapping.

Consider the statement, "The accident happened after the chief guest had left the event." and propositions, P1: `(accident, happen, ϕ)` and P2: `(chief guest, had left, event)`. Using dependency edge, `nmod:after`, the linking returns `(P1,after,P2)`.

- Proposition P1 is argument in proposition P2.

Consider the statement, "A senior official said the body appeared to have been thrown from a vehicle." and propositions, P1: `(body,appeared to have been thrown from,vehicle)` and P2: `(senior official,said,ϕ)`. The linking updates P2 to `(senior official,said,P1)`.

- An inner nested proposition is replaced with a more descriptive alternative proposition.

We use dependency parse patterns to link propositions. We find correspondences between: a `ccomp` edge and a clausal complement, an `advcl` edge and a conditional, a `nmod` edge and a relation modifier. For clausal complements, a null argument in the source proposition is updated with the target proposition. For conditionals and nominal modifiers, a new proposition is created with the source and target propositions as arguments. The relation of the new proposition is derived from the target of the `mark` edge from the relation head of target proposition.

4.5 Comparison with Ollie

NESTIE uses an approach similar to OLLIE and WOE to learn dependency parse based syntactic patterns. However, there are significant differences. First, OLLIE and WOE rely on extractions from REVERB and Wikipedia info-boxes respectively for bootstrapping. Most of these relations are binary. On the contrary, our algorithm is based on high-confidence seed templates that are more expressive and hence learn patterns expressing different ways in which the proposition as a whole can be expressed. Though the arguments in OLLIE can be expanded to include the n-ary arguments, NESTIE encodes them in the seed templates and learns different ways of expressing these arguments. Also, similar to OLLIE, NESTIE can extract propositions that are not just mediated by verbs.

5 Experiments

We conducted an experimental study to compare NESTIE to other state-of-the-art extractors. We found that it achieves higher informativeness and produces more correct and minimal propositions than other extractors.

5.1 Experimental Setup

We used two datasets released by (Del Corro and Gemulla, 2013) in our experiments: 200 random sentences from Wikipedia, and 200 random sentences from New York Times (NYT). We compared

Dataset		Reverb	Ollie	ClausIE	NestIE
NYT dataset	Avg. Informativeness	1.437/5	2.09/5	2.32/5	**2.762/5**
	Correct	187/275 (0.680)	359/529 (0.678)	527/882 (0.597)	469/914 (0.513)
	Minimal (among correct)	161/187 (0.861)	238/359 (0.663)	199/527 (0.377)	**355/469 (0.757)**
Wikipedia dataset	Avg. Informativeness	1.63/5	2.267/5	2.432/5	**2.602/5**
	Correct	194/258 (0.752)	336/582 (0.577)	453/769 (0.589)	415/827 (0.501)
	Minimal (among correct)	171/194 (0.881)	256/336 (0.761)	214/453 (0.472)	**362/415 (0.872)**

Figure 5: Informativeness and number of correct and minimal extractions as fraction of total extractions.

NESTIE against three OIE systems: REVERB, OLLIE and CLAUSIE. Since the source code for each of the extractors was available, we independently ran the extractors on the two datasets. Next, to make the extractions comparable, we configured the extractors to generate triple propositions. REVERB and CLAUSIE extractions were available as triples by default. OLLIE extends its triple proposition representation. So, we generated an additional extraction for each of the possible extensions of a proposition. NESTIE uses a nested representation. So, we simply extracted the innermost proposition in a nested representation as a triple and allowed the subject and the object in the outer proposition to contain a *reference* to the inner triple. By preserving references the context of a proposition is retained while allowing for queries at various granularity levels.

We manually labeled the extractions obtained from all extractors to 1) maintain consistency, 2) additionally, assess if extracted triples were informative and minimal. Some extractors use heuristics to identify arguments and/or relation phrase boundaries, which leads to over-specific arguments that render the extractions unusable for other downstream applications. To assess the usability of extractions, we evaluated them for minimality (Bast and Haussmann, 2013). Furthermore, the goal of our system is to extract as many propositions as possible and lose as little information as possible. We measure this as *informativeness* of the set of the extractions for a sentence. Since computing informativeness as a percentage of text contained in at least one extraction could be biased towards long extractions, we used an explicit rating scale to measure informativeness.

Two CS graduate student labeled each extraction for correctness (0 or 1) and minimality (0 or 1). For each sentence, they label the set of extractions for informativeness (0-5). An extraction is marked correct if it is asserted in the text and correctly captures the contextual information. An extraction is considered minimal if the arguments are not over-specified i.e. they don't subsume another extraction or have conjunctions or are excessively long. Lastly, they rank the set of extractions on a scale of 0-5 (0 for bad, 5 for good) based on the coverage of information in the original sentence. The agreement between labelers was measured in terms of Cohens Kappa.

5.2 Comparative Results

The results of our experimental study are summarized in Figure 5 which shows the number of correct and minimal extractions, as well as the total number of extractions for each extractor and dataset. For each dataset, we also report the macro-average of informativeness reported by the labelers. We found moderate inter-annotator agreement: 0.59 on correctness and 0.53 on minimality for both the datasets. Each extractor also includes a confidence score for the propositions. But since each extractor has its unique method to find confidence, we compare the precision over all the extractions instead of a subset of high-confidence extractions.

NESTIE produced many more extractions, and more informative extractions than other systems. There appears to be a trade-off between informativeness and correctness (which are akin to recall and precision, respectively). CLAUSEIE is the system with results closer to NESTIE than other systems. However, the nested representation and proposition linking used by NESTIE produce substantially more (1.7-1.8 times more) minimal extractions than CLAUSEIE, which generates propositions from the constituents of the clause. Learning non-verb medi-

ated extraction patterns and proposition linking also increase the syntactic scope of relation expressions and context. This is also reflected in the average informativeness score of the extractions. NESTIE achieves 1.1-1.9 times higher informativeness score than the other systems.

We believe that nested representation directly improves minimality, independent of other aspects of extractor design. To explore this idea, we conducted experiments on OLLIE, which does not expand the context of the arguments heuristically unlike other extractors. Of the extractions labeled correct but not minimal by the annotators on the Wikipedia dataset, we identified extractions that satisfy one of: 1) has an argument for which there is an equivalent extraction (nested extractions), 2) shares the same subject with another extraction whose relation phrase contains the relation and object of this extraction (n-ary extractions), 3) has an object with conjunction. Any such extractions can be made minimal and informative with a nested representation. 73.75% of the non-minimal correct extractions met at least one of these conditions, so by a post-processing step, we could raise the minimality score of OLLIE by 17.65%, from 76.1% to 93.75%.

5.3 Error Analysis of NestIE

We did a preliminary analysis of the errors made by NESTIE. We found that in most of the cases (about 33%-35%), extraction errors were due to incorrect dependency parsing. This is not surprising as NESTIE relies heavily on the parser for learning extraction patterns and linking propositions. An incorrect parse affects NESTIE more than other systems which are not focused on extracting finer grained information and can trade-off minimality for correctness. An incorrect parse not only affects the pattern matching but also proposition linking which either fails to link two propositions or produces an incorrect proposition.

Example 4 Consider the statement, "*A day after strong winds stirred up the Hauraki Gulf and broke the mast of Team New Zealand, a lack of wind caused Race 5 of the America's Cup to be abandoned today.*". The statement entails following assertions:

A1: "strong winds stirred up the Hauraki Gulf"

A2: "strong winds broke the mast of Team New Zealand"

A3: "a lack of wind caused Race 5 of the America's Cup to be abandoned"

A1 and A2 are parsed correctly. A3 is parsed incorrectly with `Race 5` as object of the verb `caused`. Some extractors either don't capture A3 or return an over-specified extraction, *(a lack of wind, caused, Race 5 of the America 's Cup to be abandoned today)*. Such an extraction is correct but not minimal.

To maintain minimality, NESTIE aims to extract propositions, P1: *(Race 5 of the America 's Cup, be abandoned, ϕ)* and P2: *(a lack of wind, caused, P1)*. However, it fails because of parser errors. It extracts incorrect proposition, P3: *(a lack of wind, caused, Race 5)* corresponding to A3 and links it to propositions for A1 and A2. Linking an incorrect proposition generates more incorrect propositions which hurt the system performance.

However, we hope this problem can be alleviated to some extent as parsers become more robust. Another approach could be to use clause segmentation to first identify clause boundaries and then use NESTIE on reduced clauses. As the problem becomes more severe for longer sentences, we wish to explore clause processing for complex sentences in future.

Another source of errors was under-specified propositions. Since our nested representation allows null arguments for intransitive verb phrases and for linking propositions, failure to find an argument/proposition results in an under-specified extraction. We found that 27% of the errors were because of null arguments. However, by ignoring extractions with null arguments we found that precision increases by only 4%-6% (on Wikipedia). This explains that many of the extractions with empty arguments were correct, and need special handling. Other sources of errors were: aggressive generalization of an extraction pattern to unseen relations (24%), unidentified dependency types while parsing long, complex sentences (21%), and errors in expanding the scope of arguments and linking extractions (20%).

6 Related Work

As OIE has gained popularity to extract propositions from large corpora of unstructured text, the problem of the extractions being uninformative and incomplete has surfaced. A recent paper (Bast and Haussmann, 2014) pointed out that a significant fraction of the extracted propositions is not informative. A simple inference algorithm was proposed that uses generic rules for each semantic class of predicate to derive new triples from extracted triples. Though it improved the informativeness of extracted triples, it did not alleviate the problem of lost context in complex sentences. We, therefore, create our own extractions.

Some recent works (Bast and Haussmann, 2013; Angeli et al., 2015) have tried to address the problem of long and uninformative extractions in open-domain information extraction by finding short entailment or clusters of semantically related constituents from a longer utterance. These clusters are reduced to triples using schema mapping to known relation types or using a set of hand-crafted rules. NESTIE shares similar objectives but uses bootstrapping to learn extraction patterns.

Bootstrapping and pattern learning has a long history in traditional information extraction. Systems like DIPRE (Brin, 1998), SNOWBALL (Agichtein and Gravano, 2000), NELL (Mitchell, 2010), and OLLIE bootstrap based on seed instances of a relation and then learn patterns for extraction. We follow a similar bootstrapping algorithm to learn extraction patterns for n-ary and nested propositions.

Using a nested representation to express complex and n-ary assertions has been studied in closed-domain or ontology-aided information extraction. Yago (Suchanek et al., 2008) and (Nakashole and Mitchell, 2015) extend binary relations to capture temporal, geospatial and prepositional context information. We study such a representation for open-domain information extraction.

7 Conclusions

We presented NESTIE, a novel open information extractor that uses nested representation for expressing complex propositions and inter-propositional relations. It extends the bootstrapping techniques of previous approaches to learn syntactic extraction patterns for the nested representation. This allows it to obtain higher informativeness and minimality scores for extractions at comparable precision. It produces 1.7-1.8 times more minimal extractions and achieves 1.1-1.2 times higher informativeness than CLAUSEIE. Thus far, we have tested our bootstrap learning and proposition linking approaches only on a small dataset. We believe that its performance will improve with larger datasets. NESTIE can be seen as a step towards a system that has a greater awareness of the context of each extraction and provides informative extractions to downstream applications.

Acknowledgments

This research was supported in part by NSF grants IIS 1250880 and IIS 1017296.

References

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

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *Proceedings of the AKBC-WEKEX*, pages 52–56.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of ACL*, pages 26–31.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation (amr) 1.0 specification.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *IEEE-ICSC 2013*, pages 154–159.

Hannah Bast and Elmar Haussmann. 2014. More informative open information extraction via simple inference. In *Advances in information retrieval*, pages 585–590. Springer.

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

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL-HLT 2010*, pages 52–60.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110. Citeseer.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the IW3C2*, pages 355–366.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of ACM-SIGKDD*, pages 1156–1165. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*.

Tom Mitchell. 2010. Never-ending learning. Technical report, DTIC Document, Carnegie Mellon University.

Ndapandula Nakashole and Tom M Mitchell. 2015. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of ACL*, pages 365–375.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of EMNLP-CoNLL 2012*, pages 523–534.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the ACL*, pages 118–127.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of NAACL-HLT: Demonstrations*, pages 25–26.

A Position Encoding Convolutional Neural Network Based on Dependency Tree for Relation Classification

Yunlun Yang Yunhai Tong* Shulei Ma Zhi-Hong Deng*
{incomparable-lun, yhtong, mashulei, zhdeng}@pku.edu.cn
Key Laboratory of Machine Perception (Ministry of Education),
School of Electronics Engineering and Computer Science, Peking University,
Beijing 100871, China

Abstract

With the renaissance of neural network in recent years, relation classification has again become a research hotspot in natural language processing, and leveraging parse trees is a common and effective method of tackling this problem. In this work, we offer a new perspective on utilizing syntactic information of dependency parse tree and present a position encoding convolutional neural network (PECNN) based on dependency parse tree for relation classification. First, tree-based position features are proposed to encode the relative positions of words in dependency trees and help enhance the word representations. Then, based on a redefinition of "context", we design two kinds of tree-based convolution kernels for capturing the semantic and structural information provided by dependency trees. Finally, the features extracted by convolution module are fed to a classifier for labelling the semantic relations. Experiments on the benchmark dataset show that PECNN outperforms state-of-the-art approaches. We also compare the effect of different position features and visualize the influence of tree-based position feature by tracing back the convolution process.

1 Introduction

Relation classification focuses on classifying the semantic relations between pairs of marked entities in given sentences (Hendrickx et al., 2010). It is a fundamental task which can serve as a pre-existing system and provide prior knowledge for information extraction, natural language understanding, information retrieval, etc. However, automatic recognition of semantic relation is challenging. Traditional feature based approaches rely heavily on the quantity and quality of hand-crafted features and lexical resources, and it is time-consuming to select an optimal subset of relevant features in order to maximize performance. Though kernel based methods get rid of the feature selection process, they need elaborately designed kernels and are also computationally expensive.

Recently, with the renaissance of neural network, deep learning techniques have been adopted to provide end-to-end solutions for many classic NLP tasks, such as sentence modeling (Socher, 2014; Kim, 2014) and machine translation (Cho et al., 2014). Recursive Neural Network (RNN) (Socher et al., 2012) and Convolutional Neural Network (CNN) (Zeng et al., 2014) have proven powerful in relation classification. In contrast to traditional approaches, neural network based methods own the ability of automatic feature learning and alleviate the problem of severe dependence on human-designed features and kernels.

However, previous researches (Socher et al., 2012) imply that some features exploited by traditional methods are still informative and can help enhance the performance of neural network in relation classification. One simple but effective approach is to concatenate lexical level features to features extracted by neural network and directly pass the combined vector to classifier. In this way, Socher et al. (2012), Liu et al. (2015) achieve better performances when considering some external features produced

*Corresponding authors

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 65–74,
Austin, Texas, November 1-5, 2016.

by existing NLP tools. Another more sophisticated method adjusts the structure of neural network according to the parse trees of input sentences. The results of (Li et al., 2015) empirically suggest syntactic structures from recursive models might offer useful power in relation classification. Besides relation classification, parse tree also gives neural network a big boost in other NLP tasks (Mou et al., 2015; Tai et al., 2015).

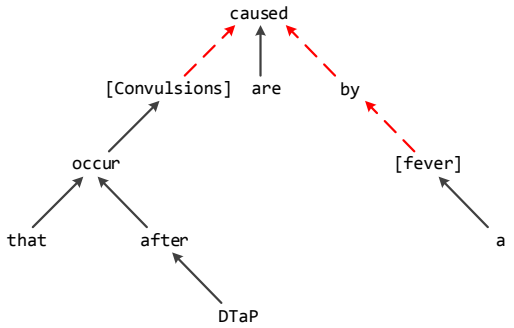


Figure 1: A dependency tree example. Words in square brackets are marked entities. The red dashed-line arrows indicate the path between two entities.

Dependency parse tree is valuable in relation classification task. According to our observation, dependency tree usually shortens the distances between pairs of marked entities and helps trim off redundant words, when comparing with plain text. For example, in the sentence shown in Figure 1, two marked entities span the whole sentence, which brings much noise to the recognition of their relation. By contrast, in the dependency tree corresponding to the sentence, the path between two marked entities comprises only four words and extracts a key phrase "caused by" that clearly implies the relation of entities. This property of dependency tree is ubiquitous and consistent with the Shortest Path Hypothesis which is accepted by previous studies (Bunescu and Mooney, 2005; Xu et al., 2015a; Xu et al., 2015b).

To better utilize the powerful neural network and make the best of the abundant linguistic knowledge provided by parse tree, we propose a position encoding convolutional neural network (PECNN) based on dependency parse tree for relation classification. In our model, to sufficiently benefit from the important property of dependency tree, we introduce the position feature and modify it in the context of parse tree. Tree-based position features encode the relative positions between each word and marked entities in a dependency tree, and help the network pay more attention to the key phrases in sentences. Moreover, with a redefinition of "context", we design two kinds of tree-based convolution kernels for capturing the structural information and salient features of sentences.

To sum up, our contributions are:

1) We propose a novel convolutional neural network with tree-based convolution kernels for relation classification.

2) We confirm the feasibility of transferring the position feature from plain text to dependency tree, and compare the performances of different position features by experiments.

3) Experimental results on the benchmark dataset show that our proposed method outperforms the state-of-the-art approaches. To make the mechanism of our model clear, we also visualize the influence of tree-based position feature on relation classification task.

2 Related Work

Recent studies usually present the task of relation classification in a supervised perspective, and traditional supervised approaches can be divided into feature based methods and kernel methods.

Feature based methods focus on extracting and selecting relevant feature for relation classification. Kambhatla (2004) leverages lexical, syntactic and semantic features, and feeds them to a maximum entropy model. Hendrickx et al. (2010) show that the winner of SemEval-2010 Task 8 used the most types of features and resources, among all participants. Nevertheless, it is difficult to find an optimal feature set, since traversing all combinations of features is time-consuming for feature based methods.

To remedy the problem of feature selection mentioned above, kernel methods represent the input data by computing the structural commonness between sentences, based on carefully designed kernels. Mooney and Bunescu (2005) split sentences into subsequences and compute the similarities using the proposed subsequence kernel. Bunescu and

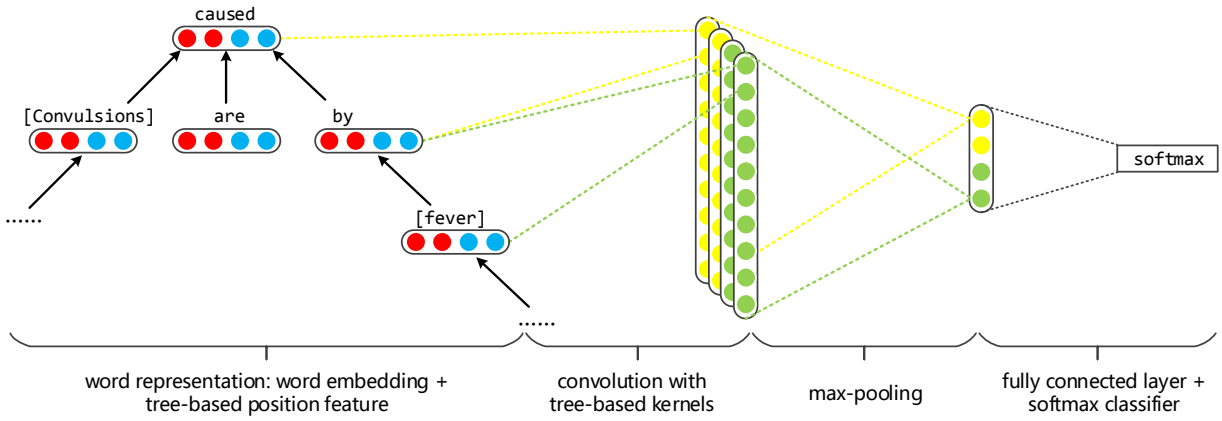


Figure 2: The framework of PECNN. The red and blue circles represent the word embeddings and tree-based position features of words. The yellow and green circles stand for the feature maps extracted by two kinds of convolution kernels respectively.

Mooney (2005) propose a dependency tree kernel and extract information from the Shortest Dependency Path (SDP) between marked entities. Since kernel methods require similarity computation between input samples, they are relatively computationally expensive when facing large-scale datasets.

Nowadays, deep neural network based approaches have become the main solutions to relation classification. Among them, some handle this task by modifying sentence modeling methods. Socher et al. (2012) build RNN on constituency trees of sentences, and apply the model to relation recognition task. Zeng et al. (2014) propose the use of position feature for improving the performance of CNN in relation classification. dos Santos et al. (2015) diminish the impact of noisy class by using a pairwise ranking loss function based CNN. Meanwhile, inspired by the ideas of traditional methods, some recent researches concentrate on mining information from the SDP. Xu et al. (2015b) use a multichannel LSTM network to model the SDP in given sentences. Liu et al. (2015) reserve the subtrees attached to the SDP and propose an augmented SDP based CNN. Neural network based methods offer the advantage of automatic feature learning and also scale well with large amounts of data.

3 Proposed Model

Given a sentence s with two marked entities $e1$ and $e2$, we aim to identify the semantic relation between $e1$ and $e2$ in relation classification. As the set of target relations is predefined, this task can be formulated as a multi-class classification problem. In this section, we detailedly describe our proposed model designed for this problem.

3.1 Framework

The schematic illustration of the framework is shown in Figure 2.

First, the dependency tree of a sentence is generated by the Stanford Parser (Klein and Manning, 2003). For each word in the tree, its word embedding and tree-based position features are concatenated as its representation. The position feature of a word is determined by the relative position between the word and marked entities in the dependency tree.

Next, with tree-based kernels, convolution operations are conducted on each node of the dependency tree. Compared with plain text, dependency tree could provide a word with more meaningful context, thus making tree-based kernel more effective. After convolution, we apply max-pooling over the extracted feature maps to capture the most important features.

At last, the output of max-pooling layer, i.e. the feature vector of input sentence, is fed to a softmax classifier for labelling the semantic relation of entities in each sentence.

3.2 Word Represetation

The representation of a word is composed of two parts: word embedding and tree-based position feature.

3.2.1 Word Embedding

Distributed representation of words in a vector space help learning algorithms to achieve better performance in NLP tasks (Mikolov et al., 2013). Such representation is usually called word embedding in recent works. High-quality word embedding is able to capture precise syntactic and semantic information by training unsupervisedly on large-scale corpora.

In our model, we initialize the word embeddings by pretraining them on a large corpus and further fine-tune them in training phase.

3.2.2 Tree-based Position Feature

Position Feature (PF) is first proposed by (Collobert et al., 2011) for semantic role labeling. (Zeng et al., 2014) exploit position feature as a substitute for traditional structure features in relation classification. The main idea of position feature is to map each discrete distance to a real-valued vector. It is similar to word embedding, except that words are replaced by discrete distances. For instance, let us examine again the sentence shown in Figure 1,

$$[\text{Convulsions}]_{e1} \text{ that occur after DTaP are caused by a } [\text{fever}]_{e2}.$$

the relative distances of *caused* to *Convulsions* and *fever* are respectively 6 and -3. Each relative distance is further mapped to a d_{pf} (a hyperparameter) dimensional vector, which is randomly initialized. Supposing $\boldsymbol{pf}_6$ and $\boldsymbol{pf}_{-3}$ are the corresponding vectors of distance 6 and -3, the position feature of *caused* is given by concatenating these two vectors $[\boldsymbol{pf}_6, \boldsymbol{pf}_{-3}]$.

Position feature on plain text proves to be informative (dos Santos et al., 2015), while it may suffer from several problems. According to our case study, adverbs or unrelated entities that appear between two entities in a sentence could significantly affect the performance of position feature, as these words only change the relative distance to entities without providing any more useful information for relation classification. Similarly, position feature often fails to handle sentences in which marked entities are too far from each other.

On the other hand, dependency tree focuses on the action and agents in a sentence (Socher et al., 2014), which is valuable for relation classification. As we have mentioned above, dependency tree is able to shorten the distances between pairs of marked entities and help trim off redundant words. Therefore, it is straightforward and reasonable to transfer the position feature from plain text to dependency tree.

We propose two kinds of Tree-based Position Feature which we refer as TPF1 and TPF2.

TPF1 encodes the relative distances of current word to marked entities in dependency trees. The "relative distance" here refers to the length of the shortest dependency path between current word and target entity. The sign of the distance is used to distinguish whether current word is a descendant of target entity. After calculating the relative distances of words in the tree, we can get their TPF1 by mapping relative distances to corresponding vectors, which is the same as the PF in plain text.

To more precisely describe the position of a word, TPF2 incorporates more information given by dependency tree. TPF2 represents the relative positions between current word and marked entities by encoding their shortest paths. For a word and an entity, the shortest path between them can be separated by their lowest common ancestor, and the lengths of the two sub-paths are sufficient for encoding the shortest path and the relative position between the word and the entity. As a result, we formally represent the relative position using a 2-tuple, in which two elements are the lengths of the two separated sub-paths respectively. Thereafter, each unique relative position is mapped to a real-valued vector.

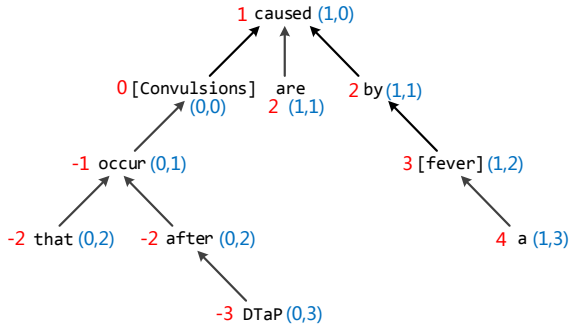


Figure 3: Example of Tree-based Position Features. The red numbers are relative distances in TPF1. The blue 2-tuples are relative positions in TPF2.

For example, in Figure 3, the path between *Convulsions* and *by* is *Convulsions*→ *caused*←*by*. In

TPF1, the relative distance of *by* to *Convulsions* is 2, the length of this path. In TPF2, the lowest common ancestor *caused* splits the path into two subpaths of length 1, so the relative position between *Convulsions* and *by* is $(1, 1)$ (encoded in 2-tuple). More examples of the tree-based position features are shown in Figure 3.

TPF1 and TPF2 both offer good strategies for encoding word position in dependency tree. TPF2 is more fine-grained than TPF1 and TPF1 is a simplified version of TPF2.

In our model, for each word in dependency trees, its word embedding and tree-based position feature are concatenated to form its representation, which is subsequently fed to the convolutional layer.

3.3 Convolution Methods

In the classic CNN architecture of (Collobert et al., 2011) and its variants (Kim, 2014), a convolution window covers a word and its context, i.e. its neighboring words. Thus convolution only captures local features around each word. Words that are not in a same window will not interact, even if they are syntactically related.

Compared with plain text, dependency tree could provide a word with more meaningful context. In a dependency tree, words are connected if they are in some dependency relationship. To capitalize on these syntactic information, we regard the parent and children of a word (i.e. nodes neighboring this word) as its new context. Changing the definition of "context" leads to modification of convolution kernel. To implement this idea, we design two kinds of tree-based kernels (Kernel-1 and Kernel-2), and apply them to sentences in dependency tree form.

Formally, for a word x in the dependency tree, let p be its parent and $c_1, \cdots, c_n$ be its n children. Their vector representation are respectively denoted by $\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{c}_1, \cdots, \boldsymbol{c}_n \in \mathbb{R}^d$. The convolution process of Kernel-1 is formulated as

$$\begin{aligned} \boldsymbol{z}^1_{xi} =& g(W^1_x \cdot \boldsymbol{x} + W^1_p \cdot \boldsymbol{p} + W^1_c \cdot \boldsymbol{c}_i) \\ & for \quad i = 1, \cdots, n \end{aligned} \tag{1}$$

where $\boldsymbol{z}^1_{xi} \in \mathbb{R}^{n_1}$ and n_1 is the number of Kernel-1, and $W^1_x, W^1_p, W^1_c \in \mathbb{R}^{n_1 \times d}$ are weight parameters corresponding to the word, its parent and children respectively. g is the non-linear activation function. For leaf nodes which have no child, i.e. $n = 0$, we assign each of them a child of which the vector representation is $\mathbf{0}$. For the root node, $\boldsymbol{p}$ is set to be $\mathbf{0}$.

Similarly, the output of Kernel-2 is given by

$$\begin{aligned} \boldsymbol{z}^2_{xi} =& g(W^2_x \cdot \boldsymbol{x} + W^2_{lc} \cdot \boldsymbol{c}_i + W^2_{rc} \cdot \boldsymbol{c}_{i+1}) \\ & for \quad i = 1, \cdots, n-1 \end{aligned} \tag{2}$$

where $\boldsymbol{z}^2_{xi} \in \mathbb{R}^{n_2}$ and n_2 is the number of Kernel-2, and $W^2_x, W^2_{lc}, W^2_{rc} \in \mathbb{R}^{n_2 \times d}$ are weight parameters associated with the word and its two neighboring children. If $n \leq 1$, we simply add one or two $\mathbf{0}$ children, just like the zero padding strategy.

Kernel-1 aims at extracting features from words of multiple levels in dependency tree, while Kernel-2 focuses on mining the semantic information between words which share the same parent. Kernel-1 and Kernel-2 both consider 3 words at a time because the experimental results of previous researches (Zeng et al., 2014; dos Santos et al., 2015) suggest that trigram features are relatively more useful in relation classification. And it is also straightforward to extend these kernels to a larger size and apply them to other tasks.

After convolution with tree-based kernels, we apply a global max-pooling operation over extracted features by taking the maximum value in each dimension, which is formulated as

$$\boldsymbol{h}^1 = \operatorname*{elemax}_{x,i} \boldsymbol{z}^1_{xi} \tag{3}$$

$$\boldsymbol{h}^2 = \operatorname*{elemax}_{x,i} \boldsymbol{z}^2_{xi} \tag{4}$$

where $\boldsymbol{h}^1 \in \mathbb{R}^{n_1}$, $\boldsymbol{h}^2 \in \mathbb{R}^{n_2}$, and elemax is the operation which gives the element-wise maximum of all input vectors. As a consequence, the output of convolution process is $[\boldsymbol{h}^1, \boldsymbol{h}^2]$, the combination of features extracted by two kinds of kernels.

3.4 Output and Training Objective

After convolution, the extracted feature is further passed to a fully connected softmax layer whose output is the probability distribution over all types of relations.

Since we treat the relation classification task as a multi-class classification problem, the training objective is the cross-entropy error. For regularization, we apply dropout (Srivastava et al., 2014) to the feature vector extracted by convolution and penalize the fully connected layer with l_2 regularizer as well.

Some other dependency tree based methods like (Liu et al., 2015), (Xu et al., 2015a) and (Xu et al., 2015b), all focus on using different kinds of neural networks to model the shortest dependency path (SDP) between entities. By contrast, PECNN extracts features from the whole dependency tree, so that the information out of SDP will be taken into consideration as well. The empirical results of (dos Santos et al., 2015) suggest that when position features exist, modeling the full sentence yields a better performance than only using the subsentence between entities. With the help of tree-based position feature, our model is capable of evaluating the importance of different parts of dependency trees and tends to pay relatively more attention to SDP.

Some methods enhancing their performances by proposing dataset-specific strategies. dos Santos et al. (2015) treat the class *Other* as a special class and omit its embedding. Xu et al. (2015a) take the relation dimensionality into account and introduce a negative sampling strategy to double the number of training samples, which can be regarded as data augmentation. These strategies do not conflict with our model, but we decide not to integrate them into our methods as we aim to offer a general and effective feature extraction model for relation classification.

4 Experiments

4.1 Dataset and Evaluation Metric

To evaluate our method, we conduct experiments on the SemEval2010 Task 8 dataset which is a widely used benchmark for relation classification. The dataset contains $8,000$ training sentences and $2,717$ test sentences. In each sentence, two entities are marked as target entities.

The predefined target relations include 9 directed relations and an undirected *Other* class. The 9 directed relations are *Cause-Effect, Component-Whole*, *Content-Container*, *Entity- Destination*, *Entity-Origin*, *Instrument-Agency*, *Member-Collection*, *Message-Topic* and *Product-Producer*. "Directed" here means, for example, *Cause-Effect*(e_1, e_2) and *Cause-Effect*(e_2, e_1) are two different relations. In another word, the directionality of relation also matters. And sentences that do not belong to any directed relation are labelled as *Other*. Therefore, relation classification on this dataset is a 19-class classification problem.

Following previous studies, we use the official evaluation metric, macro-averaged F1-score with directionality taken into account and the *Other* class ignored.

4.2 Training Details

Since there is no official validation set, 10% of the training sentences are taken out for hyperparameter tuning and early stopping.

When converting sentences to dependency trees, we note that some prepositions such as "by", "in" and "of", might be important clues to relation classification. To reserve these valuable information, we use the Stanford Parser **without** the *collapsed* option.

In the dataset, there are some entities consisting of multiple words, which make the calculation of relative position ambiguous. To solve this problem, we take the last word as the representation of an entity, as the last word is usually the predominant word.

For word embeddings, we initialize them using the 300-dimensional `word2vec` vectors[1]. The vectors are trained on 100 billion words from Google News. Words not present in the `word2vec` vectors are initialized by sampling each dimension from a uniform distribution (Kim, 2014). Tree-based position features are 50-dimensional and initialized randomly. Therefore the representation of each word has dimensionality of 400.

We use ReLU as the activation function. The number of convolution kernels is 500 for each kind, $1,000$ in total. The dropout rate is 0.5, and the coefficient of l_2 penalty of fully connected layer is set to 10^{-6}. These parameters are selected through grid search on validation set. The network is trained with the Adadelta update rule (Zeiler, 2012). The network is implemented with Theano (Theano Development Team, 2016).

[1] https://code.google.com/p/word2vec/

Classifier	Features	$F1$
Without External Lexical Features		
MVRNN	word embedding, constituency tree	79.1
CNN	word embedding, position feature	78.9
CR-CNN	word embedding	82.8*
	word embedding, position feature	84.1*
depLCNN	word embedding, dependency tree	81.9
	word embedding, dependency tree	84.0°
SDP-LSTM	word embedding, dependency tree	83.0
PECNN	word embedding, dependency tree, tree-based position feature	**84.0**
With External Lexical Features		
SVM	POS, prefixes, morphological, WordNet, dependency parse Levin classes, PropBankFrameNet, NomLex-Plus, Google n-gram paraphrases, TextRunner	82.2
MVRNN	word embedding, constituency tree, POS, NER, WordNet	82.4
CNN	word embedding, position feature, WordNet	82.7
DepNN	word embedding, dependency tree, WordNet	83.0
	word embedding, dependency tree, NER	83.6
depLCNN	word embedding, dependency tree, WordNet	83.7
	word embedding, dependency tree, WordNet	85.6°
SDP-LSTM	word embedding, dependency tree, POS embedding WordNet embedding, grammar relation embedding	83.7
PECNN	word embedding, dependency tree, tree-based position feature, POS NER, WordNet	**84.6**

Table 1: Comparison of different relation classification models. The symbol * indicates the results with special treatment of the class *Other*. The symbol ° indicates the results with data augmentation strategy.

4.3 Results

The performances of our proposed model and other state-of-the-art methods are shown in Table 1.

First, we compare PECNN with the following baselines when no external lexical feature is used.

Socher et al. (2012) assign a vector and a matrix to each word for the purpose of semantic composition, and build recursive neural network along constituency tree (MVRNN). It is noteworthy that this work is the first one who confirms the feasibility of applying neural network to relation classification.

Following the ideas of (Collobert et al., 2011), Zeng et al. (2014) first solve relation classification using convolutional neural network (CNN). The position feature introduced by them proves effective. dos Santos et al. (2015) build a similar CNN called CR-CNN but replace the objective function with a pairwise ranking loss. By treating the noisy class *Other* as a **special** class, this method achieves an $F1$ of 84.1. The $F1$ score is 82.7 if no special treatment is applied.

The rest two baselines focus on modeling the Shortest Dependency Paths (SDP) between marked entities. Xu et al. (2015a)) (depLCNN) integrate the relation directionality into CNN and achieve an $F1$ of 84.0 with a data augmentation strategy called negative sampling. Without such data augmentation, their $F1$ score is 81.9. Xu et al. (2015b) (SDP-LSTM) represent heterogeneous features as embeddings and propose a multichannel LSTM based recurrent neural network for picking up information along the SDP. Their $F1$ score is 83.0 when only word embedding is used as the word representation.

Without considering any external lexical feature and dataset-specific strategy, our model achieve an $F1$ of 84.0, suggesting that tree-based position features and kernels are effective. Comparing with the CNN based on plain text, our model benefits from dependency tree based network and obtain a notable

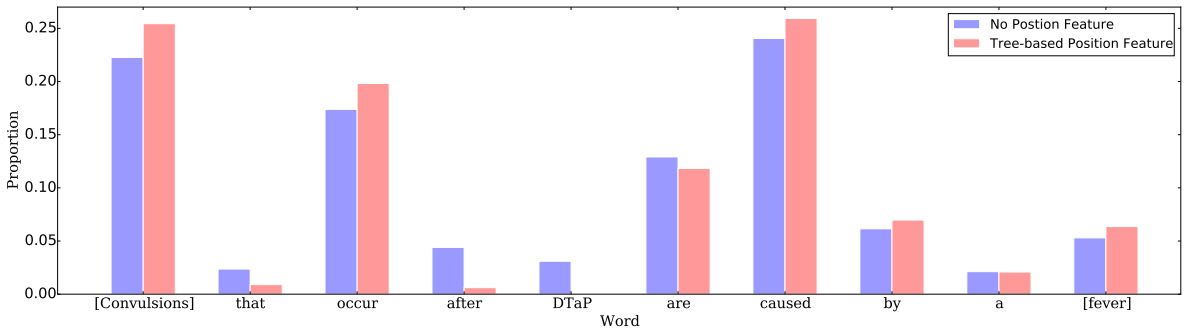


Figure 4: Visualization of the effect of tree-based position feature. The proportions of words change with the use of tree-based position feature.

improvement.

When external lexical features are available, we take two more baselines into account. The first one (SVM) is a typical example of traditional feature-based methods which rely largely on hand-crafted features. Benefitting from various features and resources, this method won the SemEval 2010 Task 8 by a large margin (Hendrickx et al., 2010). Liu et al. (2015) (DepNN) reserve the subtrees attached to the SDP and propose an augmented SDP based CNN.

Most of these baselines concatenate external lexical features to features extracted by neural network and directly pass the combined vector to classifier. SDP-LSTM represents lexical features as embeddings and enhances its word representation. For fair comparison, we add three features (POS tags, NER tags and WordNet hypernyms of marked entities) to the vector extracted by our model and retrain the network. Thus, our model achieves an $F1$ of 84.6 and outperforms all existing baselines in a fair condition where no data augmentation strategy is adopted. The enhancement we gain from external features is less, comparing with other baselines. This implies that our model is able to mine useful features from limited resources, even no extra information is available.

4.4 Effect of Different Position Features

Position Feature	$F1$
plain text PF	83.21
TPF1	83.99
TPF2	83.90

Table 2: Comparison of different position features.

Table 2 summarizes the performances of proposed model when different position features are exploited. To concentrate on studying the effect of position features, we do not involve lexical features in this section. As the table shows, the position feature on plain text is still effective in our model and we accredit its satisfactory result to the dependency information and tree-based kernels. The $F1$ scores of tree-based position features are higher since they are "specially designed" for our model.

Contrary to our expectation, the more fine-grained TPF2 does not yield a better performance than TPF1, and two kinds of TPF give fairly close results. One possible reason is that the influence of a more elaborated definition of relative position is minimal. As most sentences in this dataset are of short length and their dependency trees are not so complicated, replacing TPF1 with TPF2 usually brings little new structural information and thus results in a similar $F1$ score.

However, though the performances of different position features are close, tree-based position feature is an essential part of our model. The $F1$ score is severely reduced to 75.22 when we remove the tree-based position feature in PECNN.

4.5 Effect of Tree-based Position Feature

For shallow CNN in NLP, visualization offers clear and convincing explanations for the mechanism of neural networks (dos Santos and Gatti, 2014; Mou et al., 2015). Moreover, it is easy to implement.

Note that in the max-pooling step, for each kernel, we select the feature which has the largest value. This feature corresponds to 3 words in the convolu-

tion step, and we regard them as the most relevant words extracted by this kernel, with respect to the sentence . Since there are $1,000$ kernels in total, we count $3,000$ words ($\mathbf{0}$ will be ignored) and calculate the proportion of each different word. Intuitively, the more important a word is in this task, the larger its proportion will be.

In Figure 4, we compare the proportions of words in the example sentence when tree-based position feature (TPF) is used and not. As we can see, the proportions of two entities, *Convulsions* and *fever*, and the phrase *caused by* all increase visibly with the presence of TPF, suggesting that TPF is effective in helping the neural network pay more attention to the crucial words and phrases in a sentence. The word *occur* is also picked up by our model since it is an important candidate clue to relation classification. Meanwhile, the influence of irrelevant entity *DTaP* is remarkably diminished as expected.

5 Conclusion

This work presents a dependency parse tree based convolutional neural network for relation classification. We propose tree-based position features to encode the relative positions of words in a dependency tree. Meanwhile, tree-based convolution kernels are designed to gather semantic and syntactic information in dependency trees. Experimental results prove the effectiveness of our model. Comparing with plain text based CNN, our proposed kernels and position features boost the performance of network by utilizing dependency trees in a new perspective.

6 Acknowledgements

This work is partially supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015403) and the National Natural Science Foundation of China (Grant No. 61170091). We would also like to thank the anonymous reviewers for their helpful comments.

References

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

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

Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 626–634.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38. Association for Computational Linguistics.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computa-*

tional Linguistics-Volume 1, pages 423–430. Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2304–2314.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 285–290.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2315–2325.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Stanford University.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

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

Learning to Recognize Discontiguous Entities

Aldrian Obaja Muis **Wei Lu**
Singapore University of Technology and Design
{aldrian_muis,luwei}@sutd.edu.sg

Abstract

This paper focuses on the study of recognizing discontiguous entities. Motivated by a previous work, we propose to use a novel hypergraph representation to jointly encode discontiguous entities of unbounded length, which can overlap with one another. To compare with existing approaches, we first formally introduce the notion of *model ambiguity*, which defines the difficulty level of interpreting the outputs of a model, and then formally analyze the theoretical advantages of our model over previous existing approaches based on linear-chain CRFs. Our empirical results also show that our model is able to achieve significantly better results when evaluated on standard data with many discontiguous entities.

1 Introduction

Building effective automatic named entity recognition (NER) systems that is capable of extracting useful semantic shallow information from texts has been one of the most important tasks in the field of natural language processing. An effective NER system can typically play an important role in certain downstream NLP tasks such as relation extraction, event extraction, and knowledge base construction (Hasegawa et al., 2004; Al-Rfou and Skiena, 2012).

Most traditional NER systems are capable of extracting entities[1] as short spans of texts. Two basic assumptions are typically made when extracting entities: 1) entities do not overlap with one another, and 2) each entity consists of a contiguous sequence of words. These assumptions allow the task to be modeled as a sequence labeling task, for which many existing models are readily available, such as linear-chain CRFs (McCallum and Li, 2003).

EGD showed [***hiatal hernia***]$_1$ and vertical [***laceration***]$_2$ in distal [***esophagus***]$_2$ with [***blood in*** [***stomach***]$_4$]$_3$ and overlying [***lac***]$_4$.

Figure 1: Discontiguous entities in a medical domain. Words annotated with the same index are part of the same entity. Note that entity 3 and entity 4 overlap with one another.

While the above two assumptions are valid for most cases, they are not always true. For example, in the entity *University of New Hampshire* of type *ORG* there exists another entity *New Hampshire* of type *LOC*. This violates the first assumption above, yet it is crucial to extract both entities for subsequent tasks such as relation extraction and knowledge base construction. Researchers therefore have proposed to tackle the above issues in NER using more sophisticated models (Finkel and Manning, 2009; Lu and Roth, 2015). Such efforts still largely rely on the second assumption.

Unfortunately, the second assumption is also not always true in practice. There are also cases where the entities are composed of multiple discontiguous sequences of words, such as in disorder mention recognition in clinical texts (Pradhan et al., 2014b), where the entities (disorder mentions in this case) may be discontiguous. Consider the example shown in Figure 1. In this example there are four enti-

[1]Or sometimes *mentions* are considered, which can be named, nominal or pronominal references to entities (Florian et al., 2004). In this paper we use "mentions" and "entities" interchangeably.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 75–84,
Austin, Texas, November 1-5, 2016.

ties, the first one, ***hiatal hernia***, is a conventional contiguous entity. The second one, ***laceration ... esophagus***, is a discontiguous entity, consisting of two parts. The third and fourth ones, ***blood in stomach*** and ***stomach ... lac*** (for *stomach laceration*), are overlapping with each other, with the fourth being discontiguous at the same time.

For such discontiguous entities which can potentially overlap with other entities in complex manners, existing approaches such as those based on simple sequence tagging models have difficulties handling them accurately. This stems from the fact that there is a very large number of possible entity combinations in a sentence when the entities can be discontiguous and overlapping.

Motivated by this, in this paper we propose a novel model that can better represent both contiguous and discontiguous entities which can overlap with one another. Our major contributions can be summarized as follows:

- We propose a novel model that is able to represent both contiguous and discontiguous entities.
- Theoretically, we introduce the notion of *model ambiguity* for quantifying the ambiguity of different NER models that can handle discontiguous entities. We present a study and make comparisons about different models' ambiguity under this theoretical framework.
- Empirically, we demonstrate that our model can significantly outperform conventional approaches designed for handling discontiguous entities on data which contains many discontiguous entities.

2 Related Work

Learning to recognize named entities is a popular task in the field of natural language processing. A survey by Nadeau (2007) lists several approaches in NER, including Hidden Markov Models (HMM) (Bikel et al., 1997), Decision Trees (Sekine, 1998), Maximum Entropy Models (Borthwick and Sterling, 1998), Support Vector Machines (SVM) (Asahara and Matsumoto, 2003), and also semi-supervised and unsupervised approaches. Ratinov (2009) utilizes averaged perceptron to solve this problem and also focused on four key design decisions, achieving state-of-the-art in MUC-7 dataset. These approaches work on standard texts, such as news articles, and the entities to be recognized are defined to be contiguous and non-overlapping.

Noticing that many named entities contain other named entities inside them, Finkel and Manning (2009) proposed a model that is capable of extracting nested named entities by representing the sentence as a constituency parse tree, with named entities as phrases. As a parsing-based model, the approach has a time complexity that is cubic in the number of words in the sentence.

Recently, Lu and Roth (2015) proposed a model that can represent overlapping entities. In addition to supporting nested entities, theoretically this model can also represent overlapping entities where neither is nested in another. The model represents each sentence as a hypergraph with nodes indicating entity types and boundaries. Compared to the previous model, this model has a lower time complexity, which is linear in the number of words in the sentence.

All the above models focus on NER in conventional texts, where the assumption of contiguous entities is valid. In the past few years, there is a growing body of works on recognizing disorder mentions in clinical text. These disorder mentions may be discontiguous and also overlapping. To tackle such an issue, a research group from University of Texas Health Science Center at Houston (Tang et al., 2013; Zhang et al., 2014; Xu et al., 2015) first utilized a conventional linear-chain CRF to recognize disorder mention parts by extending the standard BIO (*Begin, Inside, Outside*) format, and next did some postprocessing to combine different components. Though effective, as we will see later, such a model comes with some drawbacks. Nevertheless, their work motivated us to perform further analysis on this issue and propose a novel model specifically designed for discontiguous entity extraction.

3 Models

3.1 Linear-chain CRF Model

Before we present our approach, we would like to spend some time to discuss a simple approach based on linear-chain CRFs (Lafferty et al., 2001). This approach is primarily based on the system by Tang et al. (2013), and this will be the baseline system

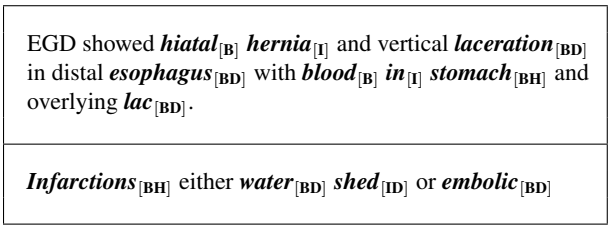


Figure 2: Entity encoding in the linear-chain model. **Top**: for the example in Fig 1. **Bottom**: for the second example in Fig 4. The **O** labels are not shown.

that we will make comparison with in later sections.

The problem is regarded as a sequence prediction task, where each word is assigned a label similar to BIO format often used for NER. We used the encoding used by Tang et al. (2013), which uses 7 tags to handle entities that can be discontiguous and overlapping. Specifically, we used **B**, **I**, **O**, **BD**, **ID**, **BH**, and **IH** to denote **B**eginning of entity, **I**nside entity, **O**utside of entity, **B**eginning of **D**iscontiguous entity, **I**nside of **D**iscontiguous entity, **B**eginning of **H**ead, and **I**nside of **H**ead. To encode a sentence in this format, first we identify the contiguous word sequences which are parts of multiple entities. We call these *head components* and we label each word inside each component with **BH** (for the first word in *each component*) or **IH**. Then we find contiguous word sequences which are parts of a discontiguous entity, which we call the *body components*. Words inside those components which have not been labeled are labeled with **BD** (for the first word in *each component*) or **ID**. Finally, words that are parts of a contiguous entity are called *contiguous component*, and, if they have not been labeled, are labeled as **B** (for the first word in *each component*) or **I**.

This encoding is lossy, since the information on which parts constitute the same entity is lost. The top example in Figure 2 is the encoding of the example shown in Figure 1. During decoding, based on the labels only it is not entirely clear whether "***laceration***" should be combined with "***esophagus***" or with "***stomach***" to form a single mention. For the bottom example, we cannot deduce that "***Infarctions***" alone is a mention, since there is no difference in the encoding of a sentence with only two mentions {"***Infarctions ... water shed***", "***Infarctions ... embolic***"} or having three mentions with "***Infarctions***" as another mention, since in both cases, the word "***Infarctions***" is labeled with **BH**.

Also, it should be noted that some of the label sequences are not valid. For example, a sentence in which there is only one word labeled as **BD** is invalid, since a discontiguous entity requires at least two words to be labeled as **BD** or **BH**. This is, however, a possible output from the linear CRF model, due to the Markov assumption inherent in linear CRF models. Later we see that our models do not have this problem.

3.2 Our Model

Linear-chain CRF models are limited in their representational power when handling complex entities, especially when they can be discontiguous and can overlap with one another. While recent models have been proposed to effectively handle overlapping entities, how to effectively handle discontiguous entities remains a research question to be answered. Motivated by previous efforts on handling overlapping entities (Lu and Roth, 2015), in this work we propose a model based on hypergraphs that can better represent entities that can be discontiguous and at the same time be overlapping with each other.

Unlike the previous work (Lu and Roth, 2015), we establish a novel theoretical framework to formally quantify the ambiguity of our hypergraph-based models and justify their effectiveness by making comparisons with the linear-chain CRF approach.

Now let us introduce our novel hypergraph representation. A hypergraph can be used to represent entities of different types and their combinations in a given sentence. Specifically, a hypergraph is constructed as follows. For the word at position k, we have the following nodes:

- $\mathbf{A}^k$: this node represents all entities that begin with the current or a future word (to the right of the current word).
- $\mathbf{E}^k$: this node represents all entities that begin with the current word.
- $\mathbf{T}_t^k$: this node represents entities of certain specific type t that begin with the current word. There is one $\mathbf{T}_t^k$ for each different type.
- $\mathbf{B}_{t,i}^k$: this node indicates that the current word is part of the i-th component of an entity of type t.
- $\mathbf{O}_{t,i}^k$: this node indicates that the current word appears in between $(i\text{-}1)$-th and i-th components of an entity of type t.

There is also a special leaf node, $\mathbf{X}$-node, which indicates the end (*i.e.*, right boundary) of an entity.

The nodes are connected by directed hyperedges, which for the purpose of explaining our models are defined as those edges that connect one node, called the parent node, to one or more child nodes. For ease of notation, in the rest of this paper we use *edge* to refer to directed hyperedge.

The edges Each $\mathbf{A}^k$ is a parent to $\mathbf{E}^k$ and $\mathbf{A}^{k+1}$, encoding the fact that the set of all entities at position k is the union of the set of entities starting **e**xactly at current position ($\mathbf{E}^k$) with the set of entities starting at or **a**fter position $k+1$ ($\mathbf{A}^{k+1}$).

Each $\mathbf{E}^k$ is a parent to $\mathbf{T}^k_1, \ldots, \mathbf{T}^k_T$, where T is the total number of possible types that we consider. Each $\mathbf{T}^k_t$ has two edges where it serves as a parent, within one it is parent to $\mathbf{B}^k_{t,0}$ and within another it is to $\mathbf{X}$. These edges encode the fact that at position k, either there is an entity of type t that begins with the current word (to $\mathbf{B}^k_{t,0}$), or there is no entity of type t that begins with the current word (to $\mathbf{X}$).

In the full hypergraph, each $\mathbf{B}^k_{t,i}$ is a parent to $\mathbf{B}^{k+1}_{t,i}$ (encoding the fact that the next word also belongs to the same component of the same entity), to $\mathbf{O}^{k+1}_{t,i+1}$ (encoding the fact that this word is part of a discontiguous entity, and the next word is the first word separating current component and the next component), and to $\mathbf{X}$ (representing that the entity ends at this word). Also there are edges with all possible combinations of $\mathbf{B}^{k+1}_{t,i}$, $\mathbf{O}^{k+1}_{t,i+1}$, and $\mathbf{X}$ as the child nodes, representing overlapping entities. For example, the edge $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{B}^{k+1}_{t,i}, \mathbf{X})$ denotes that there is an entity which continues to the next word (the edge to $\mathbf{B}^{k+1}_{t,i}$), while there is another entity ending at k-th word (the edge to $\mathbf{X}$). In total there are 7 edges in which $\mathbf{B}^k_{t,i}$ is a parent, which are:

- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{X})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{O}^{k+1}_{t,i+1})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{O}^{k+1}_{t,i+1}, \mathbf{X})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{B}^{k+1}_{t,i})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{B}^{k+1}_{t,i}, \mathbf{X})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{B}^{k+1}_{t,i}, \mathbf{O}^{k+1}_{t,i+1})$
- $\mathbf{B}^k_{t,i} \rightarrow (\mathbf{B}^{k+1}_{t,i}, \mathbf{O}^{k+1}_{t,i+1}, \mathbf{X})$

Analogously, $\mathbf{O}^k_{t,i}$ has three edges that connect to $\mathbf{O}^{k+1}_{t,i}$, $\mathbf{B}^{k+1}_{t,i+1}$, and both. Note that $\mathbf{O}^k_{t,i}$ is not a parent to $\mathbf{X}$ by definition.

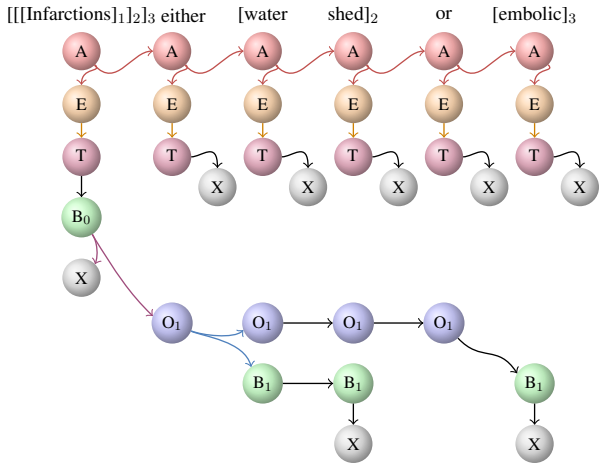


Figure 3: The hypergraph for SHARED model for the second example in Figure 4. The type information in $\mathbf{T}$, $\mathbf{B}$, and $\mathbf{O}$-nodes is not shown. The $\mathbf{X}$-node is drawn multiple times for better visualization.

During testing, the model will predict a **subgraph** which will result in the predicted entities after decoding. We call this subgraph representing certain entity combination ***entity-encoded hypergraph***.

For example, Figure 3 shows the entity-encoded hypergraph of our model encoding the three mentions in the second example in Figure 4. The edge from the $\mathbf{T}$-node for the first word to the $\mathbf{B}$-node for the first word shows that there is at least one entity starting with this word. The three places where an $\mathbf{X}$-node is connected to a $\mathbf{B}$-node show the end of the three entities. Note that this hypergraph clearly shows the presence of the three mentions without ambiguity, unlike a linear-chain encoding of this example where it cannot be inferred that "***Infarctions***" alone is a mention, as discussed previously. In this paper, we set the maximum number of components to be 3 since the dataset does not contain any mention with more than 3 components.

Also note that this model supports discontiguous and overlapping mentions of different types since each type has its own set of $\mathbf{O}$-nodes and $\mathbf{B}$-nodes, unlike the linear-chain model, which supports only overlapping mentions of the same type.

We also experimented with a variant of this model, where we split the $\mathbf{T}$-nodes, $\mathbf{B}$-nodes, and $\mathbf{O}$-nodes further according to the number of components. We split $\mathbf{B}^k_{t,i}$ into $\mathbf{B}^k_{t,i,j}$, $i = 1 \ldots j, j =$

$1 \ldots 3$ which represents that the word is part of the i-th component of a mention with total j components. Similarly we split $\mathbf{O}^k_{t,i}$ into $\mathbf{O}^k_{t,i,j}$ and $\mathbf{T}^k_t$ into $\mathbf{T}^k_{t,j}$. We call the original version SHARED model, and this variant SPLIT model. The motivation for this variant is that the majority of overlaps in the data are between discontiguous and contiguous entities, and so splitting the two cases – one component (contiguous) and more (discontiguous) – will reduce ambiguity for those cases.

These models are still ambiguous to some degree, for example when an $\mathbf{O}$-node has two child nodes and two parents, we cannot decide which of the parent node is paired with which child node. However, in this paper we argue that:

- This model is less ambiguous compared to the linear-chain model, as we will show later theoretically and empirically.
- Every output of our model is a valid prediction, unlike the linear-chain model since this model will always produce a valid path from $\mathbf{T}$-nodes to the $\mathbf{X}$-nodes representing some entities.

We will also show through experiments that our models can encode the entities more accurately.

3.3 Interpreting Output Structures

Both the linear-chain CRF model and our models are still ambiguous to some degree, so we need to handle the ambiguity in interpreting the output structures into entities. For all models, we define two general heuristics: ENOUGH and ALL. The ENOUGH heuristic handles ambiguity by trying to produce a minimal set of entities which encodes to the one produced by the model, while ALL heuristic handles ambiguity by producing the union of all possible entity combinations that encode to the one produced by the model. For more details on how these heuristics are implemented for each model, please refer to the supplementary material.

3.4 Training

For both models, the training follows a log-linear formulation, by maximizing the loglikelihood of the training data $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left[\sum_{e\in\mathcal{E}(\mathbf{x},\mathbf{y})} \left[\mathbf{w}^T \mathbf{f}(e)\right] - \log Z_{\mathbf{w}}(\mathbf{x}) \right] - \lambda ||\mathbf{w}||^2$$

Here $(\mathbf{x}, \mathbf{y})$ is a training instance consisting of the sentence $\mathbf{x}$ and the entity-encoded hypergraph $\mathbf{y} \in \mathcal{Y}$ where $\mathcal{Y}$ is the set of all possible mention-encoded hypergraphs. The vector $\mathbf{w}$ consists of feature weights, which are the model parameters to be learned. The set $\mathcal{E}(\mathbf{x}, \mathbf{y})$ consists of all edges present in the entity-encoded hypergraph $\mathbf{y}$ for input $\mathbf{x}$. The function $\mathbf{f}(e)$ returns the features defined over the edge e, $Z_{\mathbf{w}}(\mathbf{x})$ is the normalization term which gives the sum of scores over all possible entity-encoded hypergraphs in $\mathcal{Y}$ that is relevant to the input $\mathbf{x}$, and finally λ is the ℓ_2-regularization parameter.

4 Model Ambiguity

The main aim of this paper is to assess how well each model can represent the discontiguous entities, even in the presence of overlapping entities.

In this section, we will theoretically compare the models' ambiguity, which is defined as the average number of mention combinations that map to the same encoding in a model. Now, to compare two models, instead of calculating the ambiguity directly, we can calculate the ***relative ambiguity*** between the two models directly by comparing the number of *canonical encodings* in the two models.

A canonical encoding is a fixed, selected representation of a particular set of mentions in a sentence, among (possibly) several alternative representations. Several alternatives may be present due to the ambiguity of the encoding-decoding process and also since the output of the model is not restricted to a specific rule. For example, for the text "John Smith", a model trained in BIO format might output "B-PER I-PER" or "I-PER I-PER", and both will still mean that "John Smith" is a person, although the "correct" encoding would of course be "B-PER I-PER", which is selected as the canonical encoding. Intuitively, a canonical encoding is a formal way to say that we only consider the "correct" encodings.

A model with larger number of canonical encodings will, on average, have less ambiguity compared to the one with smaller number of canonical encodings. Subsequently, a model with less ambiguity will be more precise in predicting entities.

Let $\mathcal{M}_{\text{LI}}(n), \mathcal{M}_{\text{SH}}(n), \mathcal{M}_{\text{SP}}(n)$ denote the number of canonical encodings of the linear-chain, SHARED, and SPLIT model, respectively, for a sen-

tence with n words. Then we formally define the relative ambiguity of model M_1 over model M_2, $\mathcal{A}_r(M_1, M_2)$, as follows:

$$\mathcal{A}_r(M_1, M_2) = \lim_{n\to\infty} \frac{\log \sum_{i=1}^{n} \mathcal{M}_{M_2}(i)}{\log \sum_{i=1}^{n} \mathcal{M}_{M_1}(i)} \quad (1)$$

$\mathcal{A}_r(M_1, M_2) > 1$ means model M_1 is more ambiguous than M_2. Now, we claim the following:

Theorem 4.1. $\mathcal{A}_r(\text{LI}, \text{SH}) > 1$

We provide a proof sketch below. Due to space limitation, we cannot provide the full dynamic programming calculation. We refer the reader to the supplementary material for the details.

Proof Sketch The number of canonical encodings in the linear-chain model is less than 7^n since there are 7 possible tags for each of the n words and not all of the 7^n tag sequences are canonical encodings. So we have $\mathcal{M}_{\text{LI}}(n) < 7^n$ and thus we can derive $\log \sum_{i=1}^{n} \mathcal{M}_{\text{LI}}(i) < 3n \log 2$.

For our models, by employing some dynamic programming adapted from the inside algorithm (Baker, 1979), we can calculate the growth order of the number of canonical encodings for SHARED model to arrive at a conclusion that $\forall n > n_0,\ \sum_{i=1}^{n} \mathcal{M}_{\text{SH}}(i) > C \cdot 2^{10n}$ for some constants n_0, C. Then we have:

$$\mathcal{A}_r(\text{LI}, \text{SH}) \geq \lim_{n\to\infty} \frac{\log C + 10n \log 2}{3n \log 2} = \frac{10}{3} > 1 \quad (2)$$

□

Theorem 4.1 says that the linear-chain model is more ambiguous compared to our SHARED model. Similarly, we can also establish $\mathcal{A}_r(\text{SH}, \text{SP}) > 1$. Later we also see this empirically from experiments.

5 Experiments

5.1 Data

To allow us to conduct experiments to empirically assess different models' capability in handling entities that can be discontiguous and can potentially overlap with one another, we need a text corpus annotated with entities which can be discontiguous and overlapping with other entities. We found the largest of such corpus to be the dataset from the task to recognize disorder mentions in clinical text, initially organized by ShARe/CLEF eHealth Evaluation Lab (SHEL) in 2013 (Suominen et al., 2013) and continued in SemEval-2014 (Pradhan et al., 2014a).

The definition of the task is to recognize mentions of concepts that belong to the Unified Medical Language System (UMLS) semantic group *disorders* from a set of clinical texts. Each text has been annotated with a list of disorder mentions by two professional coders trained for this task, followed by an open adjudication step (Suominen et al., 2013).

Unfortunately, even in this dataset, only 8.95% of the mentions are discontiguous. Working directly on such data would prevent us from understanding the true effectiveness of different models when handling entities which can be discontiguous and overlapping. In order to truly understand how different models behave on data with discontiguous entities, we consider a subset of the data where we consider those sentences which contain at least one discontiguous entity. We call the resulting subset the "Discontiguous" subset of the "Original" dataset. Later we will also still use the training data of the "Original" dataset in the experiments.

Note that this "Discontiguous" subset still contains contiguous entities since a sentence usually contains more than one entity. The subset is a balanced dataset with 53.61% of the entities being discontiguous and the rest contiguous. We then split this dataset into training, development, and test set, according to the split given in SemEval 2014 setting (henceforth LARGE dataset). To see the impact of dataset size, we also experiment on a subset of the LARGE dataset, following the SHEL 2013 setting, with the development set in the LARGE dataset used as test set (henceforth SMALL dataset). The training and development set of the SMALL dataset comes from a random 80% (Tr80) and 20% (Tr20) split of the training set in LARGE dataset.

The statistics of the datasets, including the number of overlaps between the entities in the "All" column, are shown in Table 1.

We note that this dataset only contains one type of entity. In later experiments, in order to evaluate the models on multiple types, we create another dataset where we split the entities based on the entity-level semantic category. This information is available for some entities through the Concept Unique Identifier (CUI) annotation in the data. In total we have three types: two types (type `A` and `B`) based on the semantic category, and one type (type `N`) for those entities

Split	#Sentences	Number of mentions				#Overlaps	
		1 part	2 parts	3 parts	Total	All	Diff
Train	534	544	607	44	1,195	205	58
- Tr80	416	448	476	33	957	164	48
- Tr20	118	96	131	11	238	41	10
Dev	303	357	421	18	796	240	28
Test	430	584	610	16	1,210	327	61

Table 1: The statistics of the data. Tr80 and Tr20 refers to the 80% and 20% partitions of the full training data.

having no semantic category information[2]. See the supplementary material for more details. The number of overlaps between different types is shown in the "Diff" column in Table 1. Except for a handful overlaps in development set, all overlaps involve at least one discontiguous entity. Our main result will still be based on the dataset with one type of entity.

The patient had blood in his mouth and on his tongue, pupils were pinpoint and reactive.
- ***blood in his mouth***
- ***blood . . . on his tongue***
- ***pupils . . . pinpoint***

Infarctions either water shed or embolic
- ***Infarctions***
- ***Infarctions . . . water shed***
- ***Infarctions . . . embolic***

You see blood or dark/black material when you vomit or have a bowel movement.
- ***blood . . . vomit***
- ***blood . . . bowel movement***
- ***dark . . . material . . . vomit***
- ***dark . . . bowel movement***
- ***black material . . . vomit***
- ***black material . . . bowel movement***

Figure 4: Examples of discontiguous and overlapping mentions, taken from the dataset.

Figure 4 shows some examples of the mentions. The first example shows two discontiguous mentions that do not overlap. The second example shows a typical discontiguous and overlapping case. The last example shows a very hard case of overlapping and discontiguous mentions, as each of the components in {***blood***, ***dark***, ***black material***} is paired with each of the word in {***vomit***, ***bowel movement***}, resulting in six mentions in total, with one having three components (***dark . . . material . . . vomit***).

5.2 Features

Motivated by the features used by Zhang et al. (2014), for both the linear-chain CRF model and our models we use the following features: neighbouring words with relative position information (we consider previous and next k words, where k=1, 2, 3), neighbouring words with relative position information paired with the current word, word n-grams containing the current word (n=2,3), POS tag for the current word, POS tag n-grams containing the current word (n=2,3), orthographic features (prefix, suffix, capitalization, lemma), note type (discharge summary, echo report, radiology, and ECG report), section name (*e.g.* Medications, Past Medical History)[3], Brown cluster, and word-level semantic category information[4]. We used Stanford POS tagger (Toutanova et al., 2003) for POS tagging, and NLP4J package[5] for lemmatization. For Brown cluster features, following Tang et al. (2013), we used 1,000 clusters from the combination of training, development, and test set, and used all the subpaths of the cluster IDs as features.

5.3 Experimental Setup

We evaluated the three models on the SMALL dataset and the LARGE dataset.

Note that in both the SMALL and LARGE dataset, about half of all mentions are discontiguous, both in training and test set. We also want to see whether training on a set where the majority of the mentions are contiguous will affect the performance on recognizing discontiguous mentions. So we also performed another experiment where we trained each model on the original training set where the majority of the entities are contiguous. We refer to this original dataset as "Train-Orig" (it contains 10,405 sentences, including those with no entities) and the

[2]It is tempting to just ignore these entities since the N type does not convey any specific information about the entities in it. However, due to the dataset size, excluding this type will lead to very small number of interactions between types. So we decided to keep this type

[3]Section names were determined by some heuristics, refer to the supplementary material for more information

[4]This is standard information that can be extracted from UMLS. See (Zhang et al., 2014) for more details.

[5]http://www.github.com/emorynlp/nlp4j/

	SMALL						LARGE					
	Train-Disc			Train-Orig			Train-Disc			Train-Orig		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LI-ENH	59.7	39.8	47.8	71.0	45.8	55.7	54.7	41.2	47.0	64.1	46.5	53.9
LI-ALL	16.6	**43.5**	24.1	55.5	**49.2**	52.2	15.2	**44.9**	22.7	52.8	49.4	51.1
SH-ENH	85.9	39.7	**54.3**	82.2	48.0	60.6	76.9	40.1	52.7	73.9	49.1	59.0
SH-ALL	85.9	39.7	**54.3**	82.2	48.0	60.6	76.0	40.5	**52.8**	73.4	**49.5**	59.1
SP-ENH	**86.7**	37.8	52.7	**82.5**	48.0	**60.7**	**79.4**	38.6	52.0	**75.3**	48.8	**59.2**
SP-ALL	**86.7**	37.8	52.7	**82.5**	48.0	**60.7**	**79.4**	38.6	52.0	**75.3**	48.8	**59.2**

Table 2: Results on the two datasets and two different training data after optimizing regularization hyperparameter λ in development set. The -ENH and -ALL suffixes refer to the ENOUGH and ALL heuristics. The best result in each column is put in boldface.

earlier one as "Train-Disc".

First we trained each model on the training set, varying the regularization hyperparameter λ,[6] then the λ with best result in the development set using the respective ENOUGH heuristic for each model is chosen for final result in the test set.

For each experiment setting, we show precision (P), recall (R) and F1 measure. Precision is the percentage of the mentions predicted by the model which are correct, recall is the percentage of mentions in the dataset correctly discovered by the model, and F1 measure is the harmonic mean of precision and recall.

5.4 Results and Discussions

The full results are recorded in Table 2.

We see that in general our models have higher precision compared to the linear-chain baseline. This is expected, since our models have less ambiguity, which means that from a given output structure it is easier in our model to get the correct interpretation. We will explore this more in Section 5.5.

The ALL heuristic, as expected, results in higher recall, and this is more pronounced in the linear-chain model, with up to 4% increase from the ENOUGH heuristic, achieving the highest recall in three out of four settings. The high recall of the ALL heuristic in the linear-chain model can be explained by the high level of ambiguity the model has. Since it has more ambiguity compared to our models, one label sequence predicted by the model produces a lot of entities, and so it is more likely to overlap with the gold entities. But this has the drawback of very low precision as we can see in the result.

We see switching from one heuristic to the other does not affect the results of our models much. Looking at the output of our models, they tend to produce output structures with less ambiguity, which causes little difference in the two heuristics.

One example where the baseline made a mistake is the sentence: "Ethanol Intoxication and withdrawal". The gold mentions are "Ethanol Intoxication" and "Ethanol withdrawal". But the linear-chain model labeled it as "$[\text{Ethanol}]_{[B]}$ $[\text{Intoxication}]_{[I]}$ and $[\text{withdrawal}]_{[BD]}$", which is inconsistent since there is only one discontiguous component. Our models do not have this issue because in our models every subgraph that may be predicted translates to valid mention combinations, as discussed in Section 3.2.

In the "Train-Orig" column, we see that all models can recognize discontiguous entities better when given more data, even though the majority of the entities in "Train-Orig" are contiguous.

5.5 Experiments on Ambiguity

To see the ambiguity of each model empirically, we run the decoding process for each model given the gold output structure, which is the true label sequence for the linear-chain model and the true mention-encoded hypergraph for our models.

We used the entities from the training and development sets for this experiment, and we compare the "Original" datasets with the "Discontiguous" subset to see that the ambiguity is more pronounced when there are more discontiguous entities. Then we show the precision and recall errors (defined as $1 - P$ and $1 - R$, respectively) in Table 3.

Since the ALL heuristics generates all possible mentions from the given encoding, theoretically it should give perfect recall. However, due to errors in the training data, there are mentions which can-

[6] Taken from the set $\{0.125, 0.25, 0.5, 1.0, 2.0\}$

	Discontiguous		Original	
	Prec Err	Rec Err	Prec Err	Rec Err
LI-ALL	63.66%	0.30%	23.81%	0.17%
SH-ALL	1.73%	0.30%	0.35%	0.17%
SP-ALL	1.05%	0.30%	0.22%	0.17%
LI-ENH	2.74%	3.82%	0.52%	0.90%
SH-ENH	1.21%	1.46%	0.25%	0.38%
SP-ENH	0.75%	0.90%	0.17%	0.28%

Table 3: Precision and recall errors (%) of each model in the "Discontiguous" and "Original" datasets when given the gold output structure (label sequence in linear-chain model, hypergraph in our models). Lower numbers are better.

Type	#Ent	Linear-chain			SHARED			SPLIT		
		P	R	F	P	R	F	P	R	F
A	289	69.8	59.9	64.4	79.4	56.1	65.7	81.0	56.1	66.3
B	418	50.0	34.0	40.5	56.8	29.0	38.4	58.2	28.0	37.8
N	503	62.1	37.8	47.0	84.8	43.3	57.4	84.9	42.4	56.5
Total	1210	60.3	41.7	49.3	74.3	41.4	53.2	75.5	40.7	52.9

Table 4: Results on the LARGE dataset when entities are split into three types: A, B, and N. #Ent is the number of entities

not be properly encoded in the models[7]. Removing these errors results in perfect recall (0% recall error). This means that all models are complete: they can encode any mention combinations.

We see however, a very huge difference on the precision error between the linear-chain model and our models, even more when most of the entities are discontiguous. For the discontiguous subset with the ALL heuristic, the linear-chain model produced 5,463 entities, while the SHARED and SPLIT model produced 2,020 and 2,006 entities, respectively. The total number of gold entities is 1,991. This means one encoding in the linear-chain model produces much more distinct mention combinations compared to our model, which again shows that the linear-chain model has more ambiguity. Similarly, we can deduce that the SHARED model has slightly more ambiguity compared to the SPLIT model. This confirms our theoretical result presented previously.

It is also worth noting that in the ENOUGH heuristic our models have smaller errors compared to the linear-chain model, showing that when both models can predict the true output structure (the correct label sequence for the baseline model and mention-encoded hypergraph for our models), it is easier in our models to get the desired mention combinations.

5.6 Experiments on Multiple Entity Types

We used the LARGE dataset with the multiple-type entities for this experiment. We ran our two models and the linear-chain CRF model with the ENOUGH heuristic on this multi-type dataset, in the same setting as Train-Orig in previous experiments, and the result is shown in Table 4. We used the best lambda from the main experiment for this experiment.

There is a performance drop compared to the LARGE-Train-Orig results in Table 2, which is expected since the presence of multiple types make the task harder. But in general we still see that our models are still better than the baseline, especially the SPLIT model, which shows that in the presence of multiple types, our models can still work better than the baseline model.

6 Conclusions and Future Work

In this paper we proposed new models that can better represent discontiguous entities that can be overlapping at the same time. We validated our claims through theoretical analysis and empirical analysis on the models' ambiguity, as well as their performances on the task of recognizing disorder mentions on datasets with a substantial number of discontiguous entities. When the true output structure is given, which is still ambiguous in all models, our models show that it is easier to produce the desired mention combinations compared to the linear-chain CRF model with reasonable heuristics. We note that an extension similar to semi-Markov or weak semi-Markov (Muis and Lu, 2016) is possible for our models. We leave this for future investigations.

The supplementary material and our implementations for the models are available at:

`http://statnlp.org/research/ie`

Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback, and also the ShARe/CLEF eHealth Evaluation Lab for providing us the dataset. This work is supported by MOE Tier 1 grant SUTDT12015008.

[7]There are 19 errors in the original dataset, and 6 in the discontiguous subset, which include duplicate mentions and mentions with incorrect boundaries

References

Rami Al-Rfou and Steven Skiena. 2012. SpeedRead: A Fast Named Entity Recognition Pipeline. *Proceedings of COLING 2012*, pages 51–66.

Masayuki Asahara and Yuji Matsumoto. 2003. Japanese Named Entity Extraction with Redundant Morphological Analysis. In *Proceedings of HLT-NAACL '03*, volume 1, pages 8–15.

James K Baker. 1979. Trainable Grammars for Speech Recognition. *Journal of the Acoustical Society of America*, 65(S1):S132.

Daniel M. Bikel, Scott Miller, Richard M. Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. *Proceedings of the fifth conference on Applied Natural Language Processing (ANLP '97)*, pages 194–201.

Andrew Borthwick and John Sterling. 1998. NYU: Description of the MENE named entity system as used in MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, volume 1, pages 141–150.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of HLT-NAACL '04*, pages 1–8.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering Relations Among Named Entities from Large Corpora. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 415–422.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning (ICML '01)*, pages 282–289.

Wei Lu and Dan Roth. 2015. Joint Mention Extraction and Classification with Mention Hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*, pages 857–867.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of HLT-NAACL '03*, volume 4, pages 188–191.

Aldrian Obaja Muis and Wei Lu. 2016. Weak Semi-Markov CRFs for Noun Phrase Chunking in Informal Text. In *Proceedings of HLT-NAACL '16*, pages 714–719.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Sameer Pradhan, Noémie Elhadad, Wendy W. Chapman, Suresh Manandhar, and Guergana Savova. 2014a. SemEval-2014 Task 7: Analysis of Clinical Text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 54–62.

Sameer Pradhan, Noémie Elhadad, Brett R. South, David Martinez, Lee Christensen, Amy Vogel, Hanna Suominen, Wendy W. Chapman, and Guergana Savova. 2014b. Evaluating the state of the art in disorder recognition and normalization of the clinical narrative. *Journal of the American Medical Informatics Association : JAMIA*, 22(1):143–54.

Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL '09)*, pages 147–155.

Satoshi Sekine. 1998. NYU: Description of the Japanese NE system used for MET-2. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W. Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R. South, Danielle L. Mowery, Gareth J. F. Jones, Johannes Leveling, Liadh Kelly, Lorraine Goeuriot, David Martinez, and Guido Zuccon, 2013. *Overview of the ShARe/CLEF eHealth Evaluation Lab 2013*, pages 212–231. Springer Berlin Heidelberg, Berlin, Heidelberg.

Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2013. Recognizing clinical entities in hospital discharge summaries using Structural Support Vector Machines with word representation features. *BMC medical informatics and decision making*, 13 Suppl 1(Suppl 1):S1.

Kristina Toutanova, Dan Klein, and Christopher D Manning. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL '03*, volume 1, pages 252–259.

Jun Xu, Yaoyun Zhang, Jingqi Wang, Yonghui Wu, and Min Jiang. 2015. UTH-CCB : The Participation of the SemEval 2015 Challenge Task 14. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 311–314.

Yaoyun Zhang, Jingqi Wang, Buzhou Tang, Yonghui Wu, Min Jiang, Yukun Chen, and Hua Xu. 2014. UTH_CCB: A report for SemEval 2014 – Task 7 Analysis of Clinical Text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 802–806.

Modeling Human Reading with Neural Attention

Michael Hahn **Frank Keller**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
s1582047@inf.ed.ac.uk keller@inf.ed.ac.uk

Abstract

When humans read text, they fixate some words and skip others. However, there have been few attempts to explain skipping behavior with computational models, as most existing work has focused on predicting reading times (e.g., using surprisal). In this paper, we propose a novel approach that models both skipping and reading, using an unsupervised architecture that combines a neural attention with autoencoding, trained on raw text using reinforcement learning. Our model explains human reading behavior as a tradeoff between precision of language understanding (encoding the input accurately) and economy of attention (fixating as few words as possible). We evaluate the model on the Dundee eye-tracking corpus, showing that it accurately predicts skipping behavior and reading times, is competitive with surprisal, and captures known qualitative features of human reading.

1 Introduction

Humans read text by making a sequence of fixations and saccades. During a fixation, the eyes land on a word and remain fairly static for 200–250 ms. Saccades are the rapid jumps that occur between fixations, typically lasting 20–40 ms and spanning 7–9 characters (Rayner, 1998). Readers, however, do not simply fixate one word after another; some saccades go in reverse direction, and some words are fixated more than once or skipped altogether.

A range of computational models have been developed to account for human eye-movements in reading (Rayner and Reichle, 2010), including models of saccade generation in cognitive psychology, such as EZ-Reader (Reichle et al., 1998, 2003, 2009), SWIFT (Engbert et al., 2002, 2005), or the Bayesian Model of Bicknell and Levy (2010). More recent approaches use machine learning models trained on eye-tracking data to predict human reading patterns (Nilsson and Nivre, 2009, 2010; Hara et al., 2012; Matthies and Søgaard, 2013). Both types of models involve theoretical assumptions about human eye-movements, or at least require the selection of relevant eye-movement features. Model parameters have to be estimated in a supervised way from eye-tracking corpora.

Unsupervised approaches, that do not involve training the model on eye-tracking data, have also been proposed. A key example is *surprisal*, which measures the predictability of a word in context, defined as the negative logarithm of the conditional probability of the current word given the preceding words (Hale, 2001; Levy, 2008). Surprisal is computed by a language model, which can take the form of a probabilistic grammar, an n-gram model, or a recurrent neural network. While surprisal has been shown to correlate with word-by-word reading times (McDonald and Shillcock, 2003a,b; Demberg and Keller, 2008; Frank and Bod, 2011; Smith and Levy, 2013), it cannot explain other aspects of human reading, such as reverse saccades, re-fixations, or skipping. Skipping is a particularly intriguing phenomenon: about 40% of all words are skipped (in the Dundee corpus, see below), without apparent detriment to text understanding.

In this paper, we propose a novel model architecture that is able to explain which words are skipped and which ones are fixated, while also predicting

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 85–95,
Austin, Texas, November 1-5, 2016.

reading times for fixated words. Our approach is completely unsupervised and requires only unlabeled text for training.

Compared to language as a whole, reading is a recent innovation in evolutionary terms, and people learning to read do not have access to competent readers' eye-movement patterns as training data. This suggests that human eye-movement patterns emerge from general principles of language processing that are independent of reading. Our starting point is the *Tradeoff Hypothesis*: Human reading optimizes a tradeoff between *precision* of language understanding (encoding the input accurately) and *economy* of attention (fixating as few words as possible). Based on the Tradeoff Hypothesis, we expect that humans only fixate words to the extent necessary for language understanding, while skipping words whose contribution to the overall meaning can be inferred from context.

In order to test these assumptions, this paper investigates the following questions:

1. Can the Tradeoff Hypothesis be implemented in an unsupervised model that predicts skipping and reading times in *quantitative* terms? In particular, can we compute surprisal based only on the words that are actually fixated?

2. Can the Tradeoff Hypothesis explain known *qualitative* features of human fixation patterns? These include dependence on word frequency, word length, predictability in context, a contrast between content and function words, and the statistical dependence of the current fixation on previous fixations.

To investigate these questions, we develop a generic architecture that combines neural language modeling with recent ideas on integrating recurrent neural networks with mechanisms of attention, which have shown promise both in NLP and in computer vision. We train our model end-to-end on a large text corpus to optimize a tradeoff between minimizing input reconstruction error and minimizing the number of words fixated. We evaluate the model's reading behavior against a corpus of human eye-tracking data. Apart from the unlabeled training corpus and the generic architecture, no further assumptions about language structure are made – in particular, no lexicon or grammar or otherwise labeled data is required.

Our unsupervised model is able to predict human skips and fixations with an accuracy of 63.7%. This compares to a baseline of 52.6% and a supervised accuracy of 69.9%. For fixated words, the model significantly predicts human reading times in a linear mixed effects analysis. The performance of our model is comparable to surprisal, even though it only fixates 60.4% of all input words. Furthermore, we show that known qualitative features of human fixation sequences emerge in our model without additional assumptions.

2 Related Work

A range of attention-based neural network architectures have recently been proposed in the literature, showing promise in both NLP and computer vision (e.g., Mnih et al., 2014; Bahdanau et al., 2015). Such architectures incorporate a mechanism that allows the network to dynamically focus on a restricted part of the input. Attention is also a central concept in cognitive science, where it denotes the focus of cognitive processing. In both language processing and visual processing, attention is known to be limited to a restricted area of the visual field, and shifts rapidly through eye-movements (Henderson, 2003).

Attention-based neural architectures either employ *soft attention* or *hard attention*. Soft attention distributes real-valued attention values over the input, making end-to-end training with gradient descent possible. Hard attention mechanisms make discrete choices about which parts of the input to focus on, and can be trained with reinforcement learning (Mnih et al., 2014). In NLP, soft attention can mitigate the difficulty of compressing long sequences into fixed-dimensional vectors, with applications in machine translation (Bahdanau et al., 2015) and question answering (Hermann et al., 2015). In computer vision, both types of attention can be used for selecting regions in an image (Ba et al., 2015; Xu et al., 2015).

3 The NEAT Reading Model

The point of departure for our model is the Tradeoff Hypothesis (see Section 1): Reading optimizes a tradeoff between precision of language understand-

ing and economy of attention. We make this idea explicit by proposing NEAT (NEural Attention Tradeoff), a model that reads text and attempts to reconstruct it afterwards. While reading, the network chooses which words to process and which words to skip. The Tradeoff Hypothesis is formalized using a training objective that combines accuracy of reconstruction with economy of attention, encouraging the network to only look at words to the extent that is necessary for reconstructing the sentence.

3.1 Architecture

We use a neural sequence-to-sequence architecture (Sutskever et al., 2014) with a hard attention mechanism. We illustrate the model in Figure 1, operating on a three-word sequence $\boldsymbol{w}$. The most basic components are the *reader*, labeled R, and the *decoder*. Both of them are recurrent neural networks with Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber, 1997) units. The recurrent reader network is expanded into time steps $R_0, \ldots, R_3$ in the figure. It goes over the input sequence, reading one word w_i at a time, and converts the word sequence into a sequence of vectors $h_0, \ldots, h_3$. Each vector h_i acts as a fixed-dimensionality encoding of the word sequence $w_1, \ldots, w_i$ that has been read so far. The last vector h_3 (more generally h_N for sequence length N), which encodes the entire input sequence, is then fed into the input layer of the decoder network, which attempts to reconstruct the input sequence $\boldsymbol{w}$. It is also realized as a recurrent neural network, collapsed into a single box in the figure. It models a probability distribution over word sequences, outputting a probability distribution $P_{Decoder}(w_i|\boldsymbol{w}_{1,\ldots,i-1}, h_N)$ over the vocabulary in the i-th step, as is common in neural language modeling (Mikolov et al., 2010). As the decoder has access to the vector representation created by the reader network, it ideally is able to assign the highest probability to the word sequence $\boldsymbol{w}$ that was actually read. Up to this point, the model is a standard sequence-to-sequence architecture reconstructing the input sequence, that is, performing autoencoding.

As a basic model of human processing, NEAT contains two further components. First, experimental evidence shows that during reading, humans constantly make predictions about the upcoming input (e.g., Van Gompel and Pickering, 2007). As a model of this behavior, the reader network at each time step outputs a probability distribution P_R over the lexicon. This distribution describes which words are likely to come next (i.e., the reader network performs language modeling). Unlike the modeling performed by the decoder, P_R, via its recurrent connections, has access to the previous context only.

Second, we model skipping by stipulating that only some of the input words w_i are fed into the reader network R, while R receives a special vector representation, containing no information about the input word, in other cases. These are the words that are skipped. In NEAT, at each time step during reading, the *attention module* A decides whether the next word is shown to the reader network or not. When humans skip a word, they are able to identify it using *parafoveal preview* (Rayner, 2009). Thus, we can assume that the choice of which words to skip takes into account not only the prior context but also a preview of the word itself. We therefore allow the attention module to take the input word into account when making its decision. In addition, the attention module has access to the previous state h_{i-1} of the reader network, which summarizes what has been read so far. To allow for interaction between skipping and prediction, we also give the attention module access to the probability of the input word according to the prediction P_R made at the last time step. If we write the decision made by A as $\omega_i \in \{0,1\}$, where $\omega_i = 1$ means that word w_i is shown to the reader and 0 means that it is not, we can write the probability of showing word w_i as:

$$\begin{aligned} &P(\omega_i = 1|\boldsymbol{\omega}_{1\ldots i-1}, \boldsymbol{w}) \\ &= P_A(w_i, h_{i-1}, P_R(w_i|\boldsymbol{w}_{1\ldots i-1}, \boldsymbol{\omega}_{1\ldots i-1})) \end{aligned} \quad (1)$$

We implement A as a feed-forward network, followed by taking a binary sample ω_i.

We obtain the *surprisal* of an input word by taking the negative logarithm of the conditional probability of this word given the context words that precede it:

$$\mathrm{Surp}(w_i|\boldsymbol{w}_{1\ldots i-1}) = -\log P_R(w_i|\boldsymbol{w}_{1\ldots i-1}, \boldsymbol{\omega}_{1\ldots i-1}) \quad (2)$$

As a consequence of skipping, not all input words are accessible to the reader network. Therefore, the

probability and surprisal estimates it computes crucially only take into account the words that have actually been fixated. We will refer to this quantity as the *restricted surprisal*, as opposed to *full surprisal*, which is computed based on all prior context words.

The key quantities for predicting human reading are the fixation probabilities in equation (1), which model fixations and skips, and restricted surprisal in equation (2), which models the reading times of the words that are fixated.

3.2 Model Objective

Given network parameters θ and a sequence $\boldsymbol{w}$ of words, the network stochastically chooses a sequence $\boldsymbol{\omega}$ according to (1) and incurs a loss $L(\boldsymbol{\omega}|\boldsymbol{w},\theta)$ for language modeling and reconstruction:

$$\begin{aligned} L(\boldsymbol{\omega}|\boldsymbol{w},\theta) = &-\sum_i \log P_R(w_i|\boldsymbol{w}_{1,\ldots,i-1},\boldsymbol{\omega}_{1,\ldots,i-1};\theta) \\ &-\sum_i \log P_{Decoder}(w_i|\boldsymbol{w}_{1,\ldots,i-1};h_N;\theta) \end{aligned} \tag{3}$$

where $P_R(w_i,\ldots)$ denotes the output of the reader after reading w_{i-1}, and $P_{Decoder}(w_i|\ldots;h_N)$ is the output of the decoder at time $i-1$, with h_N being the vector representation created by the reader network for the entire input sequence.

To implement the Tradeoff Hypothesis, we train NEAT to solve language modeling and reconstruction with minimal attention, i.e., the network minimizes the expected loss:

$$Q(\theta) := \mathrm{E}_{\boldsymbol{w},\boldsymbol{\omega}}\left[L(\boldsymbol{\omega}|\boldsymbol{w},\theta) + \alpha\cdot\|\boldsymbol{\omega}\|_{\ell_1}\right] \tag{4}$$

where word sequences $\boldsymbol{w}$ are drawn from a corpus, and $\boldsymbol{\omega}$ is distributed according to $P(\boldsymbol{\omega}|\boldsymbol{w},\theta)$ as defined in (1). In (4), $\|\boldsymbol{\omega}\|_{\ell_1}$ is the number of words shown to the reader, and $\alpha > 0$ is a hyperparameter. The term $\alpha\cdot\|\boldsymbol{\omega}\|_{\ell_1}$ encourages NEAT to attend to as few words as possible.

Note that we make no assumption about linguistic structure – the only ingredients of NEAT are the neural architecture, the objective (4), and the corpus from which the sequences $\boldsymbol{w}$ are drawn.

3.3 Training

We follow previous approaches to hard attention in using a combination of gradient descent and reinforcement learning, and separate the training of the recurrent networks from the training of A. To train the reader R and the decoder, we temporarily remove the attention network A, set $\boldsymbol{\omega} \sim \mathrm{Binom}(n,p)$ (n sequence length, p a hyperparameter), and minimize $\mathrm{E}[L(\boldsymbol{w}|\theta,\boldsymbol{\omega})]$ using stochastic gradient descent, sampling a sequence $\boldsymbol{\omega}$ for each input sequence. In effect, NEAT is trained to perform reconstruction and language modeling when there is noise in the input. After R and the decoder have been trained, we fix their parameters and train A using the REINFORCE rule (Williams, 1992), which performs stochastic gradient descent using the estimate

$$\frac{1}{|B|}\sum_{\boldsymbol{w}\in B;\boldsymbol{\omega}} \left(L(\boldsymbol{\omega}|\boldsymbol{w},\theta) + \alpha\cdot\|\boldsymbol{\omega}\|_{\ell_1}\right)\partial_{\theta_A}\left(\log P(\boldsymbol{\omega}|\boldsymbol{w},\theta)\right) \tag{5}$$

for the gradient $\partial_{\theta_A} Q$. Here, B is a minibatch, $\boldsymbol{\omega}$ is sampled from $P(\boldsymbol{\omega}|\boldsymbol{w},\theta)$, and $\theta_A \subset \theta$ is the set of parameters of A. For reducing the variance of this estimator, we subtract in the i-th step an estimate of the expected loss:

$$\begin{aligned} U(\boldsymbol{w},\boldsymbol{\omega}_{1\ldots i-1}) := \mathrm{E}_{\boldsymbol{\omega}_{i\ldots N}}[&L(\boldsymbol{\omega}_{1\ldots i-1}\boldsymbol{\omega}_{i\ldots N}|\boldsymbol{w},\theta) \\ &+ \alpha\cdot\|\boldsymbol{\omega}\|_{\ell_1}] \end{aligned} \tag{6}$$

We compute the expected loss using an LSTM that we train simultaneously with A to predict $L + \alpha\|\boldsymbol{\omega}\|_{\ell_1}$ based on $\boldsymbol{w}$ and $\boldsymbol{\omega}_{1\ldots i-1}$. To make learning more stable, we add an entropy term encouraging the distribution to be smooth, following Xu et al. (2015). The parameter updates to A are thus:

$$\begin{aligned} \sum_{\boldsymbol{w},\boldsymbol{\omega}}\sum_i &\left(L(\boldsymbol{\omega}|\boldsymbol{w},\theta) + \alpha\|\boldsymbol{\omega}\|_{\ell_1} - U(\boldsymbol{w},\boldsymbol{\omega}_{1\ldots i-1})\right) \\ &\cdot \partial_{\theta_A}\left(log\ P(\omega_i|\boldsymbol{\omega}_{1\ldots i-1},\boldsymbol{w},\theta)\right) \\ -\gamma\,\partial_{\theta_A}&\left(\sum_{\boldsymbol{w},\boldsymbol{\omega}}\sum_i \mathrm{H}[P(\omega_i|\boldsymbol{\omega}_{1,\ldots,i-1},\boldsymbol{w},\theta)]\right) \end{aligned} \tag{7}$$

where γ is a hyperparameter, and H the entropy.

4 Methods

Our aim is to evaluate how well NEAT predicts human fixation behavior and reading times. Furthermore, we want show that known qualitative properties emerge from the Tradeoff Hypothesis, even though no prior knowledge about useful features is hard-wired in NEAT.

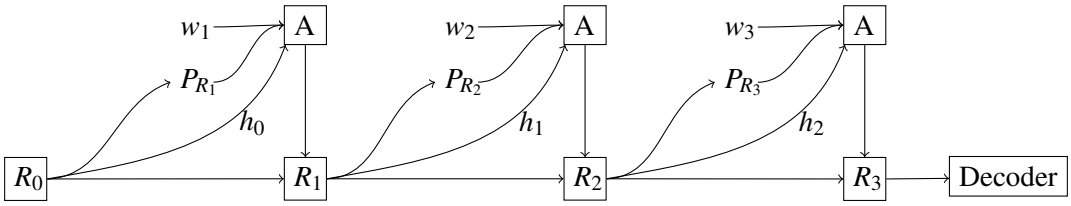


Figure 1: The architecture of the proposed model, reading a three-word input sequence w_1, w_2, w_3. R is the reader network and P_R the probability distribution it computes in each time step. A is the attention network. At each time step, the input, its probability according to P_R, and the previous state h_{i-1} of R are fed into A, which then decides whether the word is read or skipped.

4.1 Training Setup

For both the reader and the decoder networks, we choose a one-layer LSTM network with 1,000 memory cells. The attention network is a one-layer feed-forward network. For the loss estimator U, we use a bidirectional LSTM with 20 memory cells. Input data is split into sequences of 50 tokens, which are used as the input sequences for NEAT, disregarding sentence boundaries. Word embeddings have 100 dimensions, are shared between the reader and the attention network, and are only trained during the training of the reader. The vocabulary consists of the 10,000 most frequent words from the training corpus. We trained NEAT on the training set of the *Daily Mail* section of the corpus described by Hermann et al. (2015), which consists of 195,462 articles from the *Daily Mail* newspaper, containing approximately 200 million tokens. The recurrent networks and the attention network were each trained for one epoch. For initialization, weights are drawn from the uniform distribution. We set $\alpha = 5.0$, $\gamma = 5.0$, and used a constant learning rate of 0.01 for A.

4.2 Corpus

For evaluation, we use the English section of the Dundee corpus (Kennedy and Pynte, 2005), which consists of 20 texts from *The Independent*, annotated with eye-movement data from ten English native speakers. Each native speakers read all 20 texts and answered a comprehension question after each text. We split the Dundee corpus into a development and a test set, with texts 1–3 constituting the development set. The development set consists of 78,300 tokens, and the test set of 281,911 tokens. For evaluation, we removed the datapoints removed by Demberg and Keller (2008), mainly consisting of words at the beginning or end of lines, outliers, and cases of track loss. Furthermore, we removed datapoints where the word was outside of the vocabulary of the model, and those datapoints mapped to positions 1–3 or 48–50 of a sequence when splitting the data. After preprocessing, 62.9% of the development tokens and 64.7% of the test tokens remained. To obtain the number of fixations on a token and reading times, we used the eye-tracking measures computed by Demberg and Keller (2008). The overall fixation rate was 62.1% on the development set, and 61.3% on the test set.

The development set was used to run preliminary versions of the human evaluation studies, and to determine the human skipping rate (see Section 5). All the results reported in this paper were computed on the test set, which remained unseen until the model was final.

5 Results and Discussion

Throughout this section, we consider the following baselines for the attention network: *random attention* is defined by $\boldsymbol{\omega} \sim \text{Binom}(n, p)$, with $p = 0.62$, the human fixation rate in the development set. For *full attention*, we take $\boldsymbol{\omega} = 1$, i.e., all words are fixated. We also derive fixation predictions from *full surprisal*, *word frequency*, and *word length* by choosing a threshold such that the resulting fixation rate matches the human fixation rate on the development set.

5.1 Quantitative Properties

By averaging over all possible fixation sequences, NEAT defines for each word in a sequence a probability that it will be fixated. This probability is not efficiently computable, so we approximate it by sampling a sequence $\boldsymbol{\omega}$ and taking the probabilities $P(\omega_i = 1 | \omega_{1\ldots i-1}, \boldsymbol{w})$ for $i = 1, \ldots, 50$. These sim-

ulated fixation probabilities can be interpreted as defining a distribution of attention over the input sequence. Figure 2 shows heatmaps of the simulated and human fixation probabilities, respectively, for the beginning of a text from the Dundee corpus. While some differences between simulated and human fixation probabilities can be noticed, there are similarities in the general qualitative features of the two heatmaps. In particular, function words and short words are less likely to be fixated than content words and longer words in both the simulated and the human data.

Reconstruction and Language Modeling We first evaluate NEAT intrinsically by measuring how successful the network is at predicting the next word and reconstructing the input while minimizing the number of fixations. We compare perplexity on reconstruction and language modeling for $\boldsymbol{\omega} \sim P(\boldsymbol{\omega}|\boldsymbol{w}, \theta)$. In addition to the baselines, we run NEAT on the fixations generated by the human readers of the Dundee corpus, i.e., we use the human fixation sequence as $\boldsymbol{\omega}$ instead of the fixation sequence generated by A to compute perplexity. This will tell us to what extent the human behavior minimizes the NEAT objective (4).

The results are given in Table 1. In all settings, the fixation rates are similar (60.4% to 62.1%) which makes the perplexity figures directly comparable. While NEAT has a higher perplexity on both tasks compared to full attention, it considerably outperforms random attention. It also outperforms the word length, word frequency, and full surprisal baselines. The perplexity on human fixation sequences is similar to that achieved using word frequency. Based on these results, we conclude that REINFORCE successfully optimizes the objective (4).

Likelihood of Fixation Data Human reading behavior is stochastic in the sense that different runs of eye-tracking experiments such as the ones recorded in the Dundee corpus yield different eye-movement sequences. NEAT is also stochastic, in the sense that, given a word sequence $\mathbf{w}$, it defines a probability distribution over fixation sequences $\boldsymbol{\omega}$. Ideally, this distribution should be close to the actual distribution of fixation sequences produced by humans reading the sequence, as measured by perplexity.

We find that the perplexity of the fixation sequences produced by the ten readers in the Dundee corpus under NEAT is 1.84. A perplexity of 2.0 corresponds to the random baseline $\text{Binom}(n, 0.5)$, and a perplexity of 1.96 to random attention $\text{Binom}(n, 0.62)$. As a lower bound on what can achieved with models disregarding the context, using the human fixation rates for each word as probabilities, we obtain a perplexity of 1.68.

	Acc	$F1_{fix}$	$F1_{skip}$
NEAT	63.7	70.4	53.0
Supervised Models			
Nilsson and Nivre (2009)	69.5	75.2	62.6
Matthies and Søgaard (2013)	69.9	72.3	66.1
Human Performance and Baselines			
Random Baseline	52.6	62.1	37.9
Full Surprisal	64.1	70.7	53.6
Word Frequency	67.9	74.0	58.3
Word Length	68.4	77.1	49.0
Human	69.5	76.6	53.6

Table 2: Evaluation of fixation sequence predictions against human data. For the human baseline, we predicted the n-th reader's fixations by taking the fixations of the $n+1$-th reader (with missing values replaced by reader average), averaging the resulting scores over the ten readers.

Accuracy of Fixation Sequences Previous work on supervised models for modeling fixations (Nilsson and Nivre, 2009; Matthies and Søgaard, 2013) has been evaluated by measuring the overlap of the fixation sequences produced by the models with those in the Dundee corpus. For NEAT, this method of evaluation is problematic as differences between model predictions and human data may be due to differences in the rate of skipping, and due to the inherently stochastic nature of fixations. We therefore derive model predictions by rescaling the simulated fixation probabilities so that their average equals the fixation rate in the development set, and then greedily take the maximum-likelihood sequence. That is, we predict a fixation if the rescaled probability is greater than 0.5, and a skip otherwise. As in previous work, we report the accuracy of fixations and skips, and also separate F1 scores for fixations and skips. As lower and upper bounds, we use the random baseline $\boldsymbol{\omega} \sim \text{Binom}(n, 0.62)$ and the agreement of the ten human readers, respectively. The re-

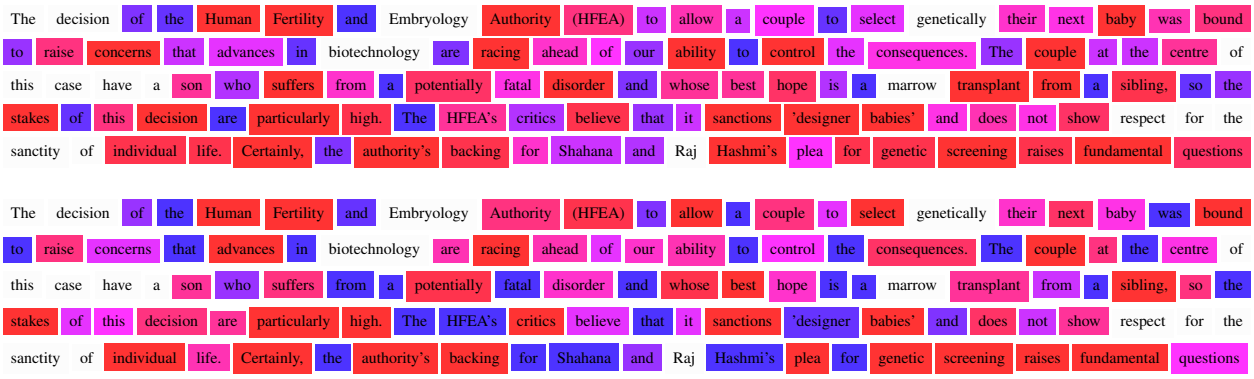


Figure 2: Top: Heatmap showing human fixation probabilities, as estimated from the ten readers in the Dundee corpus. In cases of track loss, we replaced the missing value with the corresponding reader's overall fixation rate. Bottom: Heatmap showing fixation probabilities simulated by NEAT. Color gradient ranges from blue (low probability) to red (high probability); words without color are at the beginning or end of a sequence, or out of vocabulary.

	NEAT	Rand. Att.	Word Len.	Word Freq.	Full Surp.	Human	Full Att.
Language Modeling	180	333	230	219	211	218/170	107
Reconstruction	4.5	56	40	39	34	39/31	1.6
Fixation Rate	60.4%	62.1%	62.1%	62.1%	62.1%	61.3%/72.0%	100%

Table 1: Performance on language modeling and reconstruction as measured by perplexity. Random attention is an upper bound on perplexity, while full attention is a lower bound. For the human baseline, we give two figures, which differ in the treatment of missing data. The first figure is obtained when replacing missing values with a random variable $\omega \sim \text{Binom}(n, 0.61)$; the second results from replacing missing values with 1.

sults are shown in Table 2. NEAT clearly outperforms the random baseline and shows results close to full surprisal (where we apply the same rescaling and thresholding as for NEAT). This is remarkable given that NEAT has access to only 60.4% of the words in the corpus in order to predict skipping, while full surprisal has access to all the words.

Word frequency and word length perform well, almost reaching the performance of supervised models. This shows that the bulk of skipping behavior is already explained by word frequency and word length effects. Note, however, that NEAT is completely unsupervised, and does not know that it has to pay attention to word frequency; this is something the model is able to infer.

Restricted Surprisal and Reading Times To evaluate the predictions NEAT makes for reading times, we use linear mixed-effects models containing restricted surprisal derived from NEAT for the Dundee test set. The mixed models also include a set of standard baseline predictors, viz., word length, log word frequency, log frequency of the previous word, launch distance, landing position, and the position of the word in the sentence. We treat participants and items as random factors. As the dependent variable, we take first pass duration, which is the sum of the durations of all fixations from first entering the word to first leaving it. We compare against *full surprisal* as an upper bound and against *random surprisal* as a lower bound. Random surprisal is surprisal computed by a model with random attention; this allows us to assess how much surprisal degrades when only 60.4% of all words are fixated, but no information is available as to which words should be fixated. The results in Table 3 show that restricted surprisal as computed by NEAT, full surprisal, and random surprisal are all significant predictors of reading time.

In order to compare the three surprisal estimates, we therefore need a measure of effect size. For this, we compare the model fit of the three mixed effects models using deviance, which is defined as the difference between the log likelihood of the model under consideration minus the log likelihood of the baseline model, multiplied by -2. Higher de-

	β	SE	*t*
(Intercept)	247.43	7.14	34.68*
Word Length	12.92	0.21	60.62*
Previous Word Freq.	−5.28	0.28	−18.34*
Prev. Word Fixated	−24.67	0.81	−30.55*
Launch Distance	-0.01	0.01	−0.37
Obj. Landing Pos.	−8.07	0.20	−41.25*
Word Pos. in Sent.	−0.10	0.03	−2.98*
Log Word Freq.	−1.59	0.21	−7.73*
Resid. Random Surprisal	2.69	0.10	29.27*
Resid. Restr. Surprisal	2.75	0.12	23.66*
Resid. Full Surprisal	2.99	0.12	25.23*

Table 3: Linear mixed effects models for first pass duration. The first part of the table shows the coefficients, standard errors, and *t* values for the predictors in the baseline model. The second part of the table gives the corresponding values for random surprisal, restricted surprisal computed by NEAT, and full surprisal, residualized against the baseline predictors, in three models obtained by adding these predictors.

	Human	NEAT	Word Freq.
ADJ	78.9 (2)	72.8 (1)	98.4 (3)
ADP	46.1 (8)	53.8 (8)	21.6 (9)
ADV	70.4 (3)	67.2 (4)	96.4 (4)
CONJ	36.7 (11)	50.7 (9)	14.6 (10)
DET	45.2 (9)	44.8 (11)	22.9 (8)
NOUN	80.3 (1)	69.8 (2)	98.7 (2)
NUM	63.3 (6)	71.5 (3)	99.5 (1)
PRON	49.2 (7)	57.0 (7)	42.6 (7)
PRT	37.4 (10)	46.7 (10)	13.9 (11)
VERB	66.7 (5)	64.7 (5)	74.4 (5)
X	68.6 (4)	67.8 (3)	69.0 (6)
Spearman's ρ		0.85	0.84
Pearson's *r*		0.92	0.94
MSE		57	450

Table 4: Actual and simulated fixation probabilities (in %) by PoS tag, with the ranks given in brackets, and correlations and mean squared error relative to human data.

viance indicates greater improvement in model fit over the baseline model. We find that the mixed model that includes restricted surprisal achieves a deviance of 867, compared to the model containing only the baseline features. With full surprisal, we obtain a deviance of 980. On the other hand, the model including random surprisal achieves a lower deviance of 832.

This shows that restricted surprisal as computed by NEAT not only significantly predicts reading times, it also provides an improvement in model fit compared to the baseline predictors. Such an improvement is also observed with random surprisal, but restricted surprisal achieves a greater improvement in model fit. Full surprisal achieves an even greater improvement, but this is not unexpected, as full surprisal has access to all words, unlike NEAT or random surprisal, which only have access to 60.4% of the words.

5.2 Qualitative Properties

We now examine the second key question we defined in Section 1, investigating the qualitative features of the simulated fixation sequences. We will focus on comparing the predictions of NEAT with that of word frequency, which performs comparably at the task of predicting fixation sequences (see Section 5.1). We show NEAT nevertheless makes relevant predictions that go beyond frequency.

Fixations of Successive Words While predictors derived from word frequency treat the decision whether to fixate or skip words as independent, humans are more likely to fixate a word when the previous word was skipped (Rayner, 1998). This effect is also seen in NEAT. More precisely, both in the human data and in the simulated fixation data, the conditional fixation probability $P(\omega_i = 1|\omega_{i-1} = 1)$ is lower than the marginal probability $P(\omega_i = 1)$. The ratio of these probabilities is 0.85 in the human data, and 0.81 in NEAT. The threshold predictor derived from word frequency also shows this effect (as the frequencies of successive words are not independent), but it is weaker (ratio 0.91).

To further test the context dependence of NEAT's fixation behavior, we ran a mixed model predicting the fixation probabilities simulated by NEAT, with items as random factor and the log frequency of word i as predictor. Adding ω_{i-1} as a predictor results in a significant improvement in model fit (deviance = 4,798, $t = 71.3$). This shows that NEAT captures the context dependence of fixation sequences to an extend that goes beyond word frequency alone.

Parts of Speech Part of speech categories are known to be a predictor of fixation probabilities, with content words being more likely to be fixated than function words (Carpenter and Just, 1983). In Table 4, we give the simulated fixation probabilities and the human fixation probabilities estimated from the Dundee corpus for the tags of the Universal PoS tagset (Petrov et al., 2012), using the PoS annotation of Barrett et al. (2015). We again compare with the probabilities of a threshold predictor derived from word frequency.[1] NEAT captures the differences between PoS categories well, as evidenced by the high correlation coefficients. The content word categories ADJ, ADV, NOUN, VERB and X consistently show higher probabilities than the function word categories. While the correlation coefficients for word frequency are very similar, the numerical values of the simulated probabilities are closer to the human ones than those derived from word frequency, which tend towards more extreme values. This difference can be seen clearly if we compare the mean squared error, rather than the correlation, with the human fixation probabilities (last row of Table 4).

Correlations with Known Predictors In the literature, it has been observed that skipping correlates with predictability (surprisal), word frequency, and word length (Rayner, 1998, p. 387). These correlations are also observed in the human skipping data derived from Dundee, as shown in Table 5. (Human fixation probabilities were obtained by averaging over the ten readers in Dundee.)

Comparing the known predictors of skipping with NEAT's simulated fixation probabilities, similar correlations as in the human data are observed. We observe that the correlations with surprisal are stronger in NEAT, considering both restricted surprisal and full surprisal as measures of predictability.

6 Conclusions

We investigated the hypothesis that human reading strategies optimize a tradeoff between precision of language understanding and economy of attention. We made this idea explicit in NEAT, a neural reading architecture with hard attention that can be trained end-to-end to optimize this tradeoff. Experiments on the Dundee corpus show that NEAT provides accurate predictions for human skipping behavior. It also predicts reading times, even though it only has access to 60.4% of the words in the corpus in order to estimate surprisal. Finally, we found that known qualitative properties of skipping emerge in our model, even though they were not explicitly included in the architecture, such as context dependence of fixations, differential skipping rates across parts of speech, and correlations with other known predictors of human reading behavior.

[1] We omit the tag "." for punctuation, as punctuation characters are not treated as separate tokens in Dundee.

	Human	NEAT
Restricted Surprisal	0.465	0.762
Full Surprisal	0.512	0.720
Log Word Freq.	−0.608	−0.760
Word Length	0.663	0.521

Table 5: Correlations between human and NEAT fixation probabilities and known predictors

References

Ba, Jimmy, Ruslan R. Salakhutdinov, Roger B. Grosse, and Brendan J. Frey. 2015. Learning wake-sleep recurrent attention models. In *Advances in Neural Information Processing Systems*. pages 2575–2583.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Barrett, Maria, Željko Agić, and Anders Søgaard. 2015. The Dundee treebank. In *Proceedings of the 14th International Workshop on Treebanks and Linguistic Theories*. pages 242–248.

Bicknell, Klinton and Roger Levy. 2010. A rational model of eye movement control in reading. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. pages 1168–1178.

Carpenter, P. A. and M. A. Just. 1983. What your eyes do while your mind is reading. In K. Rayner, editor, *Eye Movements in Reading*, Academic Press, New York, pages 275–307.

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

Engbert, Ralf, André Longtin, and Reinhold Kliegl. 2002. A dynamical model of saccade generation in reading based on spatially distributed lexical processing. *Vision Research* 42(5):621–636.

Engbert, Ralf, Antje Nuthmann, Eike M. Richter, and Reinhold Kliegl. 2005. SWIFT: A dynamical model of saccade generation during reading. *Psychological Review* 112(4):777–813.

Frank, S.L. and R. Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science* 22:829–834.

Hale, John. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*. volume 2, pages 159–166.

Hara, Tadayoshi, Daichi Mochihashi Yoshinobu Kano, and Akiko Aizawa. 2012. Predicting word fixations in text with a CRF model for capturing general reading strategies among readers. In *Proceedings of the 1st Workshop on Eye-tracking and Natural Language Processing*. pages 55–70.

Henderson, John. 2003. Human gaze control in real-world scene perception. *Trends in Cognitive Sciences* 7:498–504.

Hermann, Karl Moritz, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. ArXiv:1506.03340.

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

Kennedy, Alan and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research* 45(2):153–168.

Levy, Roger. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3):1126–1177.

Matthies, Franz and Anders Søgaard. 2013. With blinkers on: Robust prediction of eye movements across readers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 803–807.

McDonald, Scott A. and Richard C. Shillcock. 2003a. Eye movements reveal the on-line computation of lexical probabilities during reading. *Psychological Science* 14(6):648–652.

McDonald, Scott A. and Richard C. Shillcock. 2003b. Low-level predictive inference in reading: the influence of transitional probabilities on eye movements. *Vision Research* 43(16):1735–1751.

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*. pages 1045–1048.

Mnih, Volodymyr, Nicolas Heess, Alex Graves, and others. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. pages 2204–2212.

Nilsson, Mattias and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in reading. In *Proceedings of the 13th Conference on Computational Natural Language Learning*. pages 93–101.

Nilsson, Mattias and Joakim Nivre. 2010. Towards a data-driven model of eye movement control in reading. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*. pages 63–71.

Petrov, Slav, Dipanjan Das, and Ryan T. McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. pages 2089–2096.

Rayner, K. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124(3):372–422.

Rayner, Keith. 2009. Eye movements in reading: Models and data. *Journal of Eye Movement Research* 2(5):1–10.

Rayner, Keith and Erik D. Reichle. 2010. Models of the reading process. *Wiley Interdisciplinary Reviews: Cognitive Science* 1(6):787–799.

Reichle, E. D., A. Pollatsek, D. L. Fisher, and K. Rayner. 1998. Toward a model of eye move-

ment control in reading. *Psychological Review* 105(1):125–157.

Reichle, E. D., T. Warren, and K. McConnell. 2009. Using EZ Reader to model the effects of higher level language processing on eye movements during reading. *Psychonomic Bulletin & Review* 16(1):1–21.

Reichle, Erik D., Keith Rayner, and Alexander Pollatsek. 2003. The EZ Reader model of eye-movement control in reading: Comparisons to other models. *Behavioral and Brain Sciences* 26(04):445–476.

Smith, Nathaniel J. and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition* 128(3):302–319.

Sutskever, Ilya, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Van Gompel, Roger PG and Martin J. Pickering. 2007. Syntactic parsing. In *The Oxford Handbook of Psycholinguistics*, Oxford University Press, pages 289–307.

Williams, Ronald J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. ArXiv:1502.03044.

Comparing Computational Cognitive Models of Generalization in a Language Acquisition Task

Libby Barak, Adele E. Goldberg,
Psychology Department
Princeton University
Princeton, NJ, USA
{lbarak,adele}@princeton.edu

Suzanne Stevenson
Department of Computer Science
University of Toronto
Toronto, Canada
suzanne@cs.toronto.edu

Abstract

Natural language acquisition relies on appropriate generalization: the ability to produce novel sentences, while learning to restrict productions to acceptable forms in the language. Psycholinguists have proposed various properties that might play a role in guiding appropriate generalizations, looking at learning of verb alternations as a testbed. Several computational cognitive models have explored aspects of this phenomenon, but their results are hard to compare given the high variability in the linguistic properties represented in their input. In this paper, we directly compare two recent approaches, a Bayesian model and a connectionist model, in their ability to replicate human judgments of appropriate generalizations. We find that the Bayesian model more accurately mimics the judgments due to its richer learning mechanism that can exploit distributional properties of the input in a manner consistent with human behaviour.

1 Introduction

Native speakers of a language are mostly able to generalize appropriately beyond the observed data while avoiding overgeneralizations. A testbed area for studying generalization behavior in language acquisition is verb alternations – i.e., learning the patterns of acceptability of alternative constructions for expressing similar meanings. For example, English speakers readily use a new verb like *text* in both the double-object (DO) construction ("text me the details") and the prepositional-dative (PD) ("text the details to me") – an instance of the dative alternation. However, speakers avoid overgeneralizing the DO construction to verbs such as *explain* that resist its use ("?explain me the details"), even though they occur with analogous arguments in the PD alternative ("explain the details to me"). Psycholinguistic studies have focused on the possible properties of natural language that enable such generalization while constraining it to acceptable forms.

Initially, children are linguistically conservative: they generally use verbs in constructions that are very close to exemplars in the input (Lieven et al., 1997; Akhtar, 1999; Tomasello, 2003; Boyd and Goldberg, 2009). Children reach adult-like competence by gradually forming more general associations of constructions to meaning that allow them to extend verb usages to unwitnessed forms. Much work has emphasized the role of verb classes that capture the regularities across semantically-similar verbs, enabling appropriate generalization (e.g., Pinker, 1989; Fisher, 1999; Levin, 1993; Ambridge et al., 2008). Usage-based approaches have argued that such class-based behaviour can arise in learning through the clustering of observed usages that share semantic and syntactic properties (e.g., Bybee, 2010; Tomasello, 2003; Goldberg, 2006).

A number of studies also reveal that the statistical properties of the language play a central role in limiting generalization (e.g., Bresnan and Ford, 2010; Ambridge et al., 2012, 2014). Individual verbs often show statistical biases that favor their appearance in one construction over another (Ford et al., 1982; MacDonald et al., 1994; Garnsey et al., 1997; Trueswell et al., 1993; Losiewicz, 1992; Gahl and Garnsey, 2004). For example, while both *give* and *push* can occur in either DO or PD constructions,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 96–106,
Austin, Texas, November 1-5, 2016.

give strongly favors the DO construction ("give me the box"), while *push* strongly favors the PD ("push the box to me") (Wasow, 2002). Generally, the more frequent a verb is overall, the less likely speakers are to extend it to an unobserved construction (Braine and Brooks, 1995). In addition, when a verb repeatedly occurs in one construction when an alternative construction could have been appropriate, speakers appear to learn that the verb is inappropriate in the alternative, regardless of its overall frequency (Goldberg, 2011).

Given these observations, it has been argued that both the semantic and statistical properties of a verb underlie its degree of acceptability in alternating constructions (e.g., Braine and Brooks, 1995; Theakston, 2004; Ambridge et al., 2014). Recently, Ambridge and Blything (2015) propose a computational model designed to study the role of verb semantics and frequency in the acquisition of the dative alternation. However, they only evaluate their model preferences for one of the two constructions, which does not provide a full picture of the alternation behaviour; moreover, they incorporate certain assumptions about the input that may not match the properties of naturalistic data.

In this paper, we compare the model of Ambridge and Blything (2015) to the Bayesian model of Barak et al. (2014) that offers a general framework of verb construction learning. We replicate the approach taken in Ambridge and Blything (2015) in order to provide appropriate comparisons, but we also extend the experimental settings and analysis to enable a more fulsome evaluation, on data with more naturalistic statistical properties. Our results show that the Bayesian model provides a better fit to the psycholinguistic data, which we suggest is due to its richer learning mechanism: its two-level clustering approach can exploit distributional properties of the input in a manner consistent with human generalization behaviour.

2 Related Work

Acquisition of the dative alternation – use of the DO and PD constructions with analogous semantic arguments – has been studied in several computational cognitive models because it illustrates how people learn to appropriately generalize linguistic constructions in the face of complex, interacting factors. As noted by Ambridge et al. (2014), such models should capture influences of the verb such as its semantic properties, its overall frequency, and its frequency in various constructions.

A focus of computational models has been to show under what conditions a learner generalizes to the DO construction having observed a verb in the PD, and vice versa. For example, the hierarchical Bayesian models of Perfors et al. (2010) and Parisien and Stevenson (2010) show the ability to generalize from one construction to the other. However, both models are limited in their semantic representations. Perfors et al. (2010) use semantic properties that directly (albeit noisily) encode the knowledge of the alternating and non-alternating (DO-only or PD-only) classes. The model of Parisien and Stevenson (2010) addresses this limitation by learning alternation classes from the data (including the dative), but it uses only syntactic slot features that can be gleaned automatically from a corpus. In addition, both models use batch processing, failing to address how learning to generalize across an alternation might be achieved incrementally.

Alishahi and Stevenson (2008) presents an incremental Bayesian model shown to capture various aspects of verb argument structure acquisition (Alishahi and Pyykkoïen, 2011; Barak et al., 2012, 2013b; Matusevych et al., 2016), but the model is unable to mimic alternation learning behaviour. Barak et al. (2014) extends this construction-learning model to incrementally learn both constructions and classes of alternating verbs, and show the role of the classes in learning the dative. However, like Parisien and Stevenson (2010), the input to the model in this study is limited to syntactic properties, not allowing for a full analysis of the relevant factors that influence acquisition of alternations.

Ambridge and Blything (2015) propose the first computational model of this phenomenon to include a rich representation of the verb/construction semantics, drawn from human judgments. In evaluation, however, they only report the ability of the model to predict the DO usage (i.e., only one pair of the alternation), which does not give the full picture of the alternation behaviour. Moreover, their assumptions about the nature of the input – including the use of raw vs. log frequencies and the treatment of

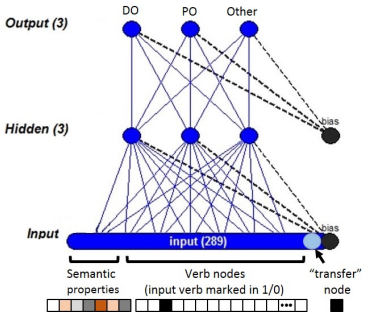


Figure 1: A visual representation of the feed-forward network used by the AB model. (The figure is adapted from output of the OXlearn package of Ruh and Westermann (2009).) The input nodes correspond to the semantic properties of the verbs, the verb lexemes, and a "transfer" node (explained in the text). The output nodes correspond to the target constructions.

non-dative construction usages – differ from earlier models, making it difficult to compare the results.

In this paper, we compare the models of Ambridge and Blything (2015) and Barak et al. (2014), using the same input settings for each, so that, for the first time, two computational models of this generalization phenomenon can be directly compared. Moreover, in contrast to Ambridge and Blything (2015) and in line with the other studies mentioned above, we evaluate the ability of the models to generate both the DO and the PD alternates, on a per verb basis, in order to more accurately assess the fit to human judgments.

3 The Computational Models

In this section, we give an overview of the connectionist model of Ambridge and Blything (2015), hereafter the AB model, and the Bayesian model of Barak et al. (2014), hereafter the BFS model, followed by a comparison of their relevant properties.

3.1 Overview of the Connectionist Model

The AB connectionist model of Ambridge and Blything (2015) aims to predict the preference of a verb for each of three target constructions, on the basis of verb semantics and the observed distribution of verbs in those constructions in the input. Figure 1 provides an illustration of the 3-layer feed-forward network, trained using backpropagation. Each input to the model consists of lexical and semantic features of a verb and its usage. The target output is a 1-hot pattern across output nodes, each of which represents the use of the verb in the associated construction. The possible constructions are DO, PD, or *other*, representing all other constructions the verb appears in. Training presents the slate of input features with the appropriate output node activated representing the construction the verb appears in. In a full sweep of training, the model observes all verbs in proportion to their frequency in the input; for each verb, the proportion of training trials with 1 in each of the output nodes corresponds to the frequency of the verb in each of those constructions. During testing, only the input nodes are activated (corresponding to a verb and its semantics), and the activation of output nodes reveals the learned proportional activation rate corresponding to the degree of verb bias toward either the DO or the PD (or *other*).

The structure of the AB model encodes some assumptions regarding the information and learning mechanisms available to the learner. The model incorporates awareness of individual verbs by having a node per verb in the input to distinguish the usage of each verb and its accompanying features. Each verb is also represented by a vector of semantic features that capture properties relevant to its meaning when used in one of the two dative constructions (based on elicited human judgments from Ambridge et al., 2014). The "transfer" input node encodes the ability to distinguish the semantic properties of the dative constructions from other constructions: i.e., this node is set to 1 for a DO or PD usage, and to 0 otherwise. Representing the construction of the input usage (DO, PD, or *other*) on the output nodes reflects the formalization of the learning as an association of semantic and lexical features with a syntactic pattern, and the knowledge of the model is demonstrated by activating the construction output nodes in response to a lexical/semantic input.

3.2 Overview of the Bayesian Model

The BFS of model Barak et al. (2014) is a Bayesian clustering model that simultaneously and incrementally learns both constructions and verb classes in a two-level design; see Figure 2 for an illustration of each level. In learning, the model processes an input sequence of verb usages, represented as collections of semantic and syntactic features, one usage at a time. The first step of processing each input aims to

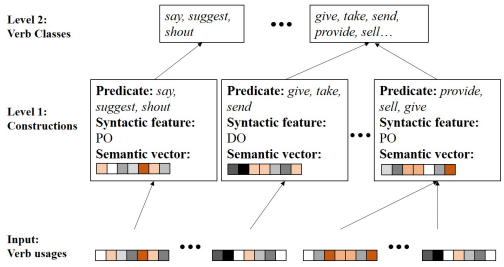


Figure 2: A visual representation of the the Bayesian model, with sample input features for verb usages, construction level, and verb class level.

find the best cluster at level one as:

$$\text{BestCluster}(F_i) = \underset{k \in Clusters}{\text{argmax}}\ P(k|F_i) \quad (1)$$

where F_i is the set of features for input i, and k ranges over all existing clusters and a new one. The number of possible clusters is not set in advanced, and thus at any step the best choice may be to start a new cluster (of size 1) with this input.

Using Bayes rule:

$$P(k|F_i) = \frac{P(k)P(F_i|k)}{P(F_i)} \propto P(k)P(F_i|k) \quad (2)$$

The prior probability of a cluster $P(k)$ is proportional to the number of verb usages clustered to k so far, thus assigning a higher prior to larger clusters. The likelihood $P(F_i|k)$ is estimated based on the match of feature values in the current verb usage to those aggregated in the cluster, where the quality of the match depends on the frequency and vector similarity of the two sets of features.

The clusters at this level correspond to constructions of the language – i.e., probabilistic associations of form and meaning. For example, a cluster emerges from semantically-similar verbs like *tell* and *ask*, in a particular syntax, such as the DO. Creating a new cluster – forming a new construction – depends on both the likelihood and the prior. Early on, the $P(F_i|k)$ term has more influence and differences in feature values between a new usage and existing clusters will often trigger a new cluster. Later, the model will favour adding a new input to an existing cluster – even if it makes it more heterogeneous – because the $P(k)$ term prefers larger clusters as the number of observed inputs increases. This mechanism mimics human language learning behavior of moving from more verb-specific constructions to more general constructions (Tomasello, 2003).

Each verb can occur in several clusters in the first level based on its association with various semantic and syntactic features. For instance, the alternating verb *give* can occur in one cluster associating a transfer meaning with PD syntax and in a second cluster associating a transfer meaning with DO syntax. To capture the common behaviour of such alternating verbs – where verbs with similar meanings occur across the same set of clusters – the model represents the similarity in distributional properties of the verbs in a second level of representation, which captures such verb class behaviours.

Formally, after each clustering decision in the first level, the model calculates the current frequency distribution of the input verb over all level-one clusters. This distribution vector is used as input for the second level: the model measures the similarity of this vector to the weighted average distribution represented by each second-level cluster, adding the input verb's distribution to the most similar one:

$$\text{BestClass}(d_{v_t}) = \underset{c \in Classes}{\text{argmax}}\ (1 - D_{\text{JS}}(d_c \| d_{v_t})) \quad (3)$$

where d_{v_t} is the current distribution of the verb v over the clusters (i.e., at time t), c ranges over all the classes in the second level, d_c is the weighted average of c given the distributions of its member verb tokens, and D_{JS} is the Jensen–Shannon divergence. As in the first level, the model may create a new cluster in the second level if none of the existing clusters is similar enough to d_{v_t}.

The resulting second-level clusters capture verb class behavior by grouping verbs that share a pattern of usages across constructions, e.g., alternating verbs that occur with the DO and PD syntax. These clusters encode a snapshot of the distribution of each verb each time it occurs in the input, reflecting the need of the language learner to incrementally update their knowledge of the distributional behavior of the verb across constructions.

3.3 Comparison of the Models

Both models capture the semantic and statistical properties of language proposed as possible factors in the ability to learn an alternation appropri-

ately – i.e., to generalize to new uses but not overgeneralize to inappropriate verbs. Semantic influences are reflected in the use of meaning features, and each model incorporates the key idea behind statistical preemption (Goldberg, 1995), namely that semantically-appropriate constructions compete with one another. The statistical effects of overall verb frequency and of frequency of the verb-in-construction are captured by inputting each verb in proportion to its frequency with each construction.

The models have a crucial difference in how they reflect the influence of the various features in learning alternations. As a feed-forward network, the AB model learns the weight of the semantic features given the entire set of input, and uses these weightings to shape the prediction of a verb's preference for each of the syntactic constructions (represented by the target output nodes). The BFS model does not explicitly weight features, but the influence of a feature is determined by its local context within a cluster. For example, if the value of a feature has high frequency in a cluster – e.g., the cluster records usages with only the DO syntax – the predictions based on this cluster would strongly prefer matching usages based on this feature value; a less-frequent feature would have less influence on this cluster's predictions, but could have more influence in a cluster where it is more represented. This property, along with the representation of an open-ended set of constructions (level one clusters) and verb classes (level two clusters), enables the model to capture rich interactions among the lexical, semantic, and syntactic features. We evaluate the role of these differences in the fit of each model to the task.

4 Experimental Setup

4.1 Input and Training

We base the learning and evaluation on the 281 distinct verbs used in Ambridge and Blything (2015), which had been determined to occur in the double object (DO) and/or the preposition-dative (PD) (Pinker, 1989; Levin, 1993). Following Ambridge and Blything (2015), we consider a third (artificial) construction labeled as *other* that corresponds to *all* non-DO and non-PD usages of a verb. The models are trained on usages of the verbs in proportion to their frequencies in the British National Corpus (BNC) (Leech, 1992). Table 1 summarizes the per-construction frequency data for the verbs.[1] Note that 98 of the verbs can occur in the DO and/or PD but have no such occurrences in the BNC; these verbs unseen in the dative are important for judging the appropriate generalization behavior of the models.

		Raw freq			Log freq		
	#Verbs	DO	PD	*other*	DO	PD	*other*
PD	101	0	93	9964	0	3	5
DO	7	49	0	877	2	0	4
Alt	75	325	1144	13332	3	3	7
Uns	98	0	0	716	0	0	4

Table 1: Frequency data for the dative verbs in the BNC for non-alternating **PD**-only and **DO**-only verbs, **ALT**ernating verbs, and **UNS**een dative-taking verbs that do not occur with the dative constructions in the BNC.

The input to the models include: the lexeme of the verb, the semantic features of the verb, a "transfer" feature marking the common meaning of the dative constructions, and (in training only) a syntactic feature. The syntactic feature indicates whether a verb is used with the DO, PD, or *other* construction; in the AB model, this is given as the target output node in training. The verb semantic features are those used in Ambridge and Blything (2015). These vectors are based on the ratings of each verb on 18 meaning properties relevant to use of the verb in the dative (e.g., "The verb specifies the means of transfer" Ambridge et al., 2014), subject to Principal Component Analysis by Ambridge and Blything (2015), yielding a vector of 7 dimensions. The transfer feature is 1 for a verb usage in one of the two dative constructions, and 0 for the *other* construction, to indicate the shared transfer meaning conveyed by the DO and the PD. The input to each model is generated automatically to correspond to the BNC frequencies of each verb in each of the constructions.

It should be noted that, while we adopt the semantic features of Ambridge and Blything (2015), they reflect the meaning within the two dative constructions and may be less applicable to the *other* construction. In addition, we found that there are alternating and non-alternating verbs that have very similar semantic vectors, indicating that these fea-

[1] The full list of verbs and their frequencies can be found in Ambridge and Blything (2015).

tures may not sufficiently distinguish the alternation behaviours.

The models are trained with sufficient input to converge on stable behavior. We follow Ambridge and Blything (2015) in training and testing the AB model using the OXlearn MATLAB package (Ruh and Westermann, 2009); the input is generated using a random seed, in random input order without replacements, and the model is trained with a learning rate of 0.01 for 1K sweeps for log frequencies; 100K sweeps for raw frequencies. We train the BFS model using the input generation method described by Barak et al. (2014), with the features as above. The model is trained on 5K input verb usages (in proportion to their frequencies in the constructions).

4.2 Evaluation of the Models

As in Ambridge and Blything (2015), to test the model preferences for the DO or PD, the models are presented with an input consisting of a verb lexeme, its semantic features, and the transfer feature set to 1 (i.e., this is a "transfer" semantics suitable for a dative construction). For the AB model, we measure preferences for each construction as the activation rate of each of the corresponding output nodes, as in Ambridge and Blything (2015). In the BFS model, the preference for each construction is measured as its likelihood over the learned clusters given the verb and its semantic features. Formally, the prediction in the Bayesian model is:

$$P(s|F_{\text{test}}) = \sum_{k \in Clusters} P(s|k)P(k|F_{\text{test}}) \quad (4)$$

where s is the predicted syntactic construction (DO or PD) and F_{test} is the set of test features representing a verb v and its corresponding semantic features. $P(s|k)$ is the probability of the syntactic pattern feature having the value s in cluster k, calculated as the proportional occurrences of s in k. $P(k|F_{\text{test}})$ is the probability of cluster k given test features F_{test}, calculated as in Eqn. (2). Following Barak et al. (2014), we calculate $P(k|F_{\text{test}})$ in two ways, using just the constructions (level one) or both the classes (level two) and the constructions, to see whether verb class knowledge improves performance. Using solely the construction level, the probability of k reflects the frequency with which usages of verb v occur in cluster k. Using the verb class level in addition, the distribution of the verb over classes in the second level is combined with the distribution of those classes over the constructions in level one, to get the likelihood of k.

These model preferences of the verbs for a dative construction are compared, using Pearson correlation, to the DO/PD acceptability judgment data collected from adult participants by Ambridge et al. (2014). Note that Ambridge and Blything (2015) only evaluate their model's preferences for verbs to take the DO construction. To fully understand the preference and generalization patterns, we also analyze the results for the PD preference. Even more importantly, we calculate the *difference* between the preferences for the DO and the PD constructions *per verb*, and compare these to analogous scores for the human data, as suggested by Ambridge et al. (2014). The DO−PD difference scores, which we will refer to as the **verb bias score**, are crucial because, as in the human data, it is these scores that accurately capture a learner's relative preference for a construction *given a particular verb*.

5 Experiments and Analysis of Results

We examine the ability of each model to match the dative construction preferences of human judgments, as described just above, under two different experimental scenarios. In Section 5.1, we follow the experimental settings of Ambridge and Blything (2015). We replicate their results on the AB model showing correlation with human DO preferences, but find that only the BFS model achieves a significant correlation with the crucial verb bias score that appropriately assesses per-verb preference. We adjust the experimental settings in Section 5.2 to use more naturalistic input data – by training in proportion to raw frequencies and excluding the artificial *other* construction – achieving an improvement in the verb bias score for both models.

5.1 Exp 1: Log Freq Input; 3 Constructions

Results. We first evaluated the models under the experimental conditions of Ambridge and Blything (2015), providing input corresponding to the verbs in 3 constructions (DO, PD, and *other*), in proportion to their log frequencies; see Table 2. We replicate the positive correlation of the AB model over

	AB (Connectionist)	BFS (Bayesian)	
		Level 1	Level 2
DO	**0.54**	0.24	0.29
PD	0.39	0.30	**0.50**
DO-PD	[-0.02]	0.48	**0.53**

Table 2: Pearson correlation values between human and model preferences for each construction and the verb-bias score (DO−PD); training on log frequencies and 3 constructions. All correlations significant with p-value < 0.001, except the one value in square brackets. Best result for each row is marked in boldface.

the ratings for the DO construction found in Ambridge and Blything (2015). In addition, our analysis shows that the AB model produces a significant positive correlation with the PD acceptability rating. However, the AB model has no correlation with the verb bias score. Although the model ranks the separate verb preferences for DO and PD similarly to humans, the model does not produce the same relative preference *for individual verbs*. For example, the human data rank *give* with high acceptability in both the DO and the PD, with a higher value for the DO construction. Although the AB model has a high preference for both constructions for *give* (compared with other verbs), the model erroneously prefers *give* in the PD construction.

The BFS model also produces preferences for verbs in each construction that have a significant positive correlation with human judgments. While the AB model shows better correlation with the DO judgments, the BFS model correlates more strongly with the PD judgments. Importantly, in contrast to the AB model, the verb bias score of the BFS model also significantly correlates with the judgment data. That is, the BFS model provides a better prediction of the preference per verb, which is key to producing a verb in the appropriate syntax.

Analysis. We can explain these results by looking more closely at the properties of the input and the differences in the learning mechanisms of each model. Following Ambridge and Blything (2015), the input presents an artificial *other* construction in proportion to the frequency of the verbs with all non-dative constructions. The very high frequency of this single artificial construction (see *other* in Table 1) results in higher predictions of it for any of the verbs, even though the "transfer" feature in test inputs has a value intended to signal one of the dative constructions. As a result, the preferences for the dative constructions in both models have a very small range of values, showing relatively small differences.

The BFS model is also affected by the relatively compressed semantic space of the input, which is exacerbated by the use of log frequencies to guide the input. As noted earlier, we found that the semantic features of alternating verbs can be highly similar to non-alternating verbs – e.g., *give* (alternating) and *pull* (PO-only) have similar semantic vectors. With such input, the model cannot form sufficiently distinct first-level clusters based on the semantics, particularly when the data is presented with such a flat distribution (note the small differences in log frequencies in Table 1). Visual inspection reveals that these clusters in the model largely form around syntactic constructions, with mixed semantic properties. Despite this, the first-level clusters capture a strong enough association between individual verbs and their constructions to yield a good correlation of the verb bias score with human judgments, and drawing on the second-level (verb-class) clusters improves the results.

Conclusions. The use of an artificial high-frequency non-dative construction (*other*), and the use of log frequencies, seem to mask the influence of the semantic and syntactic properties on learning the verb-bias for each verb. Previous psycholinguistic data and computational models have found that a skewed naturalistic distribution of the input is helpful in learning constructions, due to the high-frequency verbs establishing appropriate construction-meaning associations (Casenhiser and Goldberg, 2005; Borovsky and Elman, 2006; Barak et al., 2013b; Matusevych et al., 2014). To allow a more direct analysis of the role of statistical and semantic properties in learning and generalizing the dative, we adjust the input to the models in the next section.

5.2 Exp 2: Raw Freq Input; 2 Constructions

Results. Here we perform the same type of experiments, but using input in proportion to the raw frequencies of the verbs (instead of log frequencies) over occurrences only in the two dative constructions (with no *other* construction). Since 98 of the 281 verbs do not occur with either dative construc-

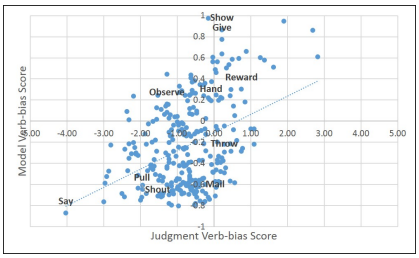
	AB (Connectionist)	BFS (Bayesian)	
		Level 1	Level 2
DO	[0.06]	0.23	**0.25**
PD	0.33	**0.38**	0.32
DO-PD	0.39	0.53	**0.59**

Table 3: Pearson correlation values between human and model preferences for each construction and the verb-bias score; training on raw frequencies and 2 constructions. All correlations significant with p-value < 0.001, except the one value in square brackets. Best result for each row is marked in boldface.

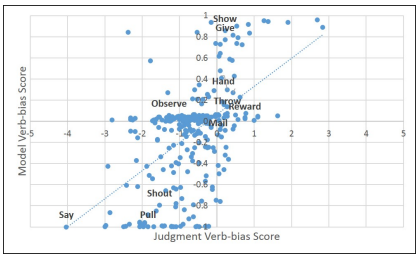
tion in the BNC, this also allows us to more stringently test the generalization ability of the models, by considering their behavior when ˜$1/3$ of the verbs are unseen in training.

Table 3 presents the correlation results for the two models' preferences for each construction and the verb bias score; we also show the correlation plots for the verb bias score in Figure 3. The AB model does not correlate with the judgments for the DO. However, the model produces significant positive correlations with the PD judgments and with the verb bias score. The BFS model, on the other hand, achieves significant positive correlations on all measures, by both levels. As in the earlier experiments, the best correlation with the verb bias score is produced by the second level of the BFS model, as Figure 3 demonstrates.

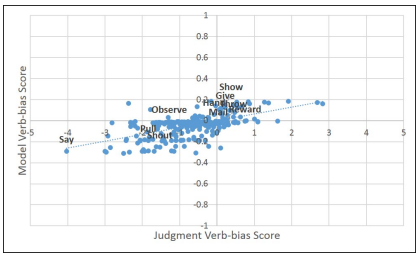
Analysis. As shown by Barak et al. (2013b), the Bayesian model is better at learning the distribution pattern of each verb class given a skewed distribution, as in the raw frequencies here. The model learns an association of each construction to the frequently-observed meaning of high-frequency verbs. For example, the semantics of the DO is most strongly influenced by the semantics of its most frequently occurring instance: *give*. The accuracy of preference judgments benefits from the entrenchment of the relevant meaning with the construction. This supports appropriate generalization – e.g., because *reward* is semantically similar to *give*, it has a good fit to the human preference judgments even though it is unseen with the dative (see Figure 3). But the same factor can serve to limit generalization – e.g., because the unseen verb *mail* is semantic *dissimilar* to a frequent PD-only verb like *pull*, its preference for the PD syntax is limited, giving it a



(a) AB model



(b) BFS model - construction level



(c) BFS model - verb class level

Figure 3: Correlation of the dative verbs with the verb bias score of each model in Exp. 2: (a) the AB model ($r = 0.39$), (b) the first level of the BFS model ($r = 0.53$), and (c) the second level of the BFS model ($r = 0.59$).

good match to human judgments by preventing its overgeneralization (see Figure 3).

The AB model can also take advantage of high frequency verbs biasing the preference toward the frequently observed association. However, the semantic similarity across verbs within alternating or non-alternating classes is less effective in this model. The representation of the lexemes as 281 nodes in the input (compared to less than a dozen other nodes) make the learning more verb specific, reducing the ability of the model to generalize to the unattested verbs.

Conclusions. The success of the BFS model, and especially the results using both constructions and classes, point to the role of probabilistic constructions and verb classes in generalizing exist-

ing knowledge while avoiding overgeneralizations. Moreover, the use of a skewed distribution reveals the role of the high verb-in-construction frequency in guiding the association of construction and meaning (see Ambridge et al., 2014, for discussion). Yet both models would benefit from a richer semantic representation that better captures the distinctive properties of verbs across various constructions.

6 Discussion

This paper presents a comparative analysis of two computational cognitive models on the sample task of learning the dative alternation. This study enables an evaluation of the psycholinguistic plausibility of each model for the given task when facing identical input and experimental settings. Adopting the semantic representation of Ambridge and Blything (2015), our input incorporates both semantic and syntactic properties over a large number of verbs. By providing the first direct comparison between two existing models of this phenomenon, we are the first to demonstrate the complex interaction of various linguistic properties in the input, and how rich learning mechanisms are required in order to achieve generalizations compatible with human judgments in this area. Moreover, comparison of learning mechanisms and of input properties can inform CL/NLP more generally by shedding light on potential factors in achieving humanlike behaviours.

We find that the Bayesian model of BFS significantly correlates with human judgments on the 3 key evaluation measures. Importantly, this model outperforms the connectionist model of AB in the correlation with the verb-bias score (the per-verb difference between DO and PD preference), which points to its advantage in choosing the more appropriate construction per verb. We argue that the fit of the model relies on a rich learning mechanism that exploits distributional properties of naturalistic input.

The AB model has a streamlined design to support learning a particular semantic-syntactic association underlying the dative alternation. While the BFS model is richer computationally, its properties were motivated in earlier work explaining many human behaviours. When we consider more natural input, the simple input[semantics]–output[syntax] association mechanism of the AB model is unable to capture the necessary interactions among the verb semantic properties, the syntactic usages, and their patterns across different types of verbs. By contrast, the two-level design of the BFS model captures these interactions. The first level learns the verb-semantics-syntax associations as clusters of similar configurations of those features. The second level captures the commonalities of behaviour of sets of verbs by forming classes of verbs that have similar distributional patterns over the first-level clusters. We also observe that the replication of adult-like language competence relies on several naturalistic properties of the input: skewed distribution, and a rich semantic representation combined with syntactic information. The skewed input enables the formation of clusters representing more entrenched associations, which are biased towards high-frequency verbs associated with certain semantic and syntactic features.

Given the role of these linguistic properties, the results here call for additional analysis and development of the input to computational cognitive models. The predictions may be improved given more realistic syntactic and semantic information about the verb usages. On the syntax side, the input should reflect the distribution of verbs across more syntactic constructions, as statistical patterns over such usages can indirectly indicate aspects of a verb's semantics (cf. Barak et al., 2013a). In the future, we aim to analyze the role of fuller syntactic distributions in restricting overgeneralization patterns. Moreover, the semantic annotations used here replicate the settings originally tested for the AB model, which correspond to the verb as used in the relevant constructions. This contrasts with typical automated extractions of verb-meaning representation (e.g., word2vec, Mikolov et al., 2013), which capture a more general verb meaning across all its usages. In preliminary experiments, we have found an advantage in using word2vec representations in addition to the semantic properties reported here. We aim to further analyze manual and automated methods for semantic feature extraction in future work.

Acknowledgments

We are grateful to Ben Ambridge for helpful discussion of his model and for sharing his data with us.

References

Nameera Akhtar. 1999. Acquiring basic word order: Evidence for data-driven learning of syntactic structure. *Journal of child language*, 26(02):339–356.

Afra Alishahi and Pirita Pyykkoïen. 2011. The onset of syntactic bootstrapping in word learning: Evidence from a computational study. In *Proceedings of the 33st Annual Conference of the Cognitive Science Society*.

Afra Alishahi and Suzanne Stevenson. 2008. A computational model of early argument structure acquisition. *Cognitive Science*, 32(5):789–834.

Ben Ambridge and Ryan P Blything. 2015. A connectionist model of the retreat from verb argument structure overgeneralization. *Journal of child language*, pages 1–32.

Ben Ambridge, Julian M Pine, Caroline F Rowland, and Franklin Chang. 2012. The roles of verb semantics, entrenchment, and morphophonology in the retreat from dative argument-structure overgeneralization errors. *Language*, 88(1):45–81.

Ben Ambridge, Julian M Pine, Caroline F Rowland, Daniel Freudenthal, and Franklin Chang. 2014. Avoiding dative overgeneralisation errors: Semantics, statistics or both? *Language, Cognition and Neuroscience*, 29(2):218–243.

Ben Ambridge, Julian M Pine, Caroline F Rowland, and Chris R Young. 2008. The effect of verb semantic class and verb frequency (entrenchment) on childrens and adults graded judgements of argument-structure overgeneralization errors. *Cognition*, 106(1):87–129.

Libby Barak, Afsaneh Fazly, and Suzanne Stevenson. 2012. Modeling the acquisition of mental state verbs. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2012)*.

Libby Barak, Afsaneh Fazly, and Suzanne Stevenson. 2013a. Acquisition of desires before beliefs: A computational investigation. In *Proceedings of CoNLL-2013*.

Libby Barak, Afsaneh Fazly, and Suzanne Stevenson. 2013b. Modeling the emergence of an exemplar verb in construction learning. In *Proceedings of the 35rd Annual Meeting of the Cognitive Science Society*.

Libby Barak, Afsaneh Fazly, and Suzanne Stevenson. 2014. Learning verb classes in an incremental model. In *Proceedings of the 5th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2014)*. Association for Computational Linguistics.

Arielle Borovsky and Jeff Elman. 2006. Language input and semantic categories: A relation between cognition and early word learning. *Journal of child language*, 33(04):759–790.

Jeremy K Boyd and Adele E Goldberg. 2009. Input effects within a constructionist framework. *The Modern Language Journal*, 93(3):418–429.

Martin DS Braine and Patricia J Brooks. 1995. Verb argument structure and the problem of avoiding an overgeneral grammar. *Beyond names for things: Young children's acquisition of verbs*, pages 353–376.

Joan Bresnan and Marilyn Ford. 2010. Predicting syntax: Processing dative constructions in american and australian varieties of english. *Language*, 86(1):168–213.

Joan Bybee. 2010. *Language, usage and cognition*. Cambridge University Press.

David Casenhiser and Adele E. Goldberg. 2005. Fast mapping between a phrasal form and meaning. *Developmental Science*, 8(6):500–508.

Cynthia Fisher. 1999. From form to meaning: A role for structural alignment in the acquisition of language. *Advances in child development and behavior*, 27:1–53.

Marilyn Ford, Joan W Bresnan, and Ronald Kaplan. 1982. A competence-based theory of syntactic closure. *American Journal of Computational Linguistics*, 8(1):49.

Susanne Gahl and Susan M Garnsey. 2004. Knowledge of grammar, knowledge of usage: Syntactic probabilities affect pronunciation variation. *Language*, pages 748–775.

Susan M Garnsey, Neal J Pearlmutter, Elizabeth Myers, and Melanie A Lotocky. 1997. The contributions of verb bias and plausibility to the com-

prehension of temporarily ambiguous sentences. *Journal of Memory and Language*, 37(1):58–93.

Adele E. Goldberg. 1995. *Constructions, A Construction Grammar Approach to Argument Structure*. {Chicago University Press}.

Adele E Goldberg. 2006. *Constructions at work: The nature of generalization in language*. Oxford University Press on Demand.

Adele E Goldberg. 2011. Corpus evidence of the viability of statistical preemption. *Cognitive Linguistics*, 22(1):131–153.

Geoffrey Leech. 1992. 100 million words of english: the british national corpus (BNC). *Language Research*, 28(1):1–13.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*, volume 348. University of Chicago press Chicago, IL.

Elena VM Lieven, Julian M Pine, and Gillian Baldwin. 1997. Lexically-based learning and early grammatical development. *Journal of child language*, 24(01):187–219.

Beth L Losiewicz. 1992. *The effect of frequency on linguistic morphology*. University of Texas.

Maryellen C MacDonald, Neal J Pearlmutter, and Mark S Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological review*, 101(4):676.

Yevgen Matusevych, Afra Alishahi, and Ad Backus. 2014. Isolating second language learning factors in a computational study of bilingual construction acquisition. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, pages 988–994.

Yevgen Matusevych, Afra Alishahi, and Ad Backus. 2016. The impact of first and second language exposure on learning second language constructions. *Bilingualism: Language and Cognition*, pages 1–22.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Christopher Parisien and Suzanne Stevenson. 2010. Learning verb alternations in a usage-based bayesian model. In *Proceedings of the 32nd annual meeting of the Cognitive Science Society*.

Amy Perfors, Joshua B. Tenenbaum, and Elizabeth Wonnacott. 2010. Variability, negative evidence, and the acquisition of verb argument constructions. *Journal of Child Language*, 37(03):607–642.

Steven Pinker. 1989. *Learnability and cognition: The acquisition of argument structure.* The MIT Press.

Nicolas Ruh and Gert Westermann. 2009. Oxlearn: A new matlab-based simulation tool for connectionist models. *Behavior research methods*, 41(4):1138–1143.

Anna L Theakston. 2004. The role of entrenchment in childrens and adults performance on grammaticality judgment tasks. *Cognitive Development*, 19(1):15–34.

Michael Tomasello. 2003. *Constructing a language: A usage-based theory of language acquisition*. Harvard University Press.

John C Trueswell, Michael K Tanenhaus, and Christopher Kello. 1993. Verb-specific constraints in sentence processing: separating effects of lexical preference from garden-paths. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 19(3):528.

Thomas Wasow. 2002. *Postverbal behavior*. Stanford Univ Center for the Study.

Rationalizing Neural Predictions

Tao Lei, Regina Barzilay and Tommi Jaakkola
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{taolei, regina, tommi}@csail.mit.edu

Abstract

Prediction without justification has limited applicability. As a remedy, we learn to extract pieces of input text as justifications – rationales – that are tailored to be short and coherent, yet sufficient for making the same prediction. Our approach combines two modular components, generator and encoder, which are trained to operate well together. The generator specifies a distribution over text fragments as candidate rationales and these are passed through the encoder for prediction. Rationales are never given during training. Instead, the model is regularized by desiderata for rationales. We evaluate the approach on multi-aspect sentiment analysis against manually annotated test cases. Our approach outperforms attention-based baseline by a significant margin. We also successfully illustrate the method on the question retrieval task.[1]

1 Introduction

Many recent advances in NLP problems have come from formulating and training expressive and elaborate neural models. This includes models for sentiment classification, parsing, and machine translation among many others. The gains in accuracy have, however, come at the cost of interpretability since complex neural models offer little transparency concerning their inner workings. In many applications, such as medicine, predictions are used to drive critical decisions, including treatment options. It is necessary in such cases to be able to verify and understand the underlying basis for the decisions. Ideally, complex neural models would not only yield improved performance but would also offer interpretable justifications – rationales – for their predictions.

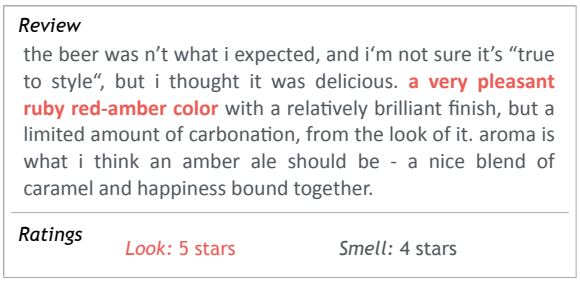


Figure 1: An example of a review with ranking in two categories. The rationale for Look prediction is shown in bold.

In this paper, we propose a novel approach to incorporating rationale generation as an integral part of the overall learning problem. We limit ourselves to extractive (as opposed to abstractive) rationales. From this perspective, our rationales are simply subsets of the words from the input text that satisfy two key properties. First, the selected words represent short and coherent pieces of text (e.g., phrases) and, second, the selected words must alone suffice for prediction as a substitute of the original text. More concretely, consider the task of multi-aspect sentiment analysis. Figure 1 illustrates a product review along with user rating in terms of two categories or aspects. If the model in this case predicts five star rating for color, it should also identify the phrase *"a very pleasant ruby red-amber color"* as the rationale underlying this decision.

In most practical applications, rationale genera-

[1]Our code and data are available at https://github.com/taolei87/rcnn.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 107–117,
Austin, Texas, November 1-5, 2016.

tion must be learned entirely in an unsupervised manner. We therefore assume that our model with rationales is trained on the same data as the original neural models, without access to additional rationale annotations. In other words, target rationales are never provided during training; the intermediate step of rationale generation is guided only by the two desiderata discussed above. Our model is composed of two modular components that we call the generator and the encoder. Our generator specifies a distribution over possible rationales (extracted text) and the encoder maps any such text to task specific target values. They are trained jointly to minimize a cost function that favors short, concise rationales while enforcing that the rationales alone suffice for accurate prediction.

The notion of what counts as a rationale may be ambiguous in some contexts and the task of selecting rationales may therefore be challenging to evaluate. We focus on two domains where ambiguity is minimal (or can be minimized). The first scenario concerns with multi-aspect sentiment analysis exemplified by the beer review corpus (McAuley et al., 2012). A smaller test set in this corpus identifies, for each aspect, the sentence(s) that relate to this aspect. We can therefore directly evaluate our predictions on the sentence level with the caveat that our model makes selections on a finer level, in terms of words, not complete sentences. The second scenario concerns with the problem of retrieving similar questions. The extracted rationales should capture the main purpose of the questions. We can therefore evaluate the quality of rationales as a compressed proxy for the full text in terms of retrieval performance. Our model achieves high performance on both tasks. For instance, on the sentiment prediction task, our model achieves extraction accuracy of 96%, as compared to 38% and 81% obtained by the bigram SVM and a neural attention baseline.

2 Related Work

Developing sparse interpretable models is of considerable interest to the broader research community(Letham et al., 2015; Kim et al., 2015). The need for interpretability is even more pronounced with recent neural models. Efforts in this area include analyzing and visualizing state activation (Hermans and Schrauwen, 2013; Karpathy et al., 2015; Li et al., 2016), learning sparse interpretable word vectors (Faruqui et al., 2015b), and linking word vectors to semantic lexicons or word properties (Faruqui et al., 2015a; Herbelot and Vecchi, 2015).

Beyond learning to understand or further constrain the network to be directly interpretable, one can estimate interpretable proxies that approximate the network. Examples include extracting "if-then" rules (Thrun, 1995) and decision trees (Craven and Shavlik, 1996) from trained networks. More recently, Ribeiro et al. (2016) propose a model-agnostic framework where the proxy model is learned only for the target sample (and its neighborhood) thus ensuring locally valid approximations. Our work differs from these both in terms of what is meant by an explanation and how they are derived. In our case, an explanation consists of a concise yet sufficient portion of the text where the mechanism of selection is learned jointly with the predictor.

Attention based models offer another means to explicate the inner workings of neural models (Bahdanau et al., 2015; Cheng et al., 2016; Martins and Astudillo, 2016; Chen et al., 2015; Xu and Saenko, 2015; Yang et al., 2015). Such models have been successfully applied to many NLP problems, improving both prediction accuracy as well as visualization and interpretability (Rush et al., 2015; Rocktäschel et al., 2016; Hermann et al., 2015). Xu et al. (2015) introduced a stochastic attention mechanism together with a more standard soft attention on image captioning task. Our rationale extraction can be understood as a type of stochastic attention although architectures and objectives differ. Moreover, we compartmentalize rationale generation from downstream encoding so as to expose knobs to directly control types of rationales that are acceptable, and to facilitate broader modular use in other applications.

Finally, we contrast our work with rationale-based classification (Zaidan et al., 2007; Marshall et al., 2015; Zhang et al., 2016) which seek to improve prediction by relying on richer annotations in the form of human-provided rationales. In our work, rationales are never given during training. The goal is to learn to generate them.

3 Extractive Rationale Generation

We formalize here the task of extractive rationale generation and illustrate it in the context of neural models. To this end, consider a typical NLP task where we are provided with a sequence of words as input, namely $\mathbf{x} = \{x_1, \cdots, x_l\}$, where each $x_t \in \mathbb{R}^d$ denotes the vector representation of the i-th word. The learning problem is to map the input sequence $\mathbf{x}$ to a target vector in $\mathbb{R}^m$. For example, in multi-aspect sentiment analysis each coordinate of the target vector represents the response or rating pertaining to the associated aspect. In text retrieval, on the other hand, the target vectors are used to induce similarity assessments between input sequences. Broadly speaking, we can solve the associated learning problem by estimating a complex parameterized mapping $\mathbf{enc}(\mathbf{x})$ from input sequences to target vectors. We call this mapping an *encoder*. The training signal for these vectors is obtained either directly (e.g., multi-sentiment analysis) or via similarities (e.g., text retrieval). The challenge is that a complex neural encoder $\mathbf{enc}(\mathbf{x})$ reveals little about its internal workings and thus offers little in the way of justification for why a particular prediction was made.

In extractive rationale generation, our goal is to select a subset of the input sequence as a *rationale*. In order for the subset to qualify as a rationale it should satisfy two criteria: 1) the selected words should be interpretable and 2) they ought to suffice to reach nearly the same prediction (target vector) as the original input. In other words, a rationale must be short and sufficient. We will assume that a short selection is interpretable and focus on optimizing sufficiency under cardinality constraints.

We encapsulate the selection of words as a *rationale generator* which is another parameterized mapping $\mathbf{gen}(\mathbf{x})$ from input sequences to shorter sequences of words. Thus $\mathbf{gen}(\mathbf{x})$ must include only a few words and $\mathbf{enc}(\mathbf{gen}(\mathbf{x}))$ should result in nearly the same target vector as the original input passed through the encoder or $\mathbf{enc}(\mathbf{x})$. We can think of the generator as a tagging model where each word in the input receives a binary tag pertaining to whether it is selected to be included in the rationale. In our case, the generator is probabilistic and specifies a distribution over possible selections.

The rationale generation task is entirely unsupervised in the sense that we assume no explicit annotations about which words should be included in the rationale. Put another way, the rationale is introduced as a latent variable, a constraint that guides how to interpret the input sequence. The encoder and generator are trained jointly, in an end-to-end fashion so as to function well together.

4 Encoder and Generator

We use multi-aspect sentiment prediction as a guiding example to instantiate the two key components – the encoder and the generator. The framework itself generalizes to other tasks.

Encoder $\mathbf{enc}(\cdot)$: Given a training instance $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = \{x_t\}_{t=1}^{l}$ is the input text sequence of length l and $\mathbf{y} \in [0, 1]^m$ is the target m-dimensional sentiment vector, the neural encoder predicts $\tilde{\mathbf{y}} = \mathbf{enc}(\mathbf{x})$. If trained on its own, the encoder would aim to minimize the discrepancy between the predicted sentiment vector $\tilde{\mathbf{y}}$ and the gold target vector $\mathbf{y}$. We will use the squared error (i.e. L_2 distance) as the sentiment loss function,

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \|\tilde{\mathbf{y}} - \mathbf{y}\|_2^2 = \|\mathbf{enc}(\mathbf{x}) - \mathbf{y}\|_2^2$$

The encoder could be realized in many ways such as a recurrent neural network. For example, let $\mathbf{h}_t = f_e(\mathbf{x}_t, \mathbf{h}_{t-1})$ denote a parameterized recurrent unit mapping input word $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$ to next state $\mathbf{h}_t$. The target vector is then generated on the basis of the final state reached by the recurrent unit after processing all the words in the input sequence. Specifically,

$$\mathbf{h}_t = f_e(\mathbf{x}_t, \mathbf{h}_{t-1}), \;\; t = 1, \ldots, l$$
$$\tilde{\mathbf{y}} = \sigma_e(\mathbf{W}^e \mathbf{h}_l + \mathbf{b}^e)$$

Generator $\mathbf{gen}(\cdot)$: The rationale generator extracts a subset of text from the original input $\mathbf{x}$ to function as an interpretable summary. Thus the rationale for a given sequence $\mathbf{x}$ can be equivalently defined in terms of binary variables $\{\mathbf{z}_1, \cdots, \mathbf{z}_l\}$ where each $\mathbf{z}_t \in 0, 1$ indicates whether word $\mathbf{x}_t$ is selected or not. From here on, we will use $\mathbf{z}$ to specify the binary selections and thus $(\mathbf{z}, \mathbf{x})$ is the actual rationale generated (selections, input). We will use generator $\mathbf{gen}(\mathbf{x})$ as synonymous with a

probability distribution over binary selections, i.e., $\mathbf{z} \sim \mathbf{gen}(\mathbf{x}) \equiv p(\mathbf{z}|\mathbf{x})$ where the length of $\mathbf{z}$ varies with the input $\mathbf{x}$.

In a simple generator, the probability that the t^{th} word is selected can be assumed to be conditionally independent from other selections given the input $\mathbf{x}$. That is, the joint probability $p(\mathbf{z}|\mathbf{x})$ factors according to

$$p(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{l} p(\mathbf{z}_t|\mathbf{x}) \quad \text{(independent selection)}$$

The component distributions $p(\mathbf{z}_t|\mathbf{x})$ can be modeled using a shared bi-directional recurrent neural network. Specifically, let $\overrightarrow{f}()$ and $\overleftarrow{f}()$ be the forward and backward recurrent unit, respectively, then

$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{f}(\mathbf{x}_t, \overrightarrow{\mathbf{h}_{t-1}})$$
$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{f}(\mathbf{x}_t, \overleftarrow{\mathbf{h}_{t+1}})$$
$$p(\mathbf{z}_t|\mathbf{x}) = \sigma_z(\mathbf{W}^z[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}] + \mathbf{b}^z)$$

Independent but context dependent selection of words is often sufficient. However, the model is unable to select phrases or refrain from selecting the same word again if already chosen. To this end, we also introduce a dependent selection of words,

$$p(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{l} p(\mathbf{z}_t|\mathbf{x}, \mathbf{z}_1 \cdots \mathbf{z}_{t-1})$$

which can be also expressed as a recurrent neural network. To this end, we introduce another hidden state $\mathbf{s}_t$ whose role is to couple the selections. For example,

$$p(\mathbf{z}_t|\mathbf{x}, \mathbf{z}_{1,t-1}) = \sigma_z(\mathbf{W}^z[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}; \mathbf{s}_{t-1}] + \mathbf{b}^z)$$
$$\mathbf{s}_t = f_z([\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}; \mathbf{z}_t], \mathbf{s}_{t-1})$$

Joint objective: A rationale in our definition corresponds to the selected words, i.e., $\{\mathbf{x}_k|\mathbf{z}_k = 1\}$. We will use $(\mathbf{z}, \mathbf{x})$ as the shorthand for this rationale and, thus, $\mathbf{enc}(\mathbf{z}, \mathbf{x})$ refers to the target vector obtained by applying the encoder to the rationale as the input. Our goal here is to formalize how the rationale can be made short and meaningful yet function well in conjunction with the encoder. Our generator and encoder are learned jointly to interact well but they are treated as independent units for modularity.

The generator is guided in two ways during learning. First, the rationale that it produces must suffice as a replacement for the input text. In other words, the target vector (sentiment) arising from the rationale should be close to the gold sentiment. The corresponding loss function is given by

$$\mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \|\mathbf{enc}(\mathbf{z}, \mathbf{x}) - \mathbf{y}\|_2^2$$

Note that the loss function depends directly (parametrically) on the encoder but only indirectly on the generator via the sampled selection.

Second, we must guide the generator to realize short and coherent rationales. It should select only a few words and those selections should form phrases (consecutive words) rather than represent isolated, disconnected words. We therefore introduce an additional regularizer over the selections

$$\Omega(\mathbf{z}) = \lambda_1\|\mathbf{z}\| + \lambda_2 \sum_t |\mathbf{z}_t - \mathbf{z}_{t-1}|$$

where the first term penalizes the number of selections while the second one discourages transitions (encourages continuity of selections). Note that this regularizer also depends on the generator only indirectly via the selected rationale. This is because it is easier to assess the rationale once produced rather than directly guide how it is obtained.

Our final cost function is the combination of the two, $\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) + \Omega(\mathbf{z})$. Since the selections are not provided during training, we minimize the expected cost:

$$\min_{\theta_e, \theta_g} \sum_{(\mathbf{x},\mathbf{y}) \in D} \mathbb{E}_{\mathbf{z} \sim \mathbf{gen}(\mathbf{x})} \left[\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y})\right]$$

where θ_e and θ_g denote the set of parameters of the encoder and generator, respectively, and D is the collection of training instances. Our joint objective encourages the generator to compress the input text into coherent summaries that work well with the associated encoder it is trained with.

Minimizing the expected cost is challenging since it involves summing over all the possible choices of rationales $\mathbf{z}$. This summation could potentially be made feasible with additional restrictive assumptions about the generator and encoder. However, we assume only that it is possible to efficiently sample from the generator.

Doubly stochastic gradient We now derive a sampled approximation to the gradient of the expected cost objective. This sampled approximation is obtained separately for each input text $\mathbf{x}$ so as to work well with an overall stochastic gradient method. Consider therefore a training pair $(\mathbf{x}, \mathbf{y})$. For the parameters of the generator θ_g,

$$\begin{aligned}
&\frac{\partial \mathbb{E}_{\mathbf{z}\sim\mathbf{gen}(\mathbf{x})}\left[\text{cost}(\mathbf{z},\mathbf{x},\mathbf{y})\right]}{\partial \theta_g} \\
&= \sum_{\mathbf{z}} \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \\
&= \sum_{\mathbf{z}} \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot \frac{p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}
\end{aligned}$$

Using the fact $(\log f(\theta))' = f'(\theta)/f(\theta)$, we get

$$\begin{aligned}
&\sum_{\mathbf{z}} \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot \frac{p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \\
&= \sum_{\mathbf{z}} \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y}) \cdot \frac{\partial \log p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot p(\mathbf{z}|\mathbf{x}) \\
&= \mathbb{E}_{z\sim\mathbf{gen}(\mathbf{x})}\left[\text{cost}(\mathbf{z},\mathbf{x},\mathbf{y}) \frac{\partial \log p(\mathbf{z}|\mathbf{x})}{\partial \theta_g}\right]
\end{aligned}$$

The last term is the expected gradient where the expectation is taken with respect to the generator distribution over rationales $\mathbf{z}$. Therefore, we can simply sample a few rationales $\mathbf{z}$ from the generator $\mathbf{gen}(\mathbf{x})$ and use the resulting average gradient in an overall stochastic gradient method. A sampled approximation to the gradient with respect to the encoder parameters θ_e can be derived similarly,

$$\begin{aligned}
&\frac{\partial \mathbb{E}_{\mathbf{z}\sim\mathbf{gen}(\mathbf{x})}\left[\text{cost}(\mathbf{z},\mathbf{x},\mathbf{y})\right]}{\partial \theta_e} \\
&= \sum_{\mathbf{z}} \frac{\partial \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y})}{\partial \theta_e} \cdot p(\mathbf{z}|\mathbf{x}) \\
&= \mathbb{E}_{z\sim\mathbf{gen}(\mathbf{x})}\left[\frac{\partial \text{cost}(\mathbf{z},\mathbf{x},\mathbf{y})}{\partial \theta_e}\right]
\end{aligned}$$

Choice of recurrent unit We employ recurrent convolution (RCNN), a refinement of local-ngram based convolution. RCNN attempts to learn n-gram features that are not necessarily consecutive, and average features in a dynamic (recurrent) fashion. Specifically, for bigrams (filter width $n = 2$) RCNN computes $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$ as follows

Number of reviews	1580k
Avg length of review	144.9
Avg correlation between aspects	63.5%
Max correlation between two aspects	79.1%
Number of annotated reviews	994

Table 1: Statistics of the beer review dataset.

$$\begin{aligned}
\lambda_t &= \sigma(\mathbf{W}^{\lambda}\mathbf{x}_t + \mathbf{U}^{\lambda}\mathbf{h}_{t-1} + \mathbf{b}^{\lambda}) \\
\mathbf{c}_t^{(1)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(1)} + (1-\lambda_t) \odot (\mathbf{W}_1\mathbf{x}_t) \\
\mathbf{c}_t^{(2)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(2)} + (1-\lambda_t) \odot (\mathbf{c}_{t-1}^{(1)} + \mathbf{W}_2\mathbf{x}_t) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t^{(2)} + \mathbf{b})
\end{aligned}$$

RCNN has been shown to work remarkably in classification and retrieval applications (Lei et al., 2015; Lei et al., 2016) compared to other alternatives such CNNs and LSTMs. We use it for all the recurrent units introduced in our model.

5 Experiments

We evaluate the proposed joint model on two NLP applications: (1) multi-aspect sentiment analysis on product reviews and (2) similar text retrieval on AskUbuntu question answering forum.

5.1 Multi-aspect Sentiment Analysis

Dataset We use the BeerAdvocate[2] review dataset used in prior work (McAuley et al., 2012).[3] This dataset contains 1.5 million reviews written by the website users. The reviews are naturally multi-aspect – each of them contains multiple sentences describing the *overall* impression or one particular aspect of a beer, including *appearance*, *smell* (aroma), *palate* and the *taste*. In addition to the written text, the reviewer provides the ratings (on a scale of 0 to 5 stars) for each aspect as well as an overall rating. The ratings can be fractional (e.g. 3.5 stars), so we normalize the scores to $[0, 1]$ and use them as the (only) supervision for regression.

McAuley et al. (2012) also provided sentence-level annotations on around 1,000 reviews. Each sentence is annotated with one (or multiple) aspect label, indicating what aspect this sentence covers.

[2] www.beeradvocate.com

[3] http://snap.stanford.edu/data/web-BeerAdvocate.html

Method	Appearance		Smell		Palate	
	% precision	% selected	% precision	% selected	% precision	% selected
SVM	38.3	13	21.6	7	24.9	7
Attention model	80.6	13	88.4	7	65.3	7
Generator (independent)	94.8	13	93.8	7	79.3	7
Generator (recurrent)	96.3	14	95.1	7	80.2	7

Table 2: Precision of selected rationales for the first three aspects. The precision is evaluated based on whether the selected words are in the sentences describing the target aspect, based on the sentence-level annotations. Best training epochs are selected based on the objective value on the development set (no sentence annotation is used).

	D	d	l	$\|\theta\|$	MSE
SVM	260k	-	-	2.5M	0.0154
SVM	1580k	-	-	7.3M	0.0100
LSTM	260k	200	2	644k	0.0094
RCNN	260k	200	2	323k	**0.0087**

Table 3: Comparing neural encoders with bigram SVM model. MSE is the mean squared error on the test set. D is the amount of data used for training and development. d stands for the hidden dimension, l denotes the depth of network and $|\theta|$ denotes the number of parameters (number of features for SVM).

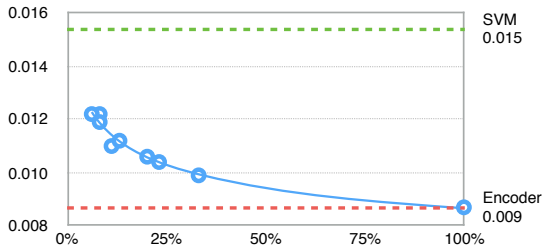


Figure 2: Mean squared error of all aspects on the test set (y-axis) when various percentages of text are extracted as rationales (x-axis). 220k training data is used.

We use this set as our test set to evaluate the precision of words in the extracted rationales.

Table 1 shows several statistics of the beer review dataset. The sentiment correlation between any pair of aspects (and the overall score) is quite high, getting 63.5% on average and a maximum of 79.1% (between the *taste* and *overall* score). If directly training the model on this set, the model can be confused due to such strong correlation. We therefore perform a preprocessing step, picking "less correlated" examples from the dataset.[4] This gives us a de-correlated subset for each aspect, each containing about 80k to 90k reviews. We use 10k as the development set. We focus on three aspects since the fourth aspect *taste* still gets $> 50\%$ correlation with the overall sentiment.

Sentiment Prediction Before training the joint model, it is worth assessing the neural encoder separately to check how accurately the neural network predicts the sentiment. To this end, we compare neural encoders with bigram SVM model, training medium and large SVM models using 260k and all 1580k reviews respectively. As shown in Table 3, the recurrent neural network models outperform the SVM model for sentiment prediction and also require less training data to achieve the performance. The LSTM and RCNN units obtain similar test error, getting 0.0094 and 0.0087 mean squared error respectively. The RCNN unit performs slightly better and uses less parameters. Based on the results, we choose the RCNN encoder network with 2 stacking layers and 200 hidden states.

To train the joint model, we also use RCNN unit with 200 states as the forward and backward recurrent unit for the generator $\mathbf{gen}()$. The dependent generator has one additional recurrent layer. For this layer we use 30 states so the dependent version still has a number of parameters comparable to the independent version. The two versions of the generator have 358k and 323k parameters respectively.

Figure 2 shows the performance of our joint dependent model when trained to predict the sentiment of all aspects. We vary the regularization λ_1 and λ_2 to show various runs that extract different amount of text as rationales. Our joint model gets performance close to the best encoder run (with full text) when few words are extracted.

[4] Specifically, for each aspect we train a simple linear regression model to predict the rating of this aspect given the ratings of the other four aspects. We then keep picking reviews with largest prediction error until the sentiment correlation in the selected subset increases dramatically.

a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a generous head that sustained life throughout . nothing out of the ordinary here , but a good brew still . body was kind of heavy , but not thick . the hop smell was excellent and enticing . very drinkable

very dark beer . pours a nice finger and a half of creamy foam and stays throughout the beer . smells of coffee and roasted malt . has a major coffee-like taste with hints of chocolate . if you like black coffee , you will love this porter . creamy smooth mouthfeel and definitely gets smoother on the palate once it warms . it 's an ok porter but i feel there are much better one 's out there .

i really did not like this . it just seemed extremely watery . i dont ' think this had any carbonation whatsoever . maybe it was flat , who knows ? but even if i got a bad brew i do n't see how this would possibly be something i 'd get time and time again . i could taste the hops towards the middle , but the beer got pretty nasty towards the bottom . i would never drink this again , unless it was free . i 'm kind of upset i bought this .

a : poured a nice dark brown with a tan colored head about half an inch thick , nice red/garnet accents when held to the light . little clumps of lacing all around the glass , not too shabby . not terribly impressive though s : smells like a more guinness-y guinness really , there are some roasted malts there , signature guinness smells , less burnt though , a little bit of chocolate ... ... m : relatively thick , it is n't an export stout or imperial stout , but still is pretty hefty in the mouth , very smooth , not much carbonation . not too shabby d : not quite as drinkable as the draught , but still not too bad . i could easily see drinking a few of these .

Figure 3: Examples of extracted rationales indicating the sentiments of various aspects. The extracted texts for appearance, smell and palate are shown in red, blue and green color respectively. The last example is shortened for space.

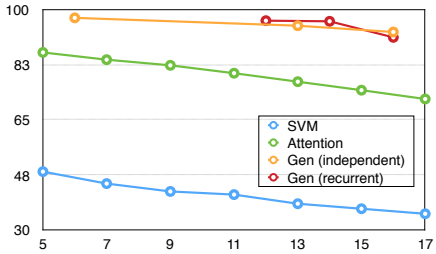


Figure 4: Precision (y-axis) when various percentages of text are extracted as rationales (x-axis) for the appearance aspect.

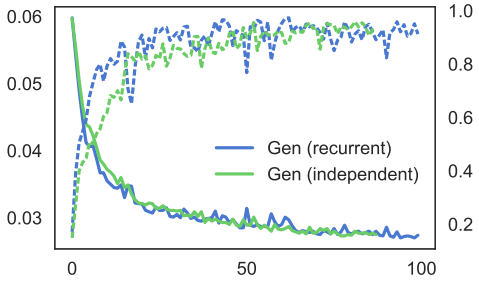


Figure 5: Learning curves of the optimized cost function on the development set and the precision of rationales on the test set. The smell (aroma) aspect is the target aspect.

Rationale Selection To evaluate the supporting rationales for each aspect, we train the joint encoder-generator model on each de-correlated subset. We set the cardinality regularization λ_1 between values $\{2e-4, 3e-4, 4e-4\}$ so the extracted rationale texts are neither too long nor too short. For simplicity, we set $\lambda_2 = 2\lambda_1$ to encourage local coherency of the extraction.

For comparison we use the bigram SVM model and implement an attention-based neural network model. The SVM model successively extracts unigram or bigram (from the test reviews) with the highest feature. The attention-based model learns a normalized attention vector of the input tokens (using similarly the forward and backward RNNs), then the model averages over the encoder states accordingly to the attention, and feed the averaged vector to the output layer. Similar to the SVM model, the attention-based model can selects words based on their attention weights.

Table 2 presents the precision of the extracted rationales calculated based on sentence-level aspect annotations. The λ_1 regularization hyper-parameter is tuned so the two versions of our model extract similar number of words as rationales. The SVM and attention-based model are constrained similarly for comparison. Figure 4 further shows the precision when different amounts of text are extracted. Again, for our model this corresponds to changing the λ_1 regularization. As shown in the table and the figure, our encoder-generator networks extract text pieces describing the target aspect with high precision, ranging from 80% to 96% across the three aspects appearance, smell and palate. The SVM baseline performs poorly, achieving around 30% accuracy. The attention-based model achieves reasonable but worse performance than the rationale generator, suggesting the potential of directly modeling rationales as explicit extraction.

Figure 5 shows the learning curves of our model for the smell aspect. In the early training epochs, both the independent and (recurrent) dependent selection models fail to produce good rationales, getting low precision as a result. After a few epochs of exploration however, the models start to achieve high accuracy. We observe that the dependent version learns more quickly in general, but both versions obtain close results in the end.

Finally we conduct a qualitative case study on the extracted rationales. Figure 3 presents several reviews, with highlighted rationales predicted by the model. Our rationale generator identifies key phrases or adjectives that indicate the sentiment of a particular aspect.

5.2 Similar Text Retrieval on QA Forum

Dataset For our second application, we use the real-world AskUbuntu[5] dataset used in recent work (dos Santos et al., 2015; Lei et al., 2016). This set contains a set of 167k unique questions (each consisting a question title and a body) and 16k user-identified similar question pairs. Following previous work, this data is used to train the neural encoder that learns the vector representation of the input question, optimizing the cosine distance (i.e. cosine similarity) between similar questions against random non-similar ones. We use the "one-versus-all" hinge loss (i.e. positive versus other negatives) for the encoder, similar to (Lei et al., 2016). During development and testing, the model is used to score 20 candidate questions given each query question, and a total of 400×20 query-candidate question pairs are annotated for evaluation[6].

Task/Evaluation Setup The question descriptions are often long and fraught with irrelevant details. In this set-up, a fraction of the original question text should be sufficient to represent its content, and be used for retrieving similar questions. Therefore, we will evaluate rationales based on the accuracy of the question retrieval task, assuming that better rationales achieve higher performance. To put this performance in context, we also report the accuracy when full body of a question is used, as well as titles alone. The latter constitutes an upper bound on the model performance as in this dataset titles provide short, informative summaries of the question content. We evaluate the rationales using the mean average precision (MAP) of retrieval.

	MAP (dev)	MAP (test)	%words
Full title	56.5	60.0	10.1
Full body	54.2	53.0	89.9
Independent	55.7	53.6	9.7
	56.3	52.6	19.7
Dependent	56.1	54.6	11.6
	56.5	55.6	32.8

Table 4: Comparison between rationale models (middle and bottom rows) and the baselines using full title or body (top row).

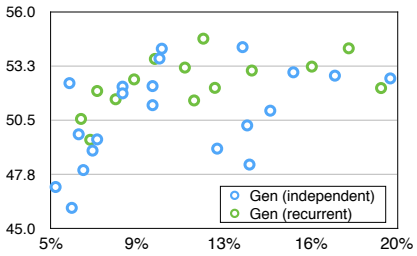


Figure 6: Retrieval MAP on the test set when various percentages of the texts are chosen as rationales. Data points correspond to models trained with different hyper-parameters.

Results Table 4 presents the results of our rationale model. We explore a range of hyper-parameter values[7]. We include two runs for each version. The first one achieves the highest MAP on the development set, The second run is selected to compare the models when they use roughly 10% of question text (7 words on average). We also show the results of different runs in Figure 6. The rationales achieve the MAP up to 56.5%, getting close to using the titles. The models also outperform the baseline of using the noisy question bodies, indicating the the models' capacity of extracting short but important fragments.

Figure 7 shows the rationales for several questions in the AskUbuntu domain, using the recurrent version with around 10% extraction. Interestingly, the model does not always select words from the question title. The reasons are that the question body can contain the same or even complementary information useful for retrieval. Indeed, some rationale fragments shown in the figure are error messages,

[5] askubuntu.com

[6] https://github.com/taolei87/askubuntu

[7] $\lambda_1 \in \{.008, .01, .012, .015\}$, $\lambda_2 = \{0, \lambda_1, 2\lambda_1\}$, dropout $\in \{0.1, 0.2\}$

what is the easiest way to install all the media codec available for ubuntu ? i am having issues with multiple applications prompting me to install codecs before they can play my files . how do i install media codecs ?

---

what should i do when i see <unk> report this <unk> ? an unresolvable problem occurred while initializing the package information . please report this bug against the 'update-manager ' package and include the following error message : e : encountered a section with no package : header e : problem with mergelist <unk> e : the package lists or status file could not be parsed or opened .

---

please any one give the solution for this whenever i try to convert the rpm file to deb file i always get this problem error : <unk> : not an rpm package (or package manifest) error executing `` lang=c rpm -qp -- queryformat % { name } <unk> ' '' : at <unk> line 489 thanks converting rpm file to debian fle

---

how do i mount a hibernated partition with windows 8 in ubuntu ? i ca n't mount my other partition with windows 8 , i have ubuntu 12.10 amd64 : error mounting /dev/sda1 at <unk> : command-line `mount -t `` ntfs '' -o `` uhelper=udisks2 , nodev , nosuid , uid=1000 , gid=1000 , dmask=0077 , fmask=0177 '' `` /dev/sda1 '' `` <unk> '' ' exited with non-zero exit status 14 : windows is hibernated , refused to mount . failed to mount '/dev/sda1 ' : operation not permitted the ntfs partition is hibernated . please resume and shutdown windows properly , or mount the volume read-only with the 'ro ' mount option

Figure 7: Examples of extracted rationales of questions in the AskUbuntu domain.

which are typically not in the titles but very useful to identify similar questions.

6 Discussion

We proposed a novel modular neural framework to automatically generate concise yet sufficient text fragments to justify predictions made by neural networks. We demonstrated that our encoder-generator framework, trained in an end-to-end manner, gives rise to quality rationales in the absence of any explicit rationale annotations. The approach could be modified or extended in various ways to other applications or types of data.

Choices of $\mathbf{enc}(\cdot)$ and $\mathbf{gen}(\cdot)$. The encoder and generator can be realized in numerous ways without changing the broader algorithm. For instance, we could use a convolutional network (Kim, 2014; Kalchbrenner et al., 2014), deep averaging network (Iyyer et al., 2015; Joulin et al., 2016) or a boosting classifier as the encoder. When rationales can be expected to conform to repeated stereotypical patterns in the text, a simpler encoder consistent with this bias can work better. We emphasize that, in this paper, rationales are flexible explanations that may vary substantially from instance to another. On the generator side, many additional constraints could be imposed to further guide acceptable rationales.

Dealing with Search Space. Our training method employs a REINFORCE-style algorithm (Williams, 1992) where the gradient with respect to the parameters is estimated by sampling possible rationales. Additional constraints on the generator output can be helpful in alleviating problems of exploring potentially a large space of possible rationales in terms of their interaction with the encoder. We could also apply variance reduction techniques to increase stability of stochastic training (cf. (Weaver and Tao, 2001; Mnih et al., 2014; Ba et al., 2015; Xu et al., 2015)).

7 Acknowledgments

We thank Prof. Julian McAuley for sharing the review dataset and annotations. We also thank MIT NLP group and the reviewers for their helpful comments. The work is supported by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the IYAS project. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. Abc-

cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Mark W Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems (NIPS)*.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015a. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015b. Sparse overcomplete word vector representations. In *Proceedings of ACL*.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

B Kim, JA Shah, and F Doshi-Velez. 2015. Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL*.

Iain J Marshall, Joël Kuiper, and Byron C Wallace. 2015. Robotreviewer: evaluation of a system for automatically assessing bias in clinical trials. *Journal of the American Medical Informatics Association*.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. *CoRR*, abs/1602.02068.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1020–1025. IEEE.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NIPS)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Sebastian Thrun. 1995. Extracting rules from artificial neural networks with distributed representations. In *Advances in neural information processing systems (NIPS)*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Omar Zaidan, Jason Eisner, and Christine D. Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 260–267.

Ye Zhang, Iain James Marshall, and Byron C. Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. *CoRR*, abs/1605.04469.

Deep Multi-Task Learning with Shared Memory

Pengfei Liu Xipeng Qiu* Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Neural network based models have achieved impressive results on various specific tasks. However, in previous works, most models are learned separately based on single-task supervised objectives, which often suffer from insufficient training data. In this paper, we propose two deep architectures which can be trained jointly on multiple related tasks. More specifically, we augment neural model with an external memory, which is shared by several tasks. Experiments on two groups of text classification tasks show that our proposed architectures can improve the performance of a task with the help of other related tasks.

1 Introduction

Neural network based models have been shown to achieved impressive results on various NLP tasks rivaling or in some cases surpassing traditional models, such as text classification (Kalchbrenner et al., 2014; Socher et al., 2013; Liu et al., 2015a), semantic matching (Hu et al., 2014; Liu et al., 2016a), parser (Chen and Manning, 2014) and machine translation (Bahdanau et al., 2014).

Usually, due to the large number of parameters these neural models need a large-scale corpus. It is hard to train a deep neural model that generalizes well with size-limited data, while building the large scale resources for some NLP tasks is also a challenge. To overcome this problem, these models often involve an unsupervised pre-training phase. The final model is fine-tuned on specific task with respect to a supervised training criterion. However, most pre-training methods are based on unsupervised objectives (Collobert et al., 2011; Turian et al., 2010; Mikolov et al., 2013), which is effective to improve the final performance, but it does not directly optimize the desired task.

Multi-task learning is an approach to learn multiple related tasks simultaneously to significantly improve performance relative to learning each task independently. Inspired by the success of multi-task learning (Caruana, 1997), several neural network based models (Collobert and Weston, 2008; Liu et al., 2015b) are proposed for NLP tasks, which utilized multi-task learning to jointly learn several tasks with the aim of mutual benefit. The characteristic of these multi-task architectures is they share some lower layers to determine common features. After the shared layers, the remaining layers are split into multiple specific tasks.

In this paper, we propose two deep architectures of sharing information among several tasks in multi-task learning framework. All the related tasks are integrated into a single system which is trained jointly. More specifically, inspired by Neural Turing Machine (NTM) (Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), we equip task-specific long short-term memory (LSTM) neural network (Hochreiter and Schmidhuber, 1997) with an external shared memory. The external memory has capability to store long term information and knowledge shared by several related tasks. Different with NTM, we use a deep fusion strategy to integrate the information from the external memory into task-specific LSTM, in which a fusion gate controls the

* Corresponding author.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 118–127,
Austin, Texas, November 1-5, 2016.

information flowing flexibly and enables the model to selectively utilize the shared information.

We demonstrate the effectiveness of our architectures on two groups of text classification tasks. Experimental results show that jointly learning of multiple related tasks can improve the performance of each task relative to learning them independently.

Our contributions are of three-folds:

- We proposed a generic multi-task framework, in which different tasks can share information by an external memory and communicate by a reading/writing mechanism. Two proposed models are complementary to prior multi-task neural networks.
- Different with Neural Turing Machine and memory network, we introduce a deep fusion mechanism between internal and external memories, which helps the LSTM units keep them interacting closely without being conflated.
- As a by-product, the fusion gate enables us to better understand how the external shared memory helps specific task.

2 Neural Memory Models for Specific Task

In this section, we briefly describe LSTM model, and then propose an external memory enhanced LSTM with deep fusion.

2.1 Long Short-term Memory

Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies. LSTM maintains an internal memory cell that updates and exposes its content only when deemed necessary.

Architecturally speaking, the memory state and output state are explicitly separated by activation gates (Wang and Cho, 2015). However, the limitation of LSTM is that it lacks a mechanism to index its memory while writing and reading (Danihelka et al., 2016).

While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The elements of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$.

The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_p \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}_p \right), \quad (1)$$

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where $\mathbf{x}_t \in \mathbb{R}^m$ is the input at the current time step; $\mathbf{W} \in \mathbb{R}^{4h \times (d+m)}$ and $\mathbf{b}_p \in \mathbb{R}^{4h}$ are parameters of affine transformation; σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

The update of each LSTM unit can be written precisely as follows:

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t, \theta_p). \quad (4)$$

Here, the function $\mathbf{LSTM}(\cdot, \cdot, \cdot, \cdot)$ is a shorthand for Eq. (1-3), and θ_p represents all the parameters of LSTM.

2.2 Memory Enhanced LSTM

LSTM has an internal memory to keep useful information for specific task, some of which may be beneficial to other tasks. However, it is non-trivial to share information stored in internal memory.

Recently, there are some works to augment LSTM with an external memory, such as neural Turing machine (Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), called memory enhanced LSTM (ME-LSTM). These models enhance the low-capacity internal memory to have a capability of modelling long pieces of text (Andrychowicz and Kurach, 2016).

Inspired by these models, we introduce an external memory to share information among several tasks. To better control shared information and understand how it is utilized from external memory, we propose a deep fusion strategy for ME-LSTM.

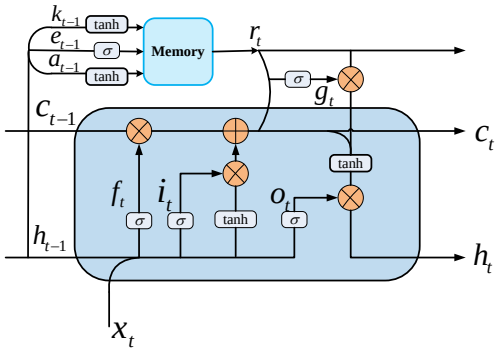


Figure 1: Graphical illustration of the proposed ME-LSTM unit with deep fusion of internal and external memories.

As shown in Figure 1, ME-LSTM consists the original LSTM and an external memory which is maintained by reading and writing operations. The LSTM not only interacts with the input and output information but accesses the external memory using selective read and write operations.

The external memory and corresponding operations will be discussed in detail below.

External Memory The form of external memory is defined as a matrix $\mathbf{M} \in \mathbb{R}^{K \times M}$, where K is the number of memory segments, and M is the size of each segment. Besides, K and M are generally instance-independent and pre-defined as hyper-parameters.

At each step t, LSTM emits output $\mathbf{h}_t$ and three key vectors $\mathbf{k}_t$, $\mathbf{e}_t$ and $\mathbf{a}_t$ simultaneously. $\mathbf{k}_t$, $\mathbf{e}_t$ and $\mathbf{a}_t$ can be computed as

$$\begin{bmatrix} \mathbf{k}_t \\ \mathbf{e}_t \\ \mathbf{a}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \tanh \end{bmatrix} (\mathbf{W}_m \mathbf{h}_t + \mathbf{b}_m) \tag{5}$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are parameters of affine transformation.

Reading The read operation is to read information $\mathbf{r}_t \in \mathbb{R}^M$ from memory $\mathbf{M}_{t-1}$.

$$\mathbf{r}_t = \alpha_t \mathbf{M}_{t-1}, \tag{6}$$

where $\mathbf{r}_t$ denotes the reading vector and $\alpha_t \in \mathbb{R}^K$ represents a distribution over the set of segments of memory $\mathbf{M}_{t-1}$, which controls the amount of information to be read from and written to the memory.

Each scalar $\alpha_{t,k}$ in attention distribution α_t can be obtained as:

$$\alpha_{t,k} = \text{softmax}(g(\mathbf{M}_{t-1,k}, \mathbf{k}_{t-1})) \tag{7}$$

where $\mathbf{M}_{t-1,k}$ represents the k-th row memory vector, and $\mathbf{k}_{t-1}$ is a key vector emitted by LSTM.

Here $g(\mathbf{x}, \mathbf{y})$ ($\mathbf{x} \in \mathbb{R}^M, \mathbf{y} \in \mathbb{R}^M$) is a align function for which we consider two different alternatives:

$$g(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathbf{v}^T \tanh(\mathbf{W}_a[\mathbf{x}; \mathbf{y}]) \\ \mathbf{cosine}(x, y) \end{cases} \tag{8}$$

where $\mathbf{v} \in \mathbb{R}^M$ is a parameter vector.

In our current implementation, the similarity measure is cosine similarity.

Writing The memory can be written by two operations: erase and add.

$$\mathbf{M}_t = \mathbf{M}_{t-1}(\mathbf{1} - \alpha_t \mathbf{e}_t^\mathbf{T}) + \alpha_t \mathbf{a}_t^\mathbf{T}, \tag{9}$$

where $\mathbf{e}_t, \mathbf{a}_t \in \mathbb{R}^M$ represent erase and add vectors respectively.

To facilitate the following statements, we re-write the writing equation as:

$$\mathbf{M}_t = \mathbf{f}_{write}(\mathbf{M}_{t-1}, \alpha_t, \mathbf{h}_t). \tag{10}$$

Deep Fusion between External and Internal Memories After we obtain the information from external memory, we need a strategy to comprehensively utilize information from both external and internal memory.

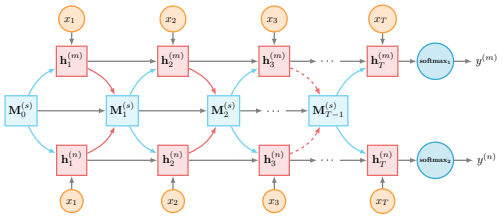
To better control signals flowing from external memory, inspired by (Wang and Cho, 2015), we propose a deep fusion strategy to keep internal and external memories interacting closely without being conflated.

In detail, the state $\mathbf{h}_t$ of LSTM at step t depends on both the read vector $\mathbf{r}_t$ from external memory, and internal memory c_t, which is computed by

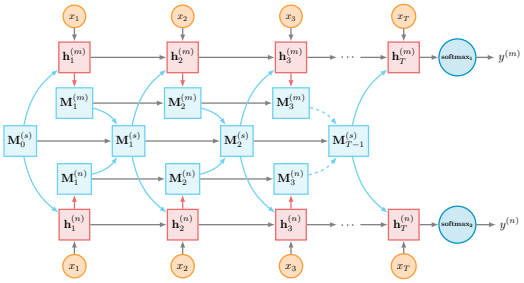
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t + \mathbf{g}_t \odot (\mathbf{W}_f \mathbf{r}_t)), \tag{11}$$

where $\mathbf{W}_f$ is parameter matrix, and $\mathbf{g}_t$ is a fusion gate to select information from external memory, which is computed by

$$\mathbf{g}_t = \sigma(\mathbf{W}_r \mathbf{r}_t + \mathbf{W}_c \mathbf{c}_t), \tag{12}$$



(a) Global Memory Architecture



(b) Local-Global Hybrid Memory Architecture

Figure 2: Two architectures for modelling text with multi-task learning.

where $\mathbf{W}_r$ and $\mathbf{W}_c$ are parameter matrices.

Finally, the update of external memory enhanced LSTM unit can be written precisely as

$$(\mathbf{h}_t, \mathbf{M}_t, \mathbf{c}_t) = \textbf{ME-LSTM}(\mathbf{h}_{t-1}, \mathbf{M}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t, \theta_p, \theta_q), \quad (13)$$

where θ_p represents all the parameters of LSTM internal structure and θ_q represents all the parameters to maintain the external memory.

Overall, the external memory enables ME-LSTM to have larger capability to store more information, thereby increasing the ability of ME-LSTM. The read and write operations allow ME-LSTM to capture complex sentence patterns.

3 Deep Architectures with Shared Memory for Multi-task Learning

Most existing neural network methods are based on supervised training objectives on a single task (Collobert et al., 2011; Socher et al., 2013; Kalchbrenner et al., 2014). These methods often suffer from the limited amounts of training data. To deal with this problem, these models often involve an unsupervised pre-training phase. This unsupervised pre-training is effective to improve the final performance, but it does not directly optimize the desired task.

Motivated by the success of multi-task learning (Caruana, 1997), we propose two deep architectures with shared external memory to leverage supervised data from many related tasks. Deep neural model is well suited for multi-task learning since the features learned from a task may be useful for other tasks. Figure 2 gives an illustration of our proposed architectures.

ARC-I: Global Shared Memory In ARC-I, the input is modelled by a task-specific LSTM and external shared memory. More formally, given an input text x, the task-specific output $\mathbf{h}_t^{(m)}$ of task m at step t is defined as

$$(\mathbf{h}_t^{(m)}, \mathbf{M}_t^{(s)}, \mathbf{c}_t^{(m)}) = \textbf{ME-LSTM}(\mathbf{h}_{t-1}^{(m)}, \mathbf{M}_{t-1}^{(s)}, \mathbf{c}_{t-1}^{(m)}, \mathbf{x}_t, \theta_p^{(m)}, \theta_q^{(s)}), \quad (14)$$

where $\mathbf{x}_t$ represents word embeddings of word x_t; the superscript s represents the parameters are shared across different tasks; the superscript m represents that the parameters or variables are task-specific for task m.

Here all tasks share single global memory $\mathbf{M}^{(s)}$, meaning that all tasks can read information from it and have the duty to write their shared or task-specific information into the memory.

$$\mathbf{M}_t^{(s)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(s)}, \alpha_t^{(s)}, \mathbf{h}_t^{(m)}) \quad (15)$$

After calculating the task-specific representation of text $\mathbf{h}_T^{(m)}$ for task m, we can predict the probability distribution over classes.

ARC-II: Local-Global Hybrid Memory In ARC-I, all tasks share a global memory, but can also record task-specific information besides shared information. To address this, we allocate each task a local task-specific external memory, which can further write shared information to a global memory for all tasks.

More generally, for task m, we assign each task-specific LSTM with a local memory $\mathbf{M}^{(m)}$, followed by a global memory $\mathbf{M}^{(s)}$, which is shared across different tasks.

The read and write operations of the local and global memory are defined as

$$\mathbf{r}_t^{(m)} = \alpha_t^{(m)} \mathbf{M}_t^{(m)}, \quad (16)$$

Dataset		Type	Train Size	Dev. Size	Test Size	Class	Avg. Length	Vocabulary Size
Movie	SST-1	Sen.	8544	1101	2210	5	19	18K
	SST-2	Sen.	6920	872	1821	2	18	15K
	SUBJ	Sen.	9000	-	1000	2	21	21K
	IMDB	Doc.	25,000	-	25,000	2	294	392K
Product	Books	Doc.	1400	200	400	2	181	27K
	DVDs	Doc.	1400	200	400	2	197	29K
	Electronics	Doc.	1400	200	400	2	117	14K
	Kitchen	Doc.	1400	200	400	2	98	12K

Table 1: Statistics of two multi-task datasets. Each dataset consists of four related tasks.

$$\mathbf{M}_t^{(m)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(m)}, \alpha_t^{(m)}, \mathbf{h}_t^{(m)}), \quad (17)$$

$$\mathbf{r}_t^{(s)} = \alpha_{t-1}^{(s)} \mathbf{M}_{t-1}^{(s)}, \quad (18)$$

$$\mathbf{M}_t^{(s)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(s)}, \alpha_t^{(s)}, \mathbf{r}_t^{(m)}), \quad (19)$$

where the superscript s represents the parameters are shared across different tasks; the superscript m represents that the parameters or variables are task-specific for task m.

In ARC-II, the local memories enhance the capacity of memorizing, while global memory enables the information flowing from different tasks to interact sufficiently.

4 Training

The task-specific representation $\mathbf{h}^{(m)}$, emitted by the deep muti-task architectures, is ultimately fed into the corresponding task-specific output layers.

$$\hat{\mathbf{y}}^{(m)} = \mathrm{softmax}(\mathbf{W}^{(m)}\mathbf{h}^{(m)} + \mathbf{b}^{(m)}), \quad (20)$$

where $\hat{\mathbf{y}}^{(m)}$ is prediction probabilities for task m.

Given M related tasks, our global cost function is the linear combination of cost function for all tasks.

$$\phi = \sum_{m=1}^{M} \lambda_m L(\hat{y}^{(m)}, y^{(m)}) \quad (21)$$

where λ_m is the weights for each task m respectively.

Computational Cost Compared with vanilla LSTM, our proposed two models do not cause much extra computational cost while converge faster. In our experiment, the most complicated ARC-II, costs 2 times as long as vanilla LSTM.

	Movie Reviews	Product Reviews
Embedding dimension	100	100
Hidden layer size	100	100
External memory size	(50,20)	(50,20)
Initial learning rate	0.01	0.1
Regularization	0	$1E{-}5$

Table 2: Hyper-parameters of our models.

5 Experiment

In this section, we investigate the empirical performances of our proposed architectures on two multi-task datasets. Each dataset contains several related tasks.

5.1 Datasets

The used multi-task datasets are briefly described as follows. The detailed statistics are listed in Table 1.

Movie Reviews The movie reviews dataset consists of four sub-datasets about movie reviews.

- **SST-1** The movie reviews with five classes in the Stanford Sentiment Treebank[1] (Socher et al., 2013).
- **SST-2** The movie reviews with binary classes. It is also from the Stanford Sentiment Treebank.
- **SUBJ** The movie reviews with labels of subjective or objective (Pang and Lee, 2004).
- **IMDB** The IMDB dataset[2] consists of 100,000 movie reviews with binary classes (Maas et al., 2011). One key aspect of this dataset is that each movie review has several sentences.

[1] http://nlp.stanford.edu/sentiment.

[2] http://ai.stanford.edu/~amaas/data/sentiment/

Model		SST-1	SST-2	SUBJ	IMDB	AvgΔ
Single Task	LSTM	45.9	85.8	91.6	88.5	-
	ME-LSTM	46.4	85.5	91.0	88.7	-
Multi-task	ARC-I	48.6	87.0	93.8	89.8	+(1.8/1.9)
	ARC-II	**49.5**	**87.8**	**95.0**	**91.2**	+(2.9/3.0)
	MT-CNN	46.7	86.1	92.2	88.4	-
	MT-DNN	44.5	84.0	90.1	85.6	-
NBOW		42.4	80.5	91.3	83.6	-
RAE (Socher et al., 2011)		43.2	82.4	-	-	-
MV-RNN (Socher et al., 2012)		44.4	82.9	-	-	-
RNTN (Socher et al., 2013)		45.7	85.4	-	-	-
DCNN (Kalchbrenner et al., 2014)		48.5	86.8	-	89.3	-
CNN-multichannel (Kim, 2014)		47.4	**88.1**	93.2	-	-
Tree-LSTM (Tai et al., 2015)		**50.6**	86.9	-	-	-

Table 3: Accuracies of our models on movie reviews tasks against state-of-the-art neural models. The last column gives the improvements relative to LSTM and ME-LSTM respectively. **NBOW**: Sums up the word vectors and applies a non-linearity followed by a softmax classification layer. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with dynamic k-max pooling (Kalchbrenner et al., 2014; Denil et al., 2014). **CNN-multichannel**: Convolutional Neural Network (Kim, 2014). **Tree-LSTM**: A generalization of LSTMs to tree-structured network topologies (Tai et al., 2015).

Product Reviews This dataset[3], constructed by Blitzer et al. (2007), contains Amazon product reviews from four different domains: Books, DVDs, Electronics and Kitchen appliances. The goal in each domain is to classify a product review as either positive or negative. The datasets in each domain are partitioned randomly into training data, development data and testing data with the proportion of 70%, 20% and 10% respectively.

5.2 Competitor Methods for Multi-task Learning

The multi-task frameworks proposed by previous works are various while not all can be applied to the tasks we focused. Nevertheless, we chose two most related neural models for multi-task learning and implement them as strong competitor methods .

- MT-CNN: This model is proposed by Collobert and Weston (2008) with convolutional layer, in which lookup-tables are shared partially while other layers are task-specific.
- MT-DNN: The model is proposed by Liu et al. (2015b) with bag-of-words input and multi-layer perceptrons, in which a hidden layer is shared.

[3] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

5.3 Hyperparameters and Training

The networks are trained with backpropagation and the gradient-based optimization is performed using the Adagrad update rule (Duchi et al., 2011).

The word embeddings for all of the models are initialized with the 100d GloVe vectors (840B token version, (Pennington et al., 2014)) and fine-tuned during training to improve the performance. The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$. The mini-batch size is set to 16.

For each task, we take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.1, 0.01]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5]$. For datasets without development set, we use 10-fold cross-validation (CV) instead. The final hyper-parameters are set as Table 2.

5.4 Multi-task Learning of Movie Reviews

We first compare our proposed models with the baseline system for single task classification. Table 3 shows the classification accuracies on the movie reviews dataset. The row of "Single Task" shows the results of LSTM and ME-LSTM for each individual task. With the help of multi-task learning, the performances of these four tasks are improved by 1.8% (ARC-I) and 2.9% (ARC-II) on average relative to LSTM. We can find that the architecture of local-global hybrid external memory has better performances. The reason is that the global memory in ARC-I could store some task-specific information besides shared information, which maybe noisy to other tasks. Moreover, both of our proposed models outperform MT-CNN and MT-DNN, which indicates the effectiveness of our proposed shared mechanism. To give an intuitive evaluation of these results, we also list the following state-of-the-art neural models. With the help of utilizing the shared information of several related tasks, our results outperform most of state-of-the-art models. Although Tree-LSTM outperforms our method on SST-1, it needs an external parser to get the sentence topological structure. It is worth noticing that our models are generic and compatible with the other LSTM based models. For example, we can easily extend our models to incorporate the Tree-LSTM model.

5.5 Multi-task Learning of Product Reviews

Table 4 shows the classification accuracies on the tasks of product reviews. The row of "Single Task" shows the results of the baseline for each individual task. With the help of global shared memory (ARC-I), the performances of these four tasks are improved by an average of 2.9%(2.6%) compared with LSTM(ME-LSTM). ARC-II achieves best performances on three sub-tasks, and its average improvement is 3.7%(3.5%). Compared with MT-CNN and MT-DNN, our models achieve a better performance. We think the reason is that our models can not only share lexical information but share complicated patterns of sentences by reading/writing operations of external memory. Furthermore, these results on product reviews are consistent with that on movie reviews, which shows our architectures are robust.

5.6 Case Study

To get an intuitive understanding of what is happening when we use shared memory to predict the class of text, we design an experiment to compare and analyze the difference between our models and vanilla LSTM, thereby demonstrating the effectiveness of our proposed architectures.

We sample two sentences from the SST-2 validation dataset, and the changes of the predicted sentiment score at different time steps are shown in Figure 3, which are obtained by vanilla LSTM and ARC-I respectively. Additionally, both models are bidirectional for better visualization. To get more insights into how the shared external memory influences the specific task, we plot and observe the evolving activation of fusion gates through time, which controls signals flowing from a shared external memory to task-specific output, to understand the behaviour of neurons.

For the sentence "*It is a cookie-cutter movie, a cut-and-paste job.*", which has a negative sentiment, while the standard LSTM gives a wrong prediction due to not understanding the informative words "*cookie-cutter*" and "*cut-and-paste*".

In contrast, our model makes a correct prediction and the reason can be inferred from the activation of fusion gates. As shown in Figure 3-(c), we can see clearly the neurons are activated much when they take input as "*cookie-cutter*" and "*cut-and-paste*", which indicates much information in shared memory has be passed into LSTM, therefore enabling the model to give a correct prediction.

Another case "*If you were not nearly moved to tears by a couple of scenes , you 've got ice water in your veins*", a subjunctive clause introduced by "*if*", has a positive sentiment.

As shown in Figure 3-(b,d), vanilla LSTM failed to capture the implicit meaning behind the sentence, while our model is sensitive to the pattern "*If ... were not ...*" and has an accurate understanding of the sentence, which indicates the shared memory mechanism can not only enrich the meaning of certain words, but teach some information of sentence structure to specific task.

Model		Books	DVDs	Electronics	Kitchen	AvgΔ
Single Task	LSTM	78.0	79.5	81.2	81.8	-
	ME-LSTM	77.5	80.2	81.5	82.1	-
Multi-task	ARC-I	81.2	82.0	84.5	**84.3**	+(2.9/2.6)
	ARC-II	**82.8**	**83.0**	**85.5**	84.0	+(3.7/3.5)
	MT-CNN	80.2	81.0	83.4	83.0	-
	MT-DNN	79.7	80.5	82.5	82.8	-

Table 4: Accuracies of our models on product reviews dataset. The last column gives the improvement relative to LSTM and ME-LSTM respectively.

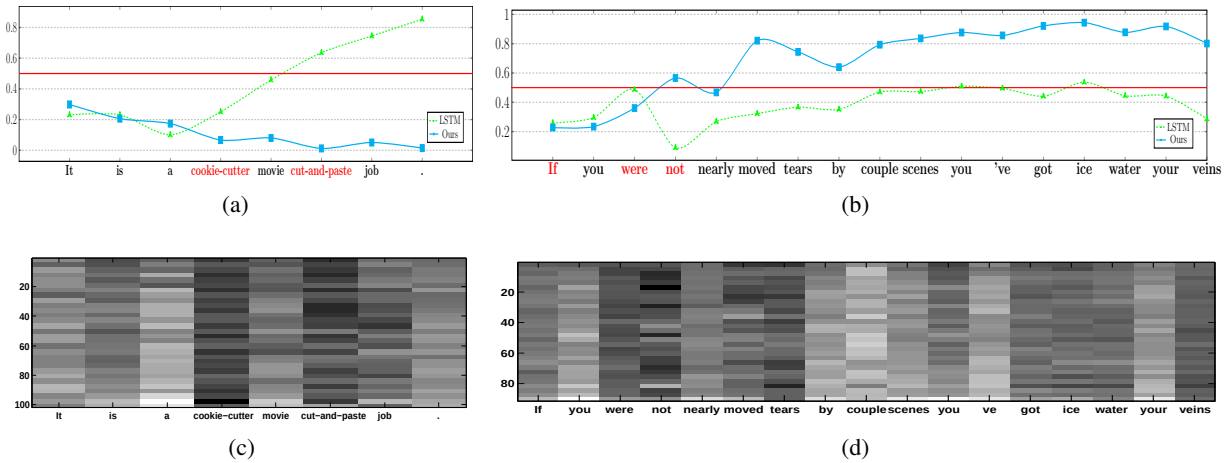


Figure 3: (a)(b) The change of the predicted sentiment score at different time steps. Y-axis represents the sentiment score, while X-axis represents the input words in chronological order. The red horizontal line gives a border between the positive and negative sentiments. (c)(d) Visualization of the fusion gate's activation.

6 Related Work

Neural networks based multi-task learning has been proven effective in many NLP problems (Collobert and Weston, 2008; Glorot et al., 2011; Liu et al., 2015b; Liu et al., 2016b). In most of these models, the lower layers are shared across all tasks, while top layers are task-specific.

Collobert and Weston (2008) used a shared representation for input words and solved different traditional NLP tasks within one framework. However, only one lookup table is shared, and the other lookup tables and layers are task-specific.

Liu et al. (2015b) developed a multi-task DNN for learning representations across multiple tasks. Their multi-task DNN approach combines tasks of query classification and ranking for web search. But the input of the model is bag-of-word representation, which loses the information of word order.

More recently, several multi-task encoder-decoder networks were also proposed for neural machine translation (Dong et al., 2015; Luong et al., 2015; Firat et al., 2016), which can make use of cross-lingual information.

Unlike these works, in this paper we design two neural architectures with shared memory for multi-task learning, which can store useful information across the tasks. Our architectures are relatively loosely coupled, and therefore more flexible to expand. With the help of shared memory, we can obtain better task-specific sentence representation by utilizing the knowledge obtained by other related tasks.

7 Conclusion and Future Work

In this paper, we introduce two deep architectures for multi-task learning. The difference with the previous models is the mechanisms of sharing information among several tasks. We design an external

memory to store the knowledge shared by several related tasks. Experimental results show that our models can improve the performances of several related tasks by exploring common features.

In addition, we also propose a deep fusion strategy to integrate the information from the external memory into task-specific LSTM with a fusion gate.

In future work, we would like to investigate the other sharing mechanisms of neural network based multi-task learning.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Marcin Andrychowicz and Karol Kurach. 2016. Learning efficient algorithms with hierarchical attentive memory. *arXiv preprint arXiv:1602.03218*.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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

Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. 2016. Associative long short-term memory. *CoRR*, abs/1602.03032.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015a. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015b. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the ACL*, pages 142–150.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

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

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

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Natural Language Comprehension with the EpiReader

Adam Trischler adam.trischler
Zheng Ye jeff.ye
Xingdi Yuan eric.yuan
Philip Bachman phil.bachman
Alessandro Sordoni alessandro.sordoni
Kaheer Suleman k.suleman
@maluuba.com
Maluuba Research
Montréal, Québec, Canada

Abstract

We present EpiReader, a novel model for machine comprehension of text. Machine comprehension of unstructured, real-world text is a major research goal for natural language processing. Current tests of machine comprehension pose questions whose answers can be inferred from some supporting text, and evaluate a model's response to the questions. EpiReader is an end-to-end neural model comprising two components: the first component proposes a small set of candidate answers after comparing a question to its supporting text, and the second component formulates hypotheses using the proposed candidates and the question, then reranks the hypotheses based on their estimated concordance with the supporting text. We present experiments demonstrating that EpiReader sets a new state-of-the-art on the CNN and Children's Book Test benchmarks, outperforming previous neural models by a significant margin.

1 Introduction

When humans reason about the world, we tend to formulate a variety of hypotheses and counterfactuals, then test them in turn by physical or thought experiments. The philosopher Epicurus first formalized this idea in his Principle of Multiple Explanations: if several theories are consistent with the observed data, retain them all until more data is observed. In this paper, we argue that the same principle can be applied to machine comprehension of natural language. We propose a deep neural comprehension model, trained end-to-end, that we call EpiReader.

Comprehension of natural language by machines, at a near-human level, is a prerequisite for an extremely broad class of useful applications of artificial intelligence. Indeed, most human knowledge is collected in the natural language of text. Machine comprehension (MC) has therefore garnered significant attention from the machine learning research community. Machine comprehension is typically evaluated by posing a set of questions based on a supporting text passage, then scoring a system's answers to those questions. Such tests are objectively gradable and may assess a range of abilities, from basic understanding to causal reasoning to inference (Richardson et al., 2013).

In the past year, two large-scale MC datasets have been released: the CNN/Daily Mail corpus, consisting of news articles from those outlets (Hermann et al., 2015), and the Children's Book Test (CBT), consisting of short excerpts from books available through Project Gutenberg (Hill et al., 2016). The size of these datasets (on the order of 10^5 distinct questions) makes them amenable to data-intensive deep learning techniques. Both corpora use Cloze-style questions (Taylor, 1953), which are formulated by replacing a word or phrase in a given sentence with a placeholder token. The task is then to find the answer that "fills in the blank".

In tandem with these corpora, a host of neural machine comprehension models has been developed (Weston et al., 2015b; Hermann et al., 2015; Hill et al., 2016; Kadlec et al., 2016; Chen et al., 2016). We compare EpiReader to these earlier models through training and evaluation on the CNN and

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 128–137,
Austin, Texas, November 1-5, 2016.

CBT datasets.[1]

EpiReader factors into two components. The first component extracts a small set of potential answers based on a shallow comparison of the question with its supporting text; we call this the *Extractor*. The second component reranks the proposed answers based on deeper semantic comparisons with the text; we call this the *Reasoner*. We can summarize this process as *Extract* $\rightarrow$ *Hypothesize* $\rightarrow$ *Test*[2]. The semantic comparisons implemented by the Reasoner are based on the concept of *recognizing textual entailment* (RTE) (Dagan et al., 2006), also known as natural language inference. This process is computationally demanding. Thus, the Extractor serves the important function of filtering a large set of potential answers down to a small, tractable set of likely candidates for more thorough testing. The two-stage process is an analogue of *structured prediction cascades* (Weiss and Taskar, 2010), wherein a sequence of increasingly complex models progressively filters the output space in order to trade off between model complexity and limited computational resources. We demonstrate that this cascade-like framework is applicable to machine comprehension and can be trained end-to-end with stochastic gradient descent.

The Extractor follows the form of a pointer network (Vinyals et al., 2015), and uses a differentiable attention mechanism to indicate words in the text that potentially answer the question. This approach was used (on its own) for question answering with the Attention Sum Reader (Kadlec et al., 2016). The Extractor outputs a small set of answer candidates along with their estimated probabilities of correctness. The Reasoner forms hypotheses by inserting the candidate answers into the question, then estimates the concordance of each hypothesis with each sentence in the supporting text. We use these estimates as a measure of the evidence for a hypothesis, and aggregate evidence over all sentences. In the end, we combine the Reasoner's evidence with the Extractor's probability estimates to produce a final ranking of the answer candidates.

[1]The CNN and Daily Mail datasets were released together and have the same form. The Daily Mail dataset is significantly larger; therefore, models consistently score higher when trained/tested on it.

[2]The Extractor performs extraction, while the Reasoner both hypothesizes and tests.

This paper is organized as follows. In Section 2 we formally define the problem to be solved and give some background on the datasets used in our tests. In Section 3 we describe EpiReader, focusing on its two components and how they combine. Section 4 discusses related work, and Section 5 details our experimental results and analysis. We conclude in Section 6.

2 Problem definition, notation, datasets

EpiReader's task is to answer a Cloze-style question by reading and comprehending a supporting passage of text. The training and evaluation data consist of tuples $(\mathcal{Q}, \mathcal{T}, a^*, A)$, where $\mathcal{Q}$ is the question (a sequence of words $\{q_1, ...q_{|\mathcal{Q}|}\}$), $\mathcal{T}$ is the text (a sequence of words $\{t_1, ..., t_{|\mathcal{T}|}\}$), A is a set of possible answers $\{a_1, ..., a_{|A|}\}$, and $a^* \in A$ is the correct answer. All words come from a vocabulary V, and $A \subset \mathcal{T}$. In each question, there is a placeholder token indicating the missing word to be filled in.

2.1 Datasets

CNN This corpus is built using articles scraped from the CNN website. The articles themselves form the text passages, and questions are generated synthetically from short summary statements that accompany each article. These summary points are (presumably) written by human authors. Each question is created by replacing a named entity in a summary point with a placeholder token. All named entities in the articles and questions are replaced with anonymized tokens that are shuffled for each $(\mathcal{Q}, \mathcal{T})$ pair. This forces the model to rely only on the text, rather than learning world knowledge about the entities during training. The CNN corpus (henceforth CNN) was presented by Hermann et al. (2015).

Children's Book Test This corpus is constructed similarly to CNN, but from children's books available through Project Gutenberg. Rather than articles, the text passages come from book excerpts of 20 sentences. Since no summaries are provided, a question is generated by replacing a single word in the next (i.e. 21st) sentence. The corpus distinguishes questions based on the type of word that is replaced: named entity, common noun, verb, or preposition. Like Kadlec et al. (2016), we focus only on the first two classes since Hill et al. (2016) showed that stan-

dard LSTM language models already achieve human-level performance on the latter two. Unlike in the CNN corpora, named entities are not anonymized and shuffled in the Children's Book Test (CBT). CBT was presented by Hill et al. (2016).

The different methods of construction for questions in each corpus mean that CNN and CBT assess different aspects of comprehension. The summary points of CNN are a condensed paraphrasing of information in the text; thus, determining the correct answer relies mostly on recognizing textual entailment. On the other hand, CBT is about story prediction. It is a comprehension task insofar as comprehension is likely necessary for story prediction, but comprehension alone may not be sufficient. Indeed, there are some CBT questions that are unanswerable given the preceding context.

3 EpiReader

3.1 Overview and intuition

EpiReader explicitly leverages the observation that the answer to a question is often a word or phrase from the related text passage. This condition holds for the CNN and CBT datasets. EpiReader's first module, the Extractor, can thus select a small set of candidate answers by pointing to their locations in the supporting passage. This mechanism is detailed in Section 3.2, and was used previously by the Attention Sum Reader (Kadlec et al., 2016). Pointing to candidate answers removes the need to apply a softmax over the entire vocabulary as in Weston et al. (2015b), which is computationally more costly and uses less-direct information about the context of a predicted answer in the supporting text.

EpiReader's second module, the Reasoner, begins by formulating hypotheses using the extracted answer candidates. It generates each hypothesis by replacing the placeholder token in the question with an answer candidate. Cloze-style questions are ideally-suited to this process, because inserting the correct answer at the placeholder location produces a well-formed, grammatical statement. Thus, the correct hypothesis will "make sense" to a language model.

The Reasoner then tests each hypothesis individually. It compares a hypothesis to the text, split into sentences, to measure textual entailment, and then aggregates entailment over all sentences. This computation uses a pair of convolutional encoder networks followed by a recurrent neural network. The convolutional encoders generate abstract representations of the hypothesis and each text sentence; the recurrent network estimates and aggregates entailment. This is described formally in Section 3.3. The end-to-end EpiReader model, combining the Extractor and Reasoner modules, is depicted in Figure 1.

Throughout our model, words will be represented with trainable embeddings (Bengio et al., 2000). We represent these embeddings using a matrix $\mathbf{W} \in \mathbb{R}^{D \times |V|}$, where D is the embedding dimension and $|V|$ is the vocabulary size.

3.2 The Extractor

The Extractor is a Pointer Network (Vinyals et al., 2015). It uses a pair of bidirectional recurrent neural networks, $f(\theta_T, \mathbf{T})$ and $g(\theta_Q, \mathbf{Q})$, to encode the text passage and the question. θ_T represents the parameters of the text encoder, and $\mathbf{T} \in \mathbb{R}^{D \times N}$ is a matrix representation of the text (comprising N words), whose columns are individual word embeddings $\mathbf{t}_i$. Likewise, θ_Q represents the parameters of the question encoder, and $\mathbf{Q} \in \mathbb{R}^{D \times N_Q}$ is a matrix representation of the question (comprising N_Q words), whose columns are individual word embeddings $\mathbf{q}_j$.

We use a recurrent neural network with gated recurrent units (GRU) (Bahdanau et al., 2015) to scan over the columns (i.e. word embeddings) of the input matrix. We selected the GRU because it is computationally simpler than Long Short-Term Memory (Hochreiter and Schmidhuber, 1997), while still avoiding the problem of vanishing/exploding gradients often encountered when training recurrent networks.

The GRU's hidden state gives a representation of the ith word conditioned on preceding words. To include context from proceeding words, we run a second GRU over $\mathbf{T}$ in the reverse direction. We refer to the combination as a biGRU. At each step the biGRU outputs two d-dimensional encoding vectors, one for the forward direction and one for the backward direction. We concatenate these to yield a vector $f(\mathbf{t}_i) \in \mathbb{R}^{2d}$. The question biGRU is similar, but we form a single-vector representation of the question by concatenating the final forward state with the final backward state, which we denote $g(\mathbf{Q}) \in \mathbb{R}^{2d}$.

As in Kadlec et al. (2016), we model the probability that the ith word in text $\mathcal{T}$ answers question $\mathcal{Q}$

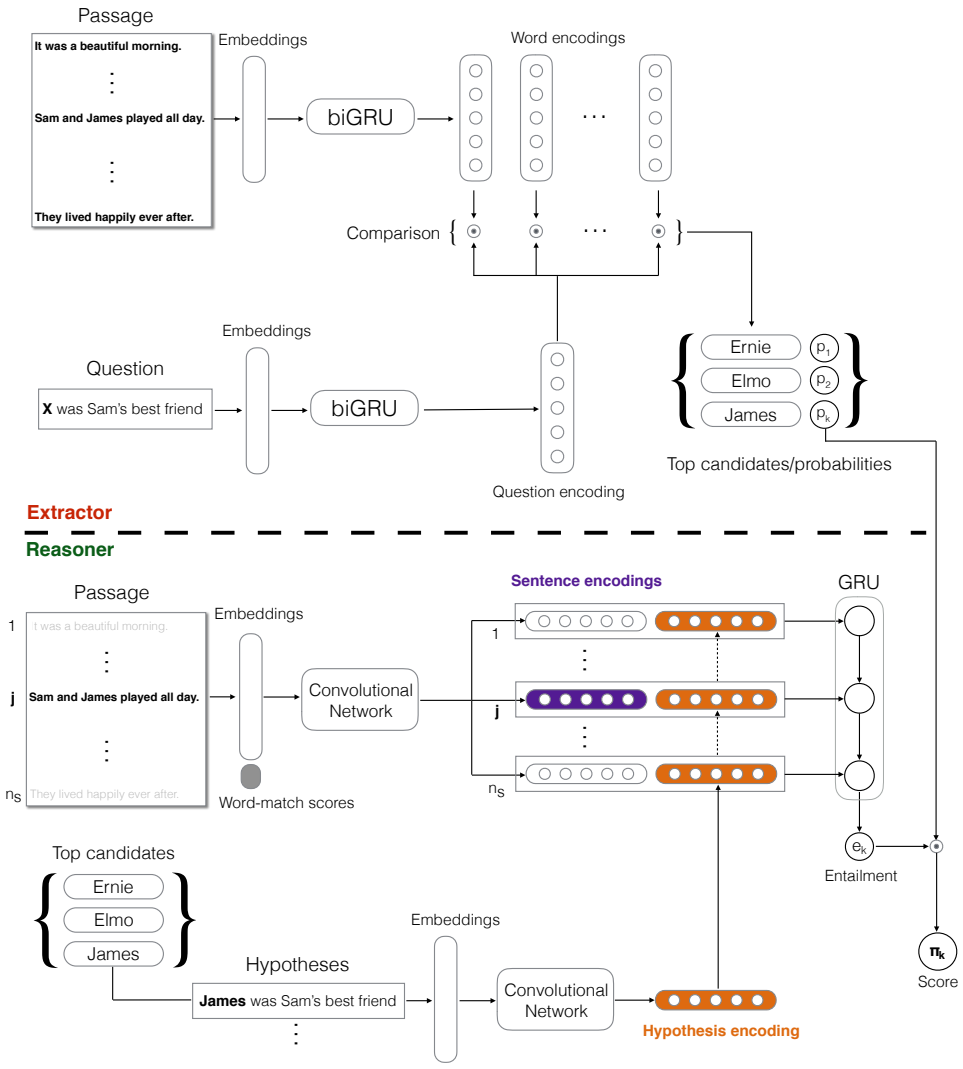


Figure 1: The complete EpiReader framework. The Extractor is above, the Reasoner below. Propagating the Extractor's probability estimates forward and combining them with the Reasoner's entailment estimates renders the model end-to-end differentiable.

using

$$s_i \propto \exp(f(\mathbf{t}_i) \cdot g(\mathbf{Q})), \tag{1}$$

which takes the inner product of the text and question representations followed by a softmax. In many cases unique words repeat in a text. Therefore, we compute the total probability that word w is the correct answer using a sum:

$$P(w \,|\, \mathcal{T}, \mathcal{Q}) = \sum_{i:\ t_i = w} s_i. \tag{2}$$

This probability is evaluated for each unique word in $\mathcal{T}$. Finally, the Extractor outputs the set $\{p_1, ..., p_K\}$ of the K highest word probabilities from 2, along with the corresponding set of K most probable answer words $\{\hat{a}_1, ..., \hat{a}_K\}$.

3.3 The Reasoner

The indicial selection involved in gathering $\{\hat{a}_1, ..., \hat{a}_K\}$, which is equivalent to a K-best $\arg\max$, is not a continuous function of its inputs. To construct an end-to-end differentiable model, we bypass this by propagating the probability estimates of the Extractor directly through the Reasoner.

The Reasoner begins by inserting the answer candidates, which are single words or phrases, into the question sequence $\mathcal{Q}$ at the placeholder location. This forms K hypotheses $\{\mathcal{H}_1, ..., \mathcal{H}_K\}$. At this

point, we consider each hypothesis to have probability $p(\mathcal{H}_k) \approx p_k$, as estimated by the Extractor. The Reasoner updates and refines this estimate.

The hypotheses represent new information in some sense—they are statements we have constructed, albeit from words already present in the question and text passage. The Reasoner estimates entailment between the statements $\mathcal{H}_k$ and the passage $\mathcal{T}$. We denote these estimates using $e_k = F(\mathcal{H}_k, \mathcal{T})$, with F to be defined. We start by reorganizing $\mathcal{T}$ into a sequence of N_s sentences: $\mathcal{T} = \{t_1, \ldots, t_N\} \rightarrow \{\mathcal{S}_1, \ldots, \mathcal{S}_{N_s}\}$, where $\mathcal{S}_i$ is a sequence of words.

For each hypothesis and each sentence of the text, Reasoner input consists of two matrices: $\mathbf{S}_i \in \mathbb{R}^{D \times |\mathcal{S}_i|}$, whose columns are the embedding vectors for each word of sentence $\mathcal{S}_i$, and $\mathbf{H}_k \in \mathbb{R}^{D \times |\mathcal{H}_k|}$, whose columns are the embedding vectors for each word in the hypothesis $\mathcal{H}_k$. The embedding vectors themselves come from matrix $\mathbf{W}$, as before.

These matrices feed into a convolutional architecture based on that of Severyn and Moschitti (2016). The architecture first augments $\mathbf{S}_i$ with matrix $\mathbf{M} \in \mathbb{R}^{2 \times |\mathcal{S}_i|}$. The first row of $\mathbf{M}$ contains the inner product of each word embedding in the sentence with the candidate answer embedding, and the second row contains the maximum inner product of each sentence word embedding with any word embedding in the question. These word-matching features were inspired by similar approaches in Wang and Jiang (2016) and Trischler et al. (2016), where they were shown to improve entailment estimates.

The augmented $\mathbf{S}_i$ is then convolved with a bank of filters $\mathbf{F}^S \in \mathbb{R}^{(D+2) \times m}$, while $\mathbf{H}_k$ is convolved with filters $\mathbf{F}^H \in \mathbb{R}^{D \times m}$, where m is the convolutional filter width. We add a bias term and apply a nonlinearity (we use a ReLU) following the convolution. Maxpooling over the sequences then yields two vectors: the representation of the text sentence, $\mathbf{r}_{\mathcal{S}_i} \in \mathbb{R}^{N_F}$, and the representation of the hypothesis, $\mathbf{r}_{\mathcal{H}_k} \in \mathbb{R}^{N_F}$, where N_F is the number of filters.

We then compute a scalar similarity score between these vector representations using the bilinear form

$$\varsigma = \mathbf{r}_{\mathcal{S}_i}^T \mathbf{R} \mathbf{r}_{\mathcal{H}_k}, \tag{3}$$

where $\mathbf{R} \in \mathbb{R}^{N_F \times N_F}$ is a matrix of trainable parameters. We then concatenate the similarity score with the sentence and hypothesis representations to get a vector, $\mathbf{x}_{ik} = [\varsigma; \mathbf{r}_{\mathcal{S}_i}; \mathbf{r}_{\mathcal{H}_k}]^T$. There are more powerful models of textual entailment that could have been used in place of this convolutional architecture. We adopted the approach of Severyn and Moschitti (2016) for computational efficiency.

The resulting sequence of N_s vectors feeds into yet another GRU for synthesis, of hidden dimension d_S. Intuitively, it is often the case that evidence for a particular hypothesis is distributed over several sentences. For instance, if we hypothesize that *the football is in the park*, perhaps it is because one sentence tells us that *Sam picked up the football* and a later one tells us that *Sam ran to the park*.[3] The Reasoner synthesizes distributed information by running a GRU network over $\mathbf{x}_{ik}$, where i indexes sentences and represents the step dimension.[4] The final hidden state of the GRU is fed through a fully-connected layer, yielding a single scalar y_k. This value represents the collected evidence for $\mathcal{H}_k$ based on the text. In practice, the Reasoner processes all K hypotheses in parallel and the estimated entailment of each is normalized by a softmax, $e_k \propto \exp(y_k)$.

As pointed out in Kadlec et al. (2016), it is a strength of the pointer framework that it does not blend the representations that are being attended. Contrast this with typical attention mechanisms where such a blended representation is used downstream to make similarity comparisons with, *e.g.*, output vectors.

Differentiable attention mechanisms (as in Bahdanau et al. (2015), for example) typically blend internal representations together through a weighted sum, then use this 'blend' downstream for similarity comparisons. The pointer framework does not resort to this blending; Kadlec et al. (2016) explain that this is an advantage, since in comprehension tasks the goal is to select the correct answer among semantically similar candidates and more exact matching is necessary. The reranking function performed by the Reasoner entails this advantage, by examining the separate hypotheses individually without blending.

[3] This example is characteristic of the *bAbI* dataset (Weston et al., 2015a).

[4] Note a benefit of forming the hypothesis: it renders bidirectional aggregation unnecessary, since knowing both the question and the putative answer "closes the loop" the same way that a bidirectional encoding would.

3.4 Combining components

Finally, we combine the evidence from the Reasoner with the probability from the Extractor. We compute the output probability of each hypothesis, π_k, according to the product

$$\pi_k \propto e_k p_k, \tag{4}$$

whereby the evidence of the Reasoner can be interpreted as a correction to the Extractor probabilities, applied as an additive shift in log-space. We experimented with other combinations of the Extractor and Reasoner, but we found the multiplicative approach to yield the best performance.

After combining results from the Extractor and Reasoner to get the probabilities π_k described in Eq. 4, we optimize the parameters of the full EpiReader to minimize a cost comprising two terms, $\mathcal{L}_E$ and $\mathcal{L}_R$. The first term is a standard negative log-likelihood objective, which encourages the Extractor to rate the correct answer above other answers. This is the same loss term used in Kadlec et al. (2016). It is given by:

$$\mathcal{L}_E = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} \left[-\log P(a^* \,|\, \mathcal{T}, \mathcal{Q}) \right], \tag{5}$$

where $P(a^* \,|\, \mathcal{T}, \mathcal{Q})$ is as defined in Eq. 2, and a^* denotes the true answer. The second term is a margin-based loss on the end-to-end probabilities π_k. We define π^* as the probability π_k corresponding to the true answer word a^*. This term is given by:

$$\mathcal{L}_R = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} \left[\sum_{\hat{a}_i \in \{\hat{a}_1,\ldots,\hat{a}_K\} \setminus a^*} [\gamma - \pi^* + \pi_{\hat{a}_i}]_+ \right], \tag{6}$$

where γ is a margin hyperparameter, $\{\hat{a}_1, ..., \hat{a}_K\}$ is the set of K answers proposed by the Extractor, and $[x]_+$ indicates truncating x to be non-negative. Intuitively, this loss says that we want the end-to-end probability π^* for the correct answer to be at least γ larger than the probability $\pi_{\hat{a}_i}$ for any other answer proposed by the Extractor. During training, the correct answer is occasionally missed by the Extractor, especially in early epochs. We counter this issue by forcing the correct answer into the top K set while training. When evaluating the model on validation and test examples we rely fully on the top K answers proposed by the Extractor.

To get the final loss term $\mathcal{L}_{ER}$, minus ℓ_2 regularization terms on the model parameters, we take a weighted combination of $\mathcal{L}_E$ and $\mathcal{L}_R$:

$$\mathcal{L}_{ER} = \mathcal{L}_E + \lambda \mathcal{L}_R, \tag{7}$$

where λ is a hyperparameter for weighting the relative contribution of the Extractor and Reasoner losses. In practice, we found that λ should be fairly large (*e.g.*, $10 < \lambda < 100$). Empirically, we observed that the output probabilities from the Extractor often peak and saturate the first softmax; hence, the Extractor term can come to dominate the Reasoner term without the weight λ (we discuss the Extractor's propensity to overfit in Section 5).

4 Related Work

The Impatient and Attentive Reader models were proposed by Hermann et al. (2015). The Attentive Reader applies bidirectional recurrent encoders to the question and supporting text. It then uses the attention mechanism described in Bahdanau et al. (2015) to compute a fixed-length representation of the text based on a weighted sum of the text encoder's output, guided by comparing the question representation to each location in the text. Finally, a joint representation of the question and supporting text is formed by passing their separate representations through a feed-forward MLP and an answer is selected by comparing the MLP output to a representation of each possible answer. The Impatient Reader operates similarly, but computes attention over the text after processing each consecutive word of the question. The two models achieved similar performance on the CNN and Daily Mail datasets.

Memory Networks were first proposed by Weston et al. (2015b) and later applied to machine comprehension by Hill et al. (2016). This model builds fixed-length representations of the question and of windows of text surrounding each candidate answer, then uses a weighted-sum attention mechanism to combine the window representations. As in the previous Readers, the combined window representation is then compared with each possible answer to form a prediction about the best answer. What distinguishes Memory Networks is how they construct the question and text window representations. Rather than a recurrent network, they use a specially-designed, trainable transformation of the word embeddings.

Most of the details for the very recent AS Reader are provided in the description of our Extractor module in Section 3.2, so we do not summarize it further here. This model (Kadlec et al., 2016) set the previous state-of-the-art on the CBT dataset.

During the write-up of this paper, another very recent model came to our attention. Chen et al. (2016) propose using a bilinear term instead of a $\tanh$ layer to compute the attention between question and passage words, and also uses the attended word encodings for direct, pointer-style prediction as in Kadlec et al. (2016). This model set the previous state-of-the-art on the CNN dataset. However, this model used embedding vectors pretrained on a large external corpus (Pennington et al., 2014).

EpiReader borrows ideas from other models as well. The Reasoner's convolutional architecture is based on Severyn and Moschitti (2016) and Blunsom et al. (2014). Our use of word-level matching was inspired by the Parallel-Hierarchical model of Trischler et al. (2016) and the natural language inference model of Wang and Jiang (2016). Finally, the idea of formulating and testing hypotheses for question-answering was used to great effect in IBM's DeepQA system for *Jeopardy!* (Ferrucci et al., 2010) (although that was a more traditional information retrieval pipeline rather than an end-to-end neural model), and also resembles the framework of structured prediction cascades (Weiss and Taskar, 2010).

5 Evaluation

5.1 Implementation and training details

To train our model we used stochastic gradient descent with the ADAM optimizer (Kingma and Ba, 2015), with an initial learning rate of 0.001. The word embeddings were initialized randomly, drawing from the uniform distribution over $[-0.05, 0.05)$. We used batches of 32 examples, and early stopping with a patience of 2 epochs. Our model was implemented in Theano (Bergstra et al., 2010) using the Keras framework (Chollet, 2015).

The results presented below for EpiReader were obtained by searching over a small grid of hyperparameter settings. We selected the model that, on each dataset, maximized accuracy on the validation set, then evaluated it on the test set. We record the best settings for each dataset in Table 1. As has been done previously, we train separate models on CBT's named entity (CBT-NE) and common noun (CBT-CN) splits. All our models used ℓ_2-regularization at 0.001, $\lambda = 50$, and $\gamma = 0.04$. We did not use dropout but plan to investigate its effect in the future. Hill et al. (2016) and Kadlec et al. (2016) also present results for ensembles of their models. Time did not permit us to generate an ensemble of EpiReaders on the CNN dataset so we omit those measures; however, EpiReader ensembles (of seven models) demonstrated improved performance on the CBT dataset.

Table 1: Hyperparameter settings for best EpiReaders. D is the embedding dimension, d is the hidden dimension in the Extractor GRUs, K is the number of candidates to consider, m is the filter width, N_F is the number of filters, and d_S is the hidden dimension in the Reasoner GRU.

	Hyperparameters					
Dataset	D	d	K	m	N_F	d_S
CBT-NE	300	128	5	3	16	32
CBT-CN	300	128	5	3	32	32
CNN	384	256	10	3	32	32

5.2 Results

In Table 5.2, we compare the performance of EpiReader against that of several baselines, on the validation and test sets of the CBT and CNN corpora. We measure EpiReader performance at the output of both the Extractor and the Reasoner. EpiReader achieves state-of-the-art performance across the board for both datasets. On CNN, we score 2.2% higher on test than the best previous model of Chen et al. (2016). Interestingly, an analysis of the CNN dataset by Chen et al. (2016) suggests that approximately 25% of the test examples contain coreference errors or questions which are "ambiguous/hard" even for a human analyst. If this estimate is accurate, then EpiReader, achieving an absolute test accuracy of 74.0%, is operating close to expected human performance. On the other hand, ambiguity is unlikely to be distributed evenly over entities, so a good model should be able to perform at better-than-chance levels even on questions where the correct answer is uncertain. If, on the 25% of "noisy" questions, the model can shift its hit rate from, *e.g.*, 1/10 to 1/3, then there is still a fair amount of performance to gain.

Model	CBT-NE valid	CBT-NE test	CBT-CN valid	CBT-CN test
Humans (context + query) [1]	-	81.6	-	81.6
LSTMs (context + query) [1]	51.2	41.8	62.6	56.0
MemNNs [1]	70.4	66.6	64.2	63.0
AS Reader [2]	73.8	68.6	68.8	63.4
EpiReader Extractor	73.2	69.4	69.9	66.7
EpiReader	**75.3**	**69.7**	**71.5**	**67.4**
AS Reader (ensemble) [2]	74.5	70.6	71.1	68.9
EpiReader (ensemble)	**76.6**	**71.8**	**73.6**	**70.6**

Model	CNN valid	CNN test
Deep LSTM Reader [3]	55.0	57.0
Attentive Reader [3]	61.6	63.0
Impatient Reader [3]	61.8	63.8
MemNNs [1]	63.4	66.8
AS Reader [2]	68.6	69.5
Stanford AR [4]	72.4	72.4
EpiReader Extractor	71.8	72.0
EpiReader	**73.4**	**74.0**

Table 2: Model comparison on the CBT and CNN datasets. Results marked with [1] are from Hill et al. (2016), with [2] are from Kadlec et al. (2016), with [3] are from Hermann et al. (2015), and with [4] are from Chen et al. (2016).

Ablated component	Validation accuracy (%)
-	**71.5**
Word-match scores	70.3
Bilinear similarity	70.0
Reasoner	68.7
Convolutional encoders	71.0

Table 3: Ablation study on CBT-CN validation set.

On CBT-CN our single model scores 4.0% higher than the previous best of the AS Reader. The improvement on CBT-NE is more modest at 1.1%. Looking more closely at our CBT-NE results, we found that the validation and test accuracies had relatively high variance even in late epochs of training. We discovered that many of the validation and test questions were asked about the same named entity, which may explain this issue.

5.3 Analysis

We measure the contribution of several components of the Reasoner by ablating them. Results on the validation set of CBT-CN are presented in Table 3. The word-match scores (cosine similarities stored in the first two rows of matrix $\mathbf{M}$, see Section 3.3) make a contribution of 1.2% to the validation performance, indicating that they are useful. Similarly, the bilinear similarity score ς, which is passed to the final GRU network, contributes 1.5%.

Removing the Reasoner altogether reduces our model to the AS Reader, whose results we have reproduced to within negligible difference. Aside from achieving state-of-the-art results at its final output, the EpiReader framework gives a boost to its Extractor component through the joint training process. This can be seen by referring back to Table 5.2, wherein we also provide accuracy scores evaluated at the output of the Extractor. These are all higher than the analogous scores reported for the AS Reader. Based on our own work with that model, we found it to overfit the training set rapidly and significantly, achieving training accuracy scores upwards of 98% after only 2 epochs. We suspect that the Reasoner module had a regularizing effect on the Extractor, but leave the confirmation for future work.

Although not exactly an ablation, we also tried bypassing the Reasoner's convolutional encoders altogether, along with the word-match scores and the bilinear similarity. This was done as follows: from the Extractor, we pass to the Reasoner's final GRU (i) the bidirectional hidden representation of the question; (ii) the bidirectional hidden representations of the *end* of each story sentence (recall that the Reasoner operates on sentence representations). Thus, we reuse (parts of) the original biGRU encodings. This cuts down on the number of model parameters and on the length of the graph through which gradients must flow, potentially providing a stronger learning signal to the initial encoders. We found that this change yielded a relatively small reduction in performance on CBT-CN, perhaps for the reasons just discussed—only 0.5%, as given in the final line of

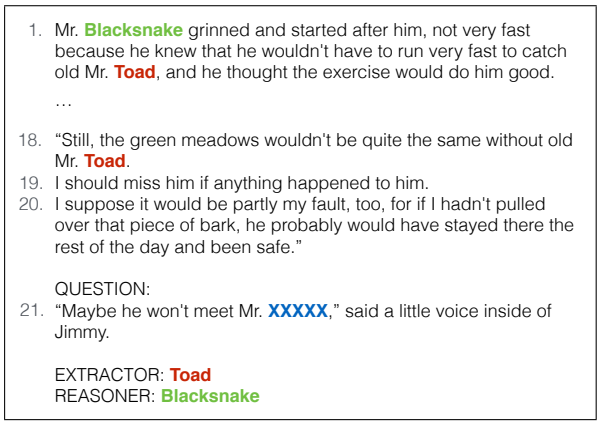


Figure 2: An abridged example from CBT-NE demonstrating corrective reranking by the Reasoner.

Table 3. This suggests that competitive performance may be achieved with other, simpler architectures for the Reasoner's entailment system and this will be the subject of future research.

An analysis by Kadlec et al. (2016) indicates that the trained AS Reader includes the correct answer among its five most probable candidates on approximately 95% of test examples for both datasets. We verified that our Extractor achieved a similar rate, and of course this is vital for performance of the full system, since the Reasoner cannot recover when the correct answer is not among its inputs.

Our results show that the Reasoner often corrects erroneous answers from the Extractor. Figure 2 gives an example of this correction. In the text passage, from CBT-NE, Mr. Blacksnake is pursuing Mr. Toad, presumably to eat him. The dialogue in the question sentence refers to both: Mr. Toad is its subject, referred to by the pronoun "he", and Mr. Blacksnake is its object. In the preceding sentences, it is clear (to a human) that Jimmy is worried about Mr. Toad and his potential encounter with Mr. Blacksnake. The Extractor, however, points most strongly to "Toad", possibly because he has been referred to most recently. The Reasoner corrects this error and selects "Blacksnake" as the answer. This relies on a deeper understanding of the text. The named entity can, in this case, be inferred through an alternation of the entities most recently referred to. This kind alternation is typical of dialogues, when two actors interact in turns. The Reasoner can capture this behavior because it examines sentences in sequence.

6 Conclusion

We presented the novel EpiReader framework for machine comprehension and evaluated it on two large, complex datasets: CNN and CBT. Our model achieves state-of-the-art results on these corpora, outperforming all previous approaches. In future work, we plan to test our framework with alternative models for natural language inference (*e.g.*, Wang and Jiang (2016)), and explore the effect of pretraining such a model specifically on an inference task.

As a general framework that consists in a two-stage cascade, EpiReader can be implemented using a variety of mechanisms in the Extractor and Reasoner stages. We have demonstrated that this cascade-like framework is applicable to machine comprehension and can be trained end-to-end. As more powerful machine comprehension models inevitably emerge, it may be straightforward to boost their performance using EpiReader's structure.

References

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

[Bengio et al.2000] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

[Bergstra et al.2010] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *In Proc. of SciPy*.

[Blunsom et al.2014] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences.

[Chen et al.2016] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn / daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

[Chollet2015] François Chollet. 2015. keras. `https://github.com/fchollet/keras`.

[Dagan et al.2006] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment*, pages 177–190. Springer.

[Ferrucci et al.2010] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

[Hermann et al.2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

[Hill et al.2016] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kadlec et al.2016] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

[Kingma and Ba2015] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. *Proc. EMNLP*, 12.

[Richardson et al.2013] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

[Severyn and Moschitti2016] Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

[Taylor1953] Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30.

[Trischler et al.2016] Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Philip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[Vinyals et al.2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

[Wang and Jiang2016] Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. *NAACL*.

[Weiss and Taskar2010] David J Weiss and Benjamin Taskar. 2010. Structured prediction cascades. In *AISTATS*, pages 916–923.

[Weston et al.2015a] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015a. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

[Weston et al.2015b] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. *ICLR*.

Creating Causal Embeddings for Question Answering with Minimal Supervision

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and **Michael Hammond**
University of Arizona, Allen Institute for Artificial Intelligence
{bsharp, msurdeanu, pajansen, hammond}@email.arizona.edu, PeterC@allenai.org

Abstract

A common model for question answering (QA) is that a good answer is one that is closely related to the question, where relatedness is often determined using general-purpose lexical models such as word embeddings. We argue that a better approach is to look for answers that are related to the question in a *relevant way*, according to the information need of the question, which may be determined through task-specific embeddings. With causality as a use case, we implement this insight in three steps. First, we generate causal embeddings cost-effectively by bootstrapping cause-effect pairs extracted from free text using a small set of seed patterns. Second, we train dedicated embeddings over this data, by using task-specific contexts, i.e., the context of a cause is its effect. Finally, we extend a state-of-the-art reranking approach for QA to incorporate these causal embeddings. We evaluate the causal embedding models both *directly* with a casual implication task, and *indirectly*, in a downstream causal QA task using data from Yahoo! Answers. We show that explicitly modeling causality improves performance in both tasks. In the QA task our best model achieves 37.3% P@1, significantly outperforming a strong baseline by 7.7% (relative).

1 Introduction

Question answering (QA), i.e., finding short answers to natural language questions, is one of the most important but challenging tasks on the road towards natural language understanding (Etzioni, 2011). A common approach for QA is to prefer answers that are closely related to the question, where relatedness is often determined using lexical semantic models such as word embeddings (Yih et al., 2013; Jansen et al., 2014; Fried et al., 2015). While appealing for its robustness to natural language variation, this one-size-fits-all approach does not take into account the wide range of distinct question types that can appear in any given question set, and that are best addressed individually (Chu-Carroll et al., 2004; Ferrucci et al., 2010; Clark et al., 2013).

Given the variety of question types, we suggest that a better approach is to look for answers that are related to the question *through the appropriate relation*, e.g., a causal question should have a cause-effect relation with its answer. If we adopt this view, and continue to work with embeddings as a mechanism for assessing relationship, this raises a key question: how do we train and use task-specific embeddings cost-effectively? Using causality as a use case, we answer this question with a framework for producing causal word embeddings with minimal supervision, and a demonstration that such task-specific embeddings significantly benefit causal QA.

In particular, the contributions of this work are:

(1) A methodology for generating causal embeddings cost-effectively by bootstrapping cause-effect pairs extracted from free text using a small set of seed patterns, e.g., *X causes Y*. We then train dedicated embedding (as well as two other distributional similarity) models over this data. Levy and Goldberg (2014) have modified the algorithm of Mikolov et al. (2013) to use an arbitrary, rather than linear, context. Here we make this context task-specific,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 138–148,
Austin, Texas, November 1-5, 2016.

i.e., the context of a cause is its effect. Further, to mitigate sparsity and noise, our models are bidirectional, and noise aware (by incorporating the likelihood of noise in the training process).

(2) The insight that QA benefits from task-specific embeddings. We implement a QA system that uses the above causal embeddings to answer questions and demonstrate that they significantly improve performance over a strong baseline. Further, we show that causal embeddings encode complementary information to vanilla embeddings, even when trained from the same knowledge resources.

(3) An analysis of direct vs. indirect evaluations for task-specific word embeddings. We evaluate our causal models both *directly*, in terms of measuring their capacity to rank causally-related word pairs over word pairs of other relations, as well as *indirectly* in the downstream causal QA task. In both tasks, our analysis indicates that including causal models significantly improves performance. However, from the direct evaluation, it is difficult to estimate which models will perform best in real-world tasks. Our analysis re-enforces recent observations about the limitations of word similarity evaluations (Faruqui et al., 2016): we show that they have limited coverage and may align poorly with real-world tasks.

2 Related Work

Addressing the need for specialized solving methods in QA, Oh et. al (2013) incorporate a dedicated causal component into their system, and note that it improves the overall performance. However, their model is limited by the need for lexical overlap between a causal construction found in their knowledge base and the question itself. Here, we develop a causal QA component that exploits specialized word embeddings to gain robustness to lexical variation.

There has been a vast body of work which demonstrates that word embeddings derived from distributional similarity are useful in many tasks, including question answering – see *inter alia* (Fried et al., 2015; Yih et al., 2013). However, Levy and Goldberg (2015) note that there are limitations on the type of semantic knowledge which is encoded in these general-purpose similarity embeddings. Therefore, here we build customized task-specific embeddings for causal QA.

Customized embeddings have been created for a variety of tasks, including semantic role labeling (FitzGerald et al., 2015; Woodsend and Lapata, 2015), and binary relation extraction (Riedel et al., 2013). Similar to Riedel et al., we train embeddings customized for specific relations, but we bootstrap training data using minimal supervision (i.e., a small set of patterns) rather than relying on distant supervision and large existing knowledge bases. Additionally, while Riedel et al. represent all relations in a general embedding space, here we train a dedicated embedding space for just the causal relations.

In QA, embeddings have been customized to have question words that are close to either their answer words (Bordes et al., 2014), or to structured knowledge base entries (Yang et al., 2014). While these methods are useful for QA, they do not distinguish between different types of questions, and as such their embeddings are not specific to a given question type.

Additionally, embeddings have been customized to distinguish functional similarity from relatedness (Levy and Goldberg, 2014; Kiela et al., 2015). In particular, Levy and Goldberg train their embeddings by replacing the standard linear context of the target word with context derived from the syntactic dependency graph of the sentence. In this work, we make use of this extension to arbitrary context in order to train our embeddings with contexts derived from binary causal relations. We extract cause-effect text pairs such that the cause text becomes the *target* text and the effect text serves as the *context*.

Recently, Faruqui et al.(2016) discussed issues surrounding the evaluation of similarity word embeddings, including the lack of correlation between their performance on word-similarity tasks and "downstream" or real-world tasks like QA, text classification, etc. As they advocate, in addition to a direct evaluation of our causal embeddings, we also evaluate them independently in a downstream QA task. We provide the same comparison for two alternative approaches (an alignment model and a convolutional neural network model), confirming that the direct evaluation performance can be misleading without the task-specific, downstream evaluation.

With respect to extracting causal relations from text, Girju et al. (2002) use modified Hearst patterns (Hearst, 1992) to extract a large number of potential cause-effect tuples, where both causes and effects must be nouns. However, Cole et al. (2005) show that these nominal-based causal relations account for a relatively small percentage of all causal relations, and for this reason, (Yang and Mao, 2014) allow for more elaborate argument structures in their causal extraction by identifying verbs, and then following the syntactic subtree of the verbal arguments to construct their candidate causes and effects. Additionally, Do et al. (2011) observe that nouns as well as verbs can signal causality. We follow these intuitions in developing our causal patterns by using both nouns and verbs to signal potential participants in causal relations, and then allowing for the entire dominated structures to serve as the cause and/or effect arguments.

3 Approach

Our focus is on reranking answers to causal questions using using task-specific distributional similarity methods. Our approach operates in three steps:

(1) We start by bootstrapping a large number of cause-effect pairs from free text using a small number of syntactic and surface patterns (Section 4).

(2) We then use these bootstrapped pairs to build several task-specific embedding (and other distributional similarity) models (Section 5). We evaluate these models directly on a causal-relation identification task (Section 6).

(3) Finally, we incorporate these models into a reranking framework for causal QA and demonstrate that the resulting approach performs better than the reranker without these task-specific models, even if trained on the same data (Section 7).

4 Extracting Cause-Effect Tuples

Because the success of embedding models depends on large training datasets (Sharp et al., 2015), and such datasets do not exist for open-domain causality, we opted to bootstrap a large number of cause-effect pairs from a small set of patterns. We wrote these patterns using Odin (Valenzuela-Escárcega et al., 2016), a rule-based information extraction framework which has the distinct advantage of being able to operate over multiple representations of content (i.e., surface and syntax). For this work, we make use of rules that operate over both surface sequences as well as dependency syntax in the grammars introduced in steps (2) and (3) below.

Odin operates as a cascade, allowing us to implement a two-stage approach. First, we identify potential participants in causal relations, i.e., the potential causes and effects, which we term **causal mentions (CM)**. A second grammar then identifies actual causal relations that take these CMs as arguments.

We consider both noun phrases (NP) as well as entire clauses to be potential CMs, since causal patterns form around participants that are syntactically more complex than flat NPs. For example, in the sentence *The collapse of the housing bubble caused stock prices to fall*, both the cause (*the collapse of the housing bubble*) and effect (*stock prices to fall*) are more complicated nested structures. Reducing these arguments to non-recursive NPs (e.g., *The collapse* and *stock prices*) is clearly insufficient to capture the relevant context.

Formally, we extract our causal relations using the following algorithm:

(1) Pre-processing: Much of the text we use to extract causal relation tuples comes from the Annotated Gigaword (Napoles et al., 2012). This text is already fully annotated and no further processing is necessary. We additionally use text from the Simple English Wikipedia[1], which we processed using the Stanford CoreNLP toolkit (Manning et al., 2014) and the dependency parser of Chen and Manning (2014).

(2) CM identification: We extract causal mentions (which are able to serve as arguments in our causal patterns) using a set of rules designed to be robust to the variety that exists in natural language. Namely, to find CMs that are noun phrases, we first find words that are tagged as nouns, then follow outgoing dependency links for modifiers and attached prepo-

[1] https://simple.wikipedia.org/wiki/Main_Page. The Simple English version was preferred over the full version due to its simpler sentence structures, which make extracting cause-effect tuples more straightforward.

Corpus	Extracted Tuples
Annotated Gigaword	798,808
Simple English Wikipedia	16,425
Total	815,233

Table 1: Number of causal tuples extracted from each corpus.

sitional phrases[2], to a maximum depth of two links. To find CMs that are clauses, we first find words that are tagged as verbs (excluding verbs which themselves were considered to signal causation[3]), then again follow outgoing dependency links for modifiers and arguments. We used a total of four rules to label CMs.

(3) Causal tuple extraction: After CMs are identified, a grammar scans the text for causal relations that have CMs as arguments. Different patterns have varying probabilities of signaling causation (Khoo et al., 1998). To minimize the noise in the extracted pairs, we restrict ourselves to a set of 13 rules designed to find unambiguously causal patterns, such as *CAUSE led to EFFECT*, where *CAUSE* and *EFFECT* are CMs. The rules operate by looking for a *trigger* phrase, e.g., *led*, and then following the dependency paths to and/or from the trigger phrase to see if all required CM arguments exist.

Applying this causal grammar over Gigaword and Simple English Wikipedia produced 815,233 causal tuples, as summarized in Table 1. As bootstrapping methods are typically noisy, we manually evaluated the quality of approximately 250 of these pairs selected at random. Of the tuples evaluated, approximately 44% contained some amount of noise. For example, from the sentence *Except for Springer's show, which still relies heavily on confrontational topics that lead to fistfights virtually every day...*, while ideally we would only extract (*confrontational topics* $\rightarrow$ *fistfights*), instead we extract the tuple (*show which still relies heavily on confrontational topics* $\rightarrow$ *fistfights virtually every day*), which contains a large amount of noise: *show, relies, heavily*, etc. This finding prompted our noise-aware model described at the end of Section 5.

[2]The outgoing dependency links from the nouns which we followed were: `nn`, `amod`, `advmod`, `ccmod`, `dobj`, `prep_of`, `prep_with`, `prep_for`, `prep_into`, `prep_on`, `prep_to`, and `prep_in`.

[3]The verbs we excluded were: *cause, result, lead, create*.

5 Models

We use the extracted causal tuples to train three distinct distributional similarity models that explicitly capture causality.

Causal Embedding Model (cEmbed): The first distributional similarity model we use is based on the skip-gram word-embedding algorithm of Mikolov et al. (2013), which has been shown to improve a variety of language processing tasks including QA (Yih et al., 2013; Fried et al., 2015). In particular, we use the variant implemented by Levy and Goldberg (2014) which modifies the original algorithm to use an arbitrary, rather than linear, context. Our novel contribution is to make this context task-specific: intuitively, the context of a cause is its effect. Further, these contexts are generated from tuples that are themselves bootstrapped, which minimizes the amount of supervision necessary.

The Levy and Goldberg model trains using single-word pairs, while our CMs could be composed of multiple words. For this reason, we decompose each cause–effect tuple, (CM_c, CM_e), such that each word $w_c \in CM_c$ is paired with each word $w_e \in CM_e$.

After filtering the extracted cause-effect tuples for stop words and retaining only nouns, verbs, and adjectives, we generated over 3.6M (w_c, w_e) word-pairs[4] from the approximately 800K causal tuples.

The model learns two embedding vectors for each word, one for when the word serves as a target word and another for when the word serves as a context word. Here, since the relation of interest is inherently directional, both sets of embeddings are meaningful, and so we make use of both – the target vectors encode the effects of given causes, whereas the context vectors capture the causes of the corresponding effects.

Causal Alignment Model (cAlign): Monolingual alignment (or translation) models have been shown to be successful in QA (Berger et al., 2000; Echihabi and Marcu, 2003; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013), and recent work has shown that they can be successfully trained with less data than embedding models (Sharp et al., 2015).

[4]For all models proposed in this section we used lemmas rather than words.

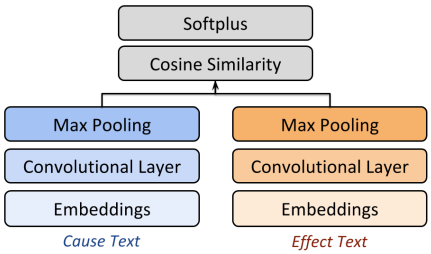


Figure 1: Architecture of the causal convolutional network.

To verify these observations in our context, we train an alignment model that "translates" causes (i.e., the "source language") into effects (i.e., the "destination language"), using our cause–effect tuples. This is done using IBM Model 1 (Brown et al., 1993) and GIZA++ (Och and Ney, 2003).

Causal Convolutional Neural Network Model (cCNN): Each of the previous models have at their root a bag-of-words representation, which is a simplification of the causality task. To address this potential limitation, we additionally trained a convolutional neural network (CNN) which operates over variable-length texts, and maintains distinct embeddings for causes and effects. The architecture of this approach is shown in Figure 1, and consists of two sub-networks (one for cause text and one for effect text), each of which begins by converting the corresponding text into 50-dimensional embeddings. These are then fed to a convolutional layer,[5] which is followed by a max-pooling layer of equal length. Then, these top sub-network layers, which can be thought of as a type of phrasal embedding, are merged by taking their cosine similarity. Finally, this cosine similarity is normalized by feeding it into a dense layer with a single node which has a softplus activation. In designing our CNN, we attempted to minimize architectural and hyperparameter tuning by taking inspiration from Iyyer et al. (2015), preferring simpler architectures. We train the network using a binary cross entropy objective function and the Adam optimizer (Kingma and Ba, 2014), using the Keras library (Chollet, 2015) operating over Theano (Theano Development Team, 2016), a popular deep-learning framework.[6]

[5]The convolutional layer contained 100 filters, had a filter length of 2 (i.e., capturing bigram information), and an inner ReLU activation.

[6]We also experimented with an equivalent architecture where the sub-networks are implemented using long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and found that they consistently under-perform this CNN architecture. Our conjecture is that CNNs perform better because LSTMs are more sensitive to overall word order than CNNs, which capture only local contexts, and we have relatively little training data, which prevents the LSTMs from generalizing well.

Noise-aware Causal Embedding Model (cEmbedNoise): We designed a variant of our cEmbed approach to address the potential impact of the noise introduced by our bootstrapping method. While training, we weigh the causal tuples by the likelihood that they are truly causal, which we approximate with pointwise mutual information (PMI). For this, we first score the tuples by their causal PMI and then scale these scores by the overall frequency of the tuple (Riloff, 1996), to account for the PMI bias toward low-frequency items. That is, the score S of a tuple, t, is computed as:

$$S(t) = \log \frac{p(t|causal)}{p(t)} * \log(freq(t)) \qquad (1)$$

We then discretize these scores into five quantiles, ascribing a linearly decreasing weight during training to datums in lower scoring quantiles.

6 Direct Evaluation: Ranking Word Pairs

We begin the assessment of our models with a *direct* evaluation to determine whether or not the proposed approaches capture causality better than general-purpose word embeddings and whether their robustness improves upon a simple database look-up. For this evaluation, we follow the protocol of Levy and Goldberg (2014). In particular, we create a collection of word pairs, half of which are causally related, with the other half consisting of other relations. These pairs are then ranked by our models and several baselines, with the goal of ranking the causal pairs above the others. The embedding models rank the pairs using the cosine similarity between the target vector for the causal word and the context vector of the effect word. The alignment model ranks pairs using the probability $P(\text{Effect}|\text{Cause})$ given by IBM Model 1, and the CNN ranks pairs by the value of the output returned by the network.

6.1 Data

In order to avoid bias towards our extraction methods, we evaluate our models on an external set of

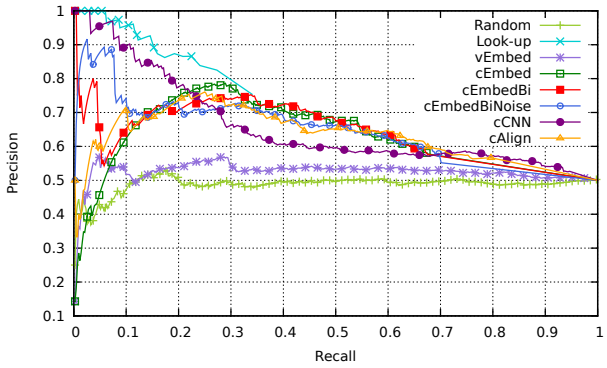


Figure 2: Precision-recall curve showing the ability of each model to rank causal pairs above non-causal pairs. For clarity, we do not plot cEmbedNoise, which performs worse than cEmbedBiNoise. The Look-up model has no data points beyond the 35% recall point.

word pairs drawn from the SemEval 2010 Task 8 (Hendrickx et al., 2009), originally a multi-way classification of semantic relations between nominals. We used a total of 1730 nominal pairs, 865 of which were from the Cause-Effect relation (e.g., (*dancing* → *happiness*)) and an equal number which were randomly selected from the other eight relations (e.g., (*juice* → *grapefruit*), from the Entity-Origin relation). This set was then randomly divided into equally-sized development and test partitions.

6.2 Baselines

We compared our distributional similarity models against three baselines:

Vanilla Embeddings Model (vEmbed): a standard `word2vec` model trained with the skip-gram algorithm and a sliding window of 5, using the original texts from which our causal pairs were extracted.[7] As with the cEmbed model, SemEval pairs were ranked using the cosine similarity between the vector representations of their arguments.

Look-up Baseline: a given SemEval pair was ranked by the number of times it appeared in our extracted cause-effect tuples.

Random: pairs were randomly shuffled.

6.3 Results

Figure 2 shows the precision-recall (PR) curve for each of the models and baselines. As expected, the causal models are better able to rank causal pairs than the vanilla embedding baseline (vEmbed), which, in turn, outperforms the random baseline. Our look-up baseline, which ranks pairs by their frequency in our causal database, shows a high precision for this task, but has coverage for only 35% of the causal SemEval pairs.

Some models perform better on the low-recall portion of the curve (e.g., the look-up baseline and cCNN), while the embedding and alignment models have a higher and more consistent performance across the PR curve. We hypothesize that models that better *balance* precision and recall will perform better in a real-world QA task, which may need to access a given causal relation through a variety of lexical patterns or variations. We empirically validate this observation in Section 7.

The PR curve for the causal embeddings shows an atypical dip at low-recall. To examine this, we analyzed its top-ranked 15% of SemEval pairs, and found that incorrectly ranked pairs were not found in the database of causal tuples. Instead, these incorrect rankings were largely driven by low frequency words whose embeddings could not be robustly estimated due to lack of direct evidence. Because this sparsity is partially driven by directionality, we implemented a bidirectional embedding model (cEmbedBi) that (a) trains a second embedding model by reversing the input (effects as targets, causes as contexts), and (b) ranks pairs by the *average* of the scores returned by these two unidirectional causal embedding models. Specifically, the final bidirectional score of the pair, (e_1, e_2), where e_1 is the can-

[7] All embedding models analyzed here, including this baseline and our causal variants, produced embedding vectors of 200 dimensions.

didate cause and e_2 is the candidate effect, is:

$$s_{bi}(e_1, e_2) = \frac{1}{2}(s_{c\to e}(e_1, e_2) + s_{e\to c}(e_2, e_1)) \quad (2)$$

where $s_{c\to e}$ is the score given by the original causal embeddings, i.e., from cause to effect, and $s_{e\to c}$ is the score given by the reversed-input causal embeddings, i.e., from effect to cause.

As Figure 2 shows, the bidirectional embedding variants consistently outperform their unidirectional counterparts. All in all, the best overall model is cEmbedBiNoise, which is both bidirectional and incorporates the noise handling approach from Section 5. This model substantially improves performance in the low-recall portion of the curve, while also showing strong performance across the curve.

7 Indirect Evaluation: QA Task

The main objective of our work is to investigate the impact of a customized causal embedding model for QA. Following our direct evaluation, which solely evaluated the degree to which our models directly encode causality, here we evaluate each of our proposed causal models in terms of their contribution to a downstream real-world QA task.

Our QA system uses a standard reranking approach (Jansen et al., 2014). In this architecture, the candidate answers are initially extracted and ranked using a shallow candidate retrieval (CR) component that uses solely information retrieval techniques, then they are re-ranked using a "learning to rank" approach. In particular, we used SVM rank[8], a Support Vector Machines classifier adapted for ranking, and re-ranked the candidate answers with a set of features derived from both the initial CR score and the models we have introduced. For our model combinations (see Table 2), the feature set includes the CR score and the features from each of the models in the combination.

7.1 Data

We evaluate on a set of causal questions extracted from the Yahoo! Answers corpus[9] with simple surface patterns such as *What causes ...* and *What is the result of ...*[10]. We extracted a total of 3031 questions, each with at least four candidate answers, and we evaluated performance using five-fold cross-validation, with three folds for training, one for development, and one for testing.

7.2 Models and Features

We evaluate the contribution of the bidirectional and noise-aware causal embedding models (cEmbedBi, and cEmbedBiNoise) as well as the causal alignment model (cAlign) and the causal CNN (cCNN). These models are compared against three baselines: the vanilla embeddings (vEmbed), the lookup baseline (LU), and additionally a vanilla alignment model (vAlign) which is trained over 65k question-answer pairs from Yahoo! Answers.

The features[11] we use for the various models are:

Embedding model features: For both our vanilla and causal embedding models, we use the same set of features as Fried et al. (2015): the maximum, minimum, and average pairwise cosine similarity between question and answer words, as well as the overall similarity between the composite question and answer vectors. When using the causal embeddings, since the relation is directed, we first determine whether the question text is the cause or the effect[12], which in turn determines which embeddings to use for the question text and which to use for the candidate answer texts. For example, in a question such as "*What causes X?*", since *X* is the effect, all cosine similarities would be found using the effect vectors for the question words and the cause vectors for the answer candidate words.

Alignment model features: We use the same global alignment probability, $p(Q|A)$ of Surdeanu et al. (2011). In our causal alignment model, we adapt this to causality as $p(\text{Effect}|\text{Cause})$, and again we first determine the direction of the causal relation implied in the question. We include the additional undirected alignment features based on

[8] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[9] Freely available through Yahoo!'s Webscope program (research-data-requests@yahoo-inc.com)

[10] We lightly filtered these with stop words to remove non-causal questions, such as those based on math problems and the results of sporting events. Our dataset will be freely available, conditioned on users having obtained the Webscope license.

[11] Due to the variety of features used, each feature described here is independently normalized to lie between 0.0 and 1.0.

[12] We do this through the use of simple regular expressions, e.g., "^ [Ww]hat ([a-z]+){0,3}cause.+"

#	Model	P@1
	Baselines	
1	Random	16.43
2	CR	24.31
3	CR + vEmbed	34.61
4	CR + vAlign	19.24
5	CR + Look-up (LU)	29.56
	Single Causal Models	
6	CR + cEmbedBi	31.32
7	CR + cEmbedBiNoise	30.15
8	CR + cAlign	23.49
9	CR + cCNN	24.66
	Model Combinations	
10	CR + vEmbed + cEmbedBi	37.08*
11	CR + vEmbed + cEmbedBiNoise	35.50*
12	CR + vEmbed + cEmbedBi + LU	36.75*
13	CR + vEmbed + cAlign	34.31
14	CR + vEmbed + cCNN	33.45
	Model Stacking	
15	CR + vEmbed + cEmbedBi + cEmbedBiNoise	**37.28***

Table 2: Performance in the QA evaluation, measured by precision-at-one (P@1). The "Bi" suffix indicates a bidirectional model; the "Noise" suffix indicates a model that is noise aware. * indicates that the difference between the corresponding model and the CR + vEmbed baseline is statistically significant ($p < 0.05$), determined through a one-tailed bootstrap resampling test with 10,000 iterations.

Jensen-Shannon distance, proposed more recently by Fried et al. (2015), in our vanilla alignment model. However, due to the directionality inherent in causality, they do not apply to our causal model so there we omit them.

Look-up feature: For the look-up baseline we count the number of times words from the question and answer appear together in our database of extracted causal pairs, once again after determining the directionality of the questions. If the total number of matches is over a threshold[13], we consider the causal relation to be established and give the candidate answer a score of 1; or a score of 0, otherwise.

7.3 Results

The overall results are summarized in Table 2. Lines 1–5 in the table show that each of our baselines performed better than CR by itself, except for vAlign, suggesting that the vanilla alignment model does not generate accurate predictions for causal questions. The strongest baseline was CR + vEmbed (line 3), the vanilla embeddings trained over Gigaword, at 34.6% P@1. For this reason, we consider this to be the baseline to "beat", and perform statistical significance of all proposed models with respect to it.

Individually, the cEmbedBi model is the best performing of the causal models. While the performance of cAlign in the direct evaluation was comparable to that of cEmbedBi, here it performs far worse (line 6 vs 8), suggesting that the robustness of embeddings is helpful in QA. Notably, despite the strong performance of the cCNN in the low-recall portion of the PR curve in the direct evaluation, here the model performs poorly (line 9).

No individual causal model outperforms the strong vanilla embedding baseline (line 3), likely owing to the reduction in generality inherent to building task-specific QA models. However, comparing lines 6–9 vs. 10–14 shows that the vanilla and causal models are capturing different and complementary kinds of knowledge (i.e., causality vs. association through distributional similarity), and are able to be combined to increase overall task performance (lines 10–12). These results highlight that QA is a complex task, where solving methods need to address the many distinct information needs in question sets, including both causal and direct association relations. This contrasts with the direct evaluation, which focuses strictly on causality, and where the vanilla embedding baseline performs near chance. This observation highlights one weakness of word similarity tasks: their narrow focus may not directly translate to estimating their utility in real-world NLP applications.

Adding in the lookup baseline (LU) to the best-performing causal model does not improve performance (compare lines 10 and 12), suggesting that the bidirectional causal embeddings subsume the contribution of the LU model. cEmbedBi (line 10) also performs better than cEmbedBiNoise (line 11). We conjecture that the "noise" filtered out by cEmbedBiNoise contains distributional similarity information, which is useful for the QA task. cEmbedBi vastly outperforms cCNN (line 14), suggesting that strong overall performance across the precision-recall curve better translates to the QA task. We hypothesize that the low cCNN performance is caused by insufficient training data, preventing the CNN ar-

[13]Empirically determined to be 100 matches. Note that using this threshold performed better than simply using the total number of matches.

Error/observation	% Q
Both chosen and gold are equally good answers	45%
Causal max similarity of chosen is higher	35%
Vanilla overall similarity of chosen is higher	35%
Chosen answer is better than the gold answer	25%
The question is very short / lacks content words	15%
Other	10%

Table 3: Results of an error analysis performed on a random sample of 20 incorrectly answered questions showing the source of the error and the percentage of questions that were affected. Note that questions can belong to multiple categories.

chitecture from generalizing well.

Our best performing overall model combines both variants of the causal embedding model (cEmbedBi and cEmbedBiNoise), reaching a P@1 of 37.3%, which shows a 7.7% relative improvement over the strong CR + vEmbed baseline.

7.4 Error Analysis

We performed an error analysis to gain more insight into our model as well as the source of the remaining errors. For simplicity, we used the combination model CR + vEmbed + cEmbedBi. Examining the model's learned feature weights, we found that the vanilla overall similarity feature had the highest weight, followed by the causal overall similarity and causal maximum similarity features. This indicates that even in causal question answering, the overall *topical* similarity between question and answer is still useful and complementary to the causal similarity features.

To determine sources of error, we randomly selected 20 questions that were incorrectly answered and analyzed them according to the categories shown in Table 3. We found that for 70% of the questions, the answer chosen by our system was as good as or better than the gold answer, often the case with community question answering datasets.

Additionally, while the maximum causal similarity feature is useful, it can be misleading due to embedding noise, low-frequency words, and even the bag-of-words nature of the model (35% of the incorrect questions). For example, in the question *What are the effects of growing up with an older sibling who is better than you at everything?*, the model chose the answer *...You are you and they are them - you will be better and different at other things...* largely because of the high causal similarity between (*grow* $\rightarrow$ *better*). While this could arguably be helpful in another context, here it is irrelevant, suggesting that in the future improvement could come from models that better incorporate textual dependencies.

8 Conclusion

We presented a framework for creating customized embeddings tailored to the information need of causal questions. We trained three popular models (embedding, alignment, and CNN) using causal tuples extracted with minimal supervision by bootstrapping cause-effect pairs from free text, and evaluated their performance both directly (i.e., the degree to which they capture causality), and indirectly (i.e., their real-world utility on a high-level question answering task).

We showed that models that incorporate a knowledge of causality perform best for both tasks. Our analysis suggests that the models that perform best in the real-world QA task are those that have consistent performance across the precision-recall curve in the direct evaluation. In QA, where the vocabulary is much larger, precision must be balanced with high-recall, and this is best achieved by our causal embedding model. Additionally, we showed that vanilla and causal embedding models address different information needs of questions, and can be combined to improve performance.

Extending this work beyond causality, we hypothesize that additional embedding spaces customized to the different information needs of questions would allow for robust performance over a larger variety of questions, and that these customized embedding models should be evaluated both directly and indirectly to accurately characterize their performance.

Resources

All code and resources needed to reproduce this work are available at `http://clulab.cs.arizona.edu/data/emnlp2016-causal/`.

Acknowledgments

We thank the Allen Institute for Artificial Intelligence for funding this work. Additionally, this work was partially funded by the Defense Advanced Research Projects Agency (DARPA) Big Mechanism program under ARO contract W911NF-14-1-0395.

References

[Berger et al.2000] A. Berger, R. Caruana, D. Cohn, D. Freytag, and V. Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In *Proc. of the 23rd Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*.

[Bordes et al.2014] A. Bordes, S. Chopra, and J. Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

[Brown et al.1993] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

[Chen and Manning2014] D. Chen and C. D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of the Conferenc on Empirical Methods for Natural Language Processing (EMNLP)*.

[Chollet2015] F. Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

[Chu-Carroll et al.2004] J. Chu-Carroll, K. Czuba, J. M. Prager, A. Ittycheriah, and S. Blair-Goldensohn. 2004. IBM's PIQUANT II in TREC 2004. In *Text Retrieval Conference (TREC)*.

[Clark et al.2013] P. Clark, P. Harrison, and N. Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proc. of the 2013 workshop on Automated Knowledge Base Construction (AKBC)*, pages 37–42.

[Cole et al.2005] S. V. Cole, M. D. Royal, M. G. Valtorta, M. N. Huhns, and J. B. Bowles. 2005. A lightweight tool for automatically extracting causal relationships from text. In *SoutheastCon, 2006. Proc. of the IEEE*, pages 125–129.

[Do et al.2011] Q. X. Do, Y. S. Chan, and D. Roth. 2011. Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

[Echihabi and Marcu2003] A. Echihabi and D. Marcu. 2003. A noisy-channel approach to question answering. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16–23.

[Etzioni2011] O. Etzioni. 2011. Search needs a shake-up. *Nature*, 476(7358):25–26.

[Faruqui et al.2016] M. Faruqui, Y. Tsvetkov, R. Rastogi, and C. Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

[Ferrucci et al.2010] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79.

[FitzGerald et al.2015] N. FitzGerald, O. Täckström, K. Ganchev, and D. Das. 2015. Semantic role labeling with neural network factors. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970.

[Fried et al.2015] D. Fried, P. Jansen, G. Hahn-Powell, M. Surdeanu, and P. Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.

[Girju and Moldovan2002] R. Girju and D. I. Moldovan. 2002. Text mining for causal relations. In *FLAIRS Conference*, pages 360–364.

[Hearst1992] M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th conference on Computational linguistics (COLING)*, pages 539–545.

[Hendrickx et al.2009] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proc. of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

[Hochreiter and Schmidhuber1997] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Iyyer et al.2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

[Jansen et al.2014] P. Jansen, M. Surdeanu, and P. Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

[Khoo et al.1998] C. S.G. Khoo, J. Kornfilt, R. N. Oddy, and S. H. Myaeng. 1998. Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, 13(4):177–186.

[Kiela et al.2015] D. Kiela, F. Hill, and S. Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Kingma and Ba2014] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Levy and Goldberg2014] O. Levy and Y. Goldberg. 2014. Dependency-based word embeddings. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 302–308.

[Levy et al.2015] O. Levy, S. Remus, C. Biemann, I. Dagan, and I. Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[Manning et al.2014] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

[Mikolov et al.2013] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Napoles et al.2012] C. Napoles, M. Gormley, and B. Van Durme. 2012. Annotated gigaword. In *Proc. of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100.

[Och and Ney2003] F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

[Oh et al.2013] J.-H. Oh, K. Torisawa, C. Hashimoto, M. Sano, S. De Saeger, and K. Ohtake. 2013. Why-question answering using intra-and inter-sentential causal relations. In *The 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1733–1743.

[Riedel et al.2013] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proc. of Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[Riezler et al.2007] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 464–471.

[Riloff1996] E. Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1044–1049.

[Sharp et al.2015] R. Sharp, P. Jansen, M. Surdeanu, and P. Clark. 2015. Spinning straw into gold. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*.

[Soricut and Brill2006] R. Soricut and E. Brill. 2006. Automatic question answering using the web: Beyond the factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2):191–206.

[Surdeanu et al.2011] M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

[Valenzuela-Escárcega et al.2016] M. A. Valenzuela-Escárcega, G. Hahn-Powell, and M. Surdeanu. 2016. Odin's runes: A rule language for information extraction. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC)*.

[Woodsend and Lapata2015] K. Woodsend and M. Lapata. 2015. Distributed representations for unsupervised semantic role labeling. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Yang and Mao2014] X. Yang and K. Mao. 2014. Multi level causal relation identification using extended features. *Expert Systems with Applications*, 41(16):7171–7181.

[Yang et al.2014] M.-C. Yang, N. Duan, M. Zhou, and H.-C. Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650.

[Yao et al.2013] X. Yao, B. Van Durme, C. Callison-Burch, and P. Clark. 2013. Semi-markov phrase-based monolingual alignment. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Yih et al.2013] W. Yih, M. Chang, C. Meek, and A. Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Improving Semantic Parsing via Answer Type Inference

Semih Yavuz[1], **Izzeddin Gur**[1], **Yu Su**[1], **Mudhakar Srivatsa**[2] **and Xifeng Yan**[1]
[1]University of California, Santa Barbara, Department of Computer Science
[2]IBM Research
{syavuz,izzeddingur,ysu,xyan}@cs.ucsb.edu
msrivats@us.ibm.com

Abstract

In this work, we show the possibility of inferring the answer type before solving a factoid question and leveraging the type information to improve semantic parsing. By replacing the topic entity in a question with its type, we are able to generate an abstract form of the question, whose answer corresponds to the answer type of the original question. A bidirectional LSTM model is built to train over the abstract form of questions and infer their answer types. It is also observed that if we convert a question into a statement form, our LSTM model achieves better accuracy. Using the predicted type information to rerank the logical forms returned by AgendaIL, one of the leading semantic parsers, we are able to improve the F1-score from 49.7% to 52.6% on the WEBQUESTIONS data.

1 Introduction

Large scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), and YAGO (Suchanek et al., 2007) that store the world's factual information in a structured fashion have become substantial resources for people to solve questions. KB-based factoid question answering (KB-QA) that attempts to find exact answers to natural language questions has gained much attention recently. KB-QA is a challenging task due to the representation variety between natural language and structural knowledge in KBs.

As one of the promising KB-QA techniques, semantic parsing maps a natural language question into its semantic representation (e.g., logical forms).

Ranking	$F1$	# Improved Qs
AgendaIL	49.7	-
w/ Oracle Types@10	57.3	+234
w/ Oracle Types@20	58.7	+282
w/ Oracle Types@50	60.1	+331
w/ Oracle Types@All	60.5	+345

Table 1: What if the correct answer type is enforced? On WebQuestions, we remove those with incorrect answer types in the top-k logical forms returned by AgendaIL (Berant and Liang, 2015), a leading semantic parsing system, and report the new average F1 score as well as the number of questions with an improved F1 score.

It uses a logical language with predicates closely related to KB schema, and constructs a dictionary that maps relations to KB predicates. The problem then reduces to generating candidate logical forms, ranking them, and selecting one to derive the final answer.

In this work, we propose an answer type prediction model that can improve the ranking of the candidate logical forms generated by semantic parsing. The type of an entity, e.g., `person`, `organization`, `location`, carries very useful information for various down-stream natural language processing tasks such as co-reference resolution (Recasens et al., 2013), knowledge base population (Carlson et al., 2010), relation extraction (Yao et al., 2012), and question answering (Lin et al., 2012). Although the potential clues for answer type from the question has been employed in the recent work AgendaIL (Berant and Liang, 2015) at the lexical level, Table 1 suggests that there is yet a large room for further improvement by explicitly enforc-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 149–159,
Austin, Texas, November 1-5, 2016.

ing answer type. Inspired by this observation, we aim to directly predict the KB type of the answer from the question. In contrast to a small set of predefined types as used in previous answer type prediction methods (e.g., (Li and Roth, 2002)), KBs could have thousands of fine-grained types. Take "When did Shaq come into the NBA?" as a running example. We aim to predict the KB type of its answer as `SportsLeagueDraft`.[1]

The value of typing answers in a fine granularity can be appreciated from two perspectives: (1) Since each entity in a KB like Freebase has a few types, answer type could help prune answer candidates, (2) since each predicate in the KB has a unique type schema, answer type can help rank logical forms.

The key challenge of using answer types to rerank logic forms and hence their corresponding answers, is that it shall be done before the answer is found. Otherwise, there is no need to further infer its type. Inspired by the observation that the answer type of a question is invariant as long as the type of the topic entity (`Shaq`) remains the same (`DraftedAthlete`), we define **abstract question** as the question where the topic entity mention is replaced by its corresponding KB type. For the aforementioned example, the best candidate abstract question is "When did `DraftedAthlete` come into the NBA?" and the answer to this question is `SportsLeagueDraft`. Hence, we can reduce the answer type prediction task to abstract question answering.

The first step in our method is question abstraction, in which we generate candidate abstract questions based on the context of question and its candidate topic entities. We build a bidirectional LSTM network over the question that recursively computes vector representations for the past and future contexts of an entity mention. Based on these context representations, we predict the right type of the entity mention. Next, in order to better utilize the syntactic features of the question, we convert the question form into a normal statement form by using dependency tree of the question. For the running example, after performing the conversion, the abstract question becomes "`DraftedAthlete` come when into the NBA?" We then construct a bidirectional LSTM neural network over this final representation of the question and predict the type of the answer. Using the inferred answer type, we are able to improve the result of AgendaIL (Berant and Liang, 2015) on WebQuestions (Berant et al., 2013) from 49.7% to 52.6%.

[1] KB type of answer ("1992 NBA Draft") in the context.

2 Background

The knowledge base we work with consists of triples in subject-predicate-object form. It can be represented as $\mathcal{K} = \{(e_1, p, e_2) : e_1, e_2 \in \mathcal{E},\ p \in \mathcal{P}\}$, where $\mathcal{E}$ denotes the set of entities (e.g., `ShaquilleOneal`), and $\mathcal{P}$ denotes the set of binary predicates (e.g., `Drafted`). A knowledge base in this format can be visualized as a graph where entities are nodes, and predicates are directed edges between entities. Freebase is used in this work as the knowledge base. It has more than 41M entities, 596M facts, and 24K types.

Types are an integral part of the Freebase schema. Each entity e in Freebase has a set of categories (types) it belongs to, and this information can be obtained by checking the out-going predicates (`Type.Object.Type`) from e. For example, `ShaquilleOneal` has 20 Freebase types including `Person`, `BasketballPlayer`, `DraftedAthlete`, `Celebrity`, and `FilmActor`. For a specific question involving `ShaquilleOneal`, among these types, only a few will be relevant.

Each predicate in Freebase is from a subject entity to an object entity, and has a type signature. It has a unique expected types for its subject and object, independent of the individual subject and object entities themselves. For example, the predicate `People.Person.Profession` expects its subject to be of `Person` type and its object to be of `Profession` type.

3 Question Abstraction

The type of the topic entity rather than the entity itself is essential for inferring the answer type, which is invariant as the topic entity changes within the same class. For example, independent of which NBA player (with `DraftedAthlete` type) is the topic entity of this question "When did Shaq come into the NBA", the type of the answer is always go-

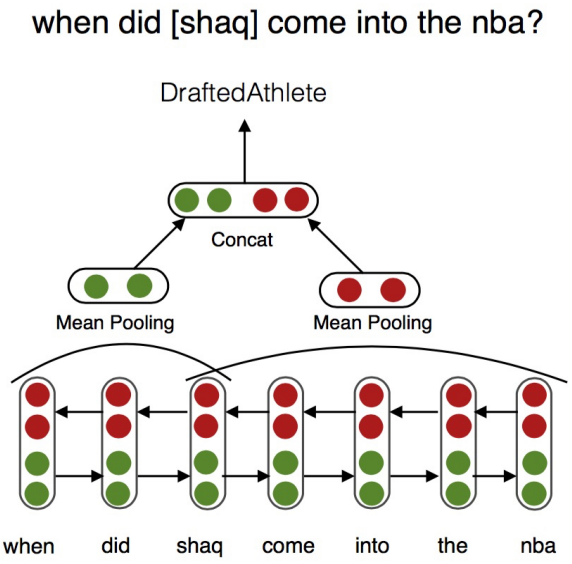


Figure 1: Bi-directional LSTM model for question abstraction. Green circles represent the forward sequence's hidden vectors, while the red circles denote the backward sequence's. `shaq` (the topic entity mention) is the single output node of the network.

ing to be `SportsLeagueDraft` in Freebase. Predicting this distinct type among the large number of candidate types in Freebase is a challenging task. We propose a two-step solution for this problem. In the first step, we compute a confidence score for each possible KB type for a given topic entity using a bidirectional LSTM network. The second step prunes candidate types using the entity type information in Freebase.

3.1 Formulation

Given a natural language question and its topic entity mention, question abstraction is to predict types of the mention in the question context. Formally, let $q = (x_1, x_2, \ldots, x_L)$ denote the question, m be the topic entity mention in q, and $T = \{t_1, t_2, \ldots, t_K\}$ the set of all types in KB. Given q and m, we compute a probability distribution $o \in \mathbb{R}^{K\times 1}$ over T, where o_k denotes the likelihood of t_k being the correct type of m in q.

3.2 Scoring Topic Entity Types with LSTM

Model. We formulate question abstraction as a classification problem. A bidirectional LSTM network is built over q whose output is computed from the nodes that correspond to the words of m. Fig. 1 illustrates the model for the question "When did Shaq come into the NBA?"

Let $u(x) \in \mathbb{R}^{D\times 1}$ denote the vector space embedding of word x. Forward and backward outputs $\overrightarrow{h}_l, \overleftarrow{h}_l \in \mathbb{R}^{D_h\times 1}$ of bidirectional LSTM are recursively computed by

$$\overrightarrow{h}_l, \overrightarrow{c}_l = LSTM(u(x_l), \overrightarrow{h}_{l-1}, \overrightarrow{c}_{l-1}) \quad (1)$$

$$\overleftarrow{h}_l, \overleftarrow{c}_l = LSTM(u(x_l), \overleftarrow{h}_{l+1}, \overleftarrow{c}_{l+1}) \quad (2)$$

as described in Graves (2012), where $\overrightarrow{c_l}, \overleftarrow{c_l} \in \mathbb{R}^{D_h\times 1}$ stand for LSTM cell states.

To encode the context of m to the final output, we apply an AVERAGE pooling layer when computing the output. For each output node $r \in [i, j]$ (i and j correspond to the starting and ending indices of m in q), we compute final forward and backward outputs by

$$\overrightarrow{v_r} = AVG(\overrightarrow{h_1}, \ldots, \overrightarrow{h_r}) \quad (3)$$

$$\overleftarrow{v_r} = AVG(\overleftarrow{h_r}, \ldots, \overleftarrow{h_n}), \quad (4)$$

where AVG stands for average pooling.

We take the average of outputs at each output node

$$\overrightarrow{v} = AVG(\overrightarrow{v_i}, \ldots, \overrightarrow{v_j}) \quad (5)$$

$$\overleftarrow{v} = AVG(\overleftarrow{v_i}, \ldots, \overleftarrow{v_j}) \quad (6)$$

as the forward and backward outputs of the whole network. The final representation v of the network is obtained by concatenating $\overrightarrow{v}$ and $\overleftarrow{v}$.

For question q, the probability distribution o over types is computed by

$$s(q) = W_{hy} v \quad (7)$$

$$o(q) = softmax(s(q)), \quad (8)$$

where $W_{hy} \in \mathbb{R}^{K\times(2D_h)}$ since v is the concatenation of two vectors of dimension D_h, where D_h is the hidden vector dimension.

Objective Function and Learning. Given an input question q with a topic entity mention m, LSTM network computes the probability distribution $o(q) \in \mathbb{R}^{K\times 1}$ as in (8). Let $y(q) \in \mathbb{R}^{K\times 1}$ denote the true target distribution over T for q, where

$y_k(q) = 1/n$ if t_k is a correct type, $y_k(q) = 0$ otherwise, and n is the number of correct types. We use the cross-entropy loss function between $y(q)$ and $o(q)$, and define the objective function over all training data as

$$J(\theta) = -\sum_{q}\sum_{k=1}^{K} y_k(q)\log o_k(q) + \frac{\lambda}{2}\|\theta\|^2,$$

where λ denotes the regularization parameter, and θ represents the set of all model parameters to be learned. We use stochastic gradient descent with RMSProp (Tieleman and Hinton, 2012) for minimizing the objective function.

3.3 Pruning

Let T_e represent the set of KB types for entity e. We define the set of candidate types for entity mention m as

$$C_m = \bigcup_e T_e,$$

where e is a possible match of m in KB. We only need to score the types in C_m. Once the hidden representation v is computed by LSTM, we use submatrix $W_{hy}[C_m]$ that consists of rows of W_{hy} corresponding to the types in C_m as the scoring matrix in (7). This returns the final scores for candidate types in C_m.

4 Conversion to Statement Form

The objective of the conversion is to canonicalize question form into declarative statement (subject-relation-object) form. We use a simple pattern-based method that relies on dependency tree[2] (Manning et al., 2014). It decides whether the sub-trees of the root need reordering based on their dependency relations[3].

Before obtaining the dependency tree, we retrieve named entity (NER) tags of the question tokens. We replace a group of question tokens corresponding a named entity with a special token, ENTITY, to simplify the parse tree. In Figure 2, the question is first transformed to "what boarding school did ENTITY go to?" Each question is represented by the root's

[2]We use Stanford CoreNLP dependency parser

[3]http://universaldependencies.org

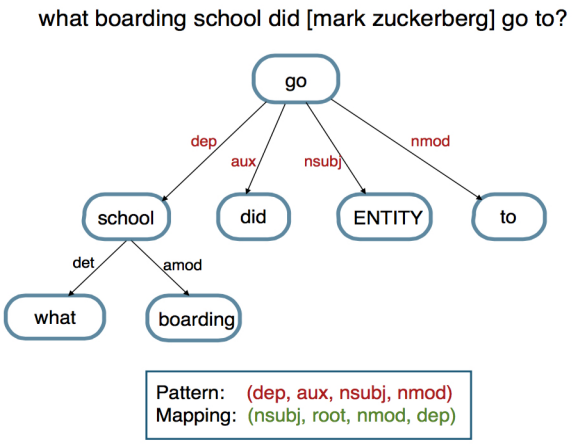


Figure 2: Conversion: red relations form the input pattern

Pattern	Conversion
(cop, nsubj) who was anakin skywalker?	(nsubj, root, cop) anakin skywalker was who
(dobj, aux, nsubj) what language does australians speak?	(nsubj, root, dobj) australians speak what language
(dobj, aux, nsubj, nmod) what did edward jenner do for a living?	(nsubj, root, dobj, nmod) edward jenner do what for a living
(nsubj, dobj) who played bilbo baggins?	(nsubj, root, dobj) who played bilbo baggins
(advmod, aux, nsubj) where did benjamin franklin died?	(nsubj, root, advmod) benjamin franklin died where

Table 2: Top-5 most common patterns with mappings.

dependency relations to its sub-trees in the original order, e.g., (dep, aux, nsubj, nmod). We cluster all these sequences and detect the patterns that appear at least 5 times in the training data. These patterns are then manually mapped to their corresponding conversion (pattern vs. mapping in Figure 2).

Once the recomposition order of the sub-trees is determined by the conversion mapping, we finalize the reordering of the question tokens by keeping the order of words within the sub-trees same as the original order in the question. The example in Figure 2 becomes "ENTITY go to what boarding school" with its corresponding sub-tree conversion mapping (nsubj, root, nmod, dep). If no mapping is created for a pattern, we keep the order of the words exactly as they occur in the original question form.

The motivation behind conversion is to overcome the potential semantic confusion stemming from varities in syntactic structures. To exemplify, consider two hypothetical questions "who plays X in

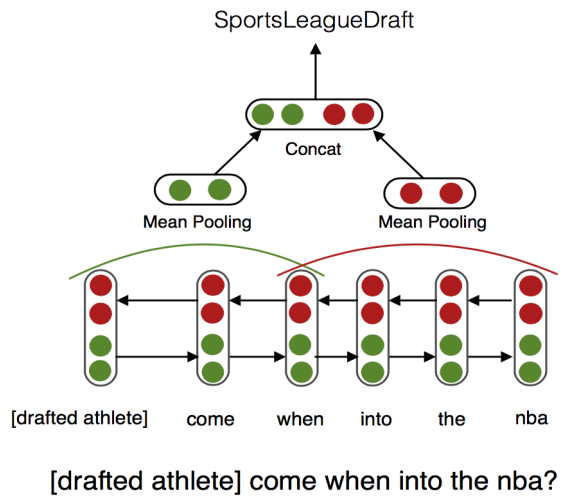


Figure 3: Bi-directional LSTM model over the final representation of the question. Green and red circles are corresponding to forward and backward hidden vectors, respectively. The output node is `when`.

Y?" and "who does Z play in Y?", where X is a `FilmCharacter`, Y is a `Film`, and Z is a `FilmActor`, with answer types `FilmActor` and `FilmCharacter`, respectively. With conversion, we aim to transform second question into "Z play who in Y", while leaving the first one as it is. Noting that the order of words affects the output of our answer type inference network, our intuition is to let the model distinguish better between such questions using their syntactic structure in this way.

5 Answer Type Prediction

Given a reordered question with topic entity mention m, and a topic entity type $t_e \in T$, our task is to predict a probability distribution $o \in \mathbb{R}^{K\times 1}$ over the answer types.

A topic entity type $t_e \in T$ is described as a set of words, $\{x_i\}$. Let $u(x_i) \in \mathbb{R}^{D\times 1}$ represent the vector space embedding of x_i, the representation of t_e is computed by the average encoding,

$$u(t_e) = \frac{1}{|\{x_i\}|}\sum_{x_i} u(x_i). \quad (9)$$

As the first step, we replace the words of entity mention m with topic entity type t_e, and obtain a new input word sequence r. t_e is treated as one word and encoded by Eq. 9. We construct a bidirectional LSTM network over this input sequence r, whose output node corresponds to the question word. The output of the network is a probability distribution over types denoting the likelihood of being the answer type. Figure 3 shows how the network is constructed for the running example. The same average pooling described in Section 3.2 is applied to obtain the final forward and backward output vectors $\overrightarrow{v}$ and $\overleftarrow{v}$ from the output node (this time, single output node) of network. The final output vector v for prediction is obtained by concatenating $\overrightarrow{v}$, and $\overleftarrow{v}$. The distribution o is computed by a standard softmax layer. The learning is performed by the same cross-entropy loss and objective function described in Section 3.2.

6 Reranking by Answer Type

In this section, we describe how to rerank logical forms based on our answer type prediction model.

Reranking Model. Let $l_1, l_2, \ldots, l_N$ be the logical forms generated for question q by a semantic parser, e.g., AgendaIL. Each logical form has a score from the semantic parser. Meanwhile, our answer type prediction model generates a score for the answer type of each logical form. Therefore, we can represent each logical form l_i using a pair of scores: the score from semantic parser and the score from our type prediction model. Suppose we know which logical forms are "correct", using the two scores as input, we train a logistic regression model with cross-entropy loss to learn a binary classifier for predicting the correct logical forms. We rerank the top-k logical forms using their probability computed by the trained logistic regression model, and select the one with the highest probability. Finally, we run the selected logical form against KB to retrieve the answer. We select the optimal value of k from $[1, N]$ using the training data. For AgendaIL on WebQuestions, we find that $k = 80$ gives the best result.

Training Data Selection. We now discuss which logical forms are "correct", i.e., how to select the positive examples to train the logistic regression model. Because a question can have more than one answer, we use the $F1$ score, the harmonic mean of *precision* and *recall*, to evaluate logical forms. We select all the logical forms with $F1 > 0$ as the set of positive examples. However, taking all the logical forms with $F1 = 0$ as negative examples will

not work well. Even though the $F1$ score of a logical form is 0, its answer type could still be correct. Therefore, we use the following trick: If there is a positive example with answer type t, we do not treat any other logical form with answer type t as negative example. The logical forms having $F1 = 0$, with the aforementioned exception, are then selected as the final set of negative examples. Our empirical study shows this trick works well.

7 Experiments

In this section, we describe the datasets, model training, and experimental results.

7.1 Dataset and Evaluation Metrics

Datasets. To evaluate our method, we use the WebQuestions dataset (Berant et al., 2013), which contains 5,810 questions crawled via Google Suggest API. The answers to these questions are annotated from Freebase using Amazon Mechanical Turk. The data is split into training and test sets of size 3,778 and 2,032 questions, respectively. This dataset has been popularly used in question answering and semantic parsing.

The SimpleQuestions (Bordes et al., 2015) contains 108,442 questions written in natural language by English-speaking human annotators. This dataset is a collection of question/Freebase-fact pairs rather than question/answer pairs. The data[4] is split and provided as training(75,910), test(21,687), and validation(10,845) sets. Each question is mapped to the subject, relation, and object of the corresponding Freebase fact. This dataset is only used for training the question abstraction model.

Training Data Preparation. Since WebQuestions only provides question-answer pairs along with annotated topic entities, we need to figure out the type information, which can be used as training data. We obtain *simulated* types as follows: We retrieve 1-hop and 2-hop predicates r from/to annotated topic entity e in Freebase. For each relation r, we query $(e, r, ?)$ and $(?, r, e)$ against Freebase and retrieve the candidate answers r_a. The $F1$ value of each candidate answer r_a is computed with respect to the annotated answer. The subject and object types of the relation r with the highest $F1$ value is selected as the *simulated* type for the topic entity and the answer. When there are multiple such relations, we obtain multiple *simulated* types for topic entity and answer, one from each relation. We treat each of them as correct with equal probability.

Candidate Logical Forms for Evaluation. To obtain candidate logical forms, we train AgendaIL (Berant and Liang, 2015) on WebQuestions with beam size 200 using the publicly available code[5] by the authors.

Evaluation Metric. We report average $F1$ score of the reranked logical forms using the predicted answer types as the main evaluation metric. It is a common performance measure in question answering as questions might have multiple answers.

7.2 Experimental Setup

We use 50 dimensional word embeddings, which are initialized by the 50 dimensional pre-trained word vectors[6] from GloVe (Pennington et al., 2014), and updated in the training process. Hyperparameters are tuned on the development set. The size of the LSTM hidden layer is set at 50. We use RMSProp (Tieleman and Hinton, 2012) with a learning rate of 0.005 and mini-batch size of 32 for the optimization. We use a dropout layer with probability 0.5 for regularization. We implemented the LSTM networks using *Theano* (Theano Development Team, 2016).

Identifying Topic Entity. We use Stanford NER tagger (Manning et al., 2014) to identify topic entity span for both training and test data. For entity linking, annotated mention span is mapped to a ranked list of candidate Freebase entities using Freebase Search API for the test data. For the training data, we use the gold Freebase topic entity linkings of each question provided by WebQuestions, coming from its question generation process.

Question Abstraction. We first pre-train the LSTM model described in Section 3.2 on the SimpleQuestions dataset. Then, we update the pre-trained model on the training portion of WebQuestions data where the *simulated* topic entity types are used as true labels. We use the detected topic entity mentions to obtain candidate matching entities in the KB using Freebase Search API. We use top-

[4] http://fb.ai/babi.

[5] https://github.com/percyliang/sempre

[6] http://nlp.stanford.edu/projects/glove/

Model	F1
(Berant et al., 2013)	35.7
(Yao and Van Durme, 2014)	33.0
(Berant and Liang, 2014)	39.9
(Bao et al., 2014)	37.5
(Bordes et al., 2014)	39.2
(Yang et al., 2014)	41.3
(Dong et al., 2015b)	40.8
(Yao, 2015)	44.3
(Berant and Liang, 2015)	49.7
(Yih et al., 2015)	52.5
(Reddy et al., 2016)	50.3
(Xu et al., 2016)	**53.3**
(Yih et al., 2015) (w/ Freebase API)	48.4
(Yih et al., 2015) (w/o ClueWeb)	50.9
(Xu et al., 2016) (w/o Wikipedia)	47.1
Our Approach (w/o SimpleQuestions)	51.6
Our Approach	**52.6**

Table 3: Comparison of our reranking-by-type system with several existing works on WebQuestions.

3 entities returned for the pruning step of *Question Abstraction* on the test examples.

Answer Type Prediction. We train *Answer Type Prediction* model using the *simulated* topic entity and answer types for each question. We perform the answer type prediction on test data using the predicted topic entity type.

7.3 Results

Our main result is presented in Table 3. Our system adds 2.9% absolute improvement over AgendaIL, and achieves 52.6% in $F1$ measure. Yih et al. (2015) achieve 52.5% by leveraging ClueWeb and S-MART (Yang and Chang, 2015), an advanced entity linking system. Xu et al. (2016) achieve 53.3% by leveraging Wikipedia and S-MART. If tested without Clueweb/Wikipedia/S-MART, their F1 scores are 48.4% and 47.1%, respectively. When our method is tested without using SimpleQuestions data for pretraining question abstraction module, it attains F1 score of 51.6%.

In Table 4, we present some question examples where our method can select a better logical form. Take the question "who did [australia] fight in the first world war?" as an example. Our topic entity type prediction module returns `MilitaryCombatant`, `StatisticalRegion`, and `Kingdom` as the top-3 results for the type of "australia" in this question, which indicates that it exploits the context of this short question successfully. The abstract question is "[military combatant] fight who in the first world war?" for which our system returns `MilitaryCombatant`, `MilitaryConflict`, and `MilitaryCommander` as answer types with probabilities 0.73, 0.25, and 0.005, respectively, `MilitaryCombatant` is indeed the right answer type. This example shows the effect of abstraction in channeling the context in the most relevant direction to find the right answer type. In Table 5, we provide a comparison of the selected logical forms based on AgendaIL rankings and our rankings.

Method	F1	Gain	Loss
Base	50.3	69	47
Base + Conv	50.5	96	56
Base + Abs	52.2	184	87
Base + Abs + Conv	52.6	203	93
AgendaIL	49.7	-	-

Table 6: Ablation analysis of modules of our method. Gain/Loss columns denote the number of questions where the F1 score of our selected logical form is greater/less than that of the top ranked logical forms from AgendaIL.

7.4 Ablation Analysis

In this section, we evaluate the effect of individual components of our model. Note that the answer type prediction model described in Section 5 can work independently from question abstraction and form conversion. We develop the following variants i) Base, ii) Base + Conversion, iii) Base + Abstraction, iv) Base + Abstraction + Conversion, where Base corresponds to a model that infers answer types without employing abstraction or form conversion. We train/test each variant separately. Table 6 shows each component contributes and question abstraction does help boost the performance.

Suppose we perform answer type prediction without question abstraction, and feed "[australia] fight who in the first world war?" into the answer type prediction model (Base + Conversion). The predicted answer type is `Location`. Unfortunately, there is neither a 1-hop or 2-hop correct relation from/to `Australia` with the expected type `Location` nor a correct (with positive $F1$) candi-

Question	Topic Entity Type Prediction	Answer Type Prediction	AgendaIL Answer Type	F1 Gain
who inspired **obama**?	InfluenceNode	InfluenceNode	UsVicePresident	1.0
what are some books that **mark twain** wrote?	Author	WrittenWork	InfluenceNode	0.3
who won the **league cup** in 2002?	SportsAwardType	SportsAwardWinner	SportsLeagueSeason	1.0
what type of government does **france** use?	Country	FormOfGovernment	Government	1.0
where are the **new orleans hornets** moving to?	SportsTeam	SportsFacility	Location	1.0
who did **australia** fight in the first world war?	MilitaryCombatant	MilitaryCombatant	MilitaryCommander	0.4
what guitar does **corey taylor** play?	Musician	MusicalInstrument	Organization	0.33
what region is **turkey** considered?	Location	AdministrativeDivision	Beer	0.93
what country does **rafael nadal** play for?	Athlete	Country	OlympicDiscipline	1.0

Table 4: Example questions where our type prediction helps select a better logical form. The F1 gain shows the difference between the F1 score of the logical form we select and the top ranked logical form from AgendaIL.

Questions and Selected Logical Forms
1. what are some books that **mark twain** wrote? AgendaIL: (MarkTwain - Influence.InfluenceNode.InfluencedBy - ?) Ours: (MarkTwain - Book.Author.WorksWritten - ?)
2. what guitar does **corey taylor** play? AgendaIL: (? - Organization.Organization.Founders - CoreyTaylor) Ours: (CoreyTaylor - Music.GroupMember.InstrumentsPlayed - ?)
3. what type of government does **france** use? AgendaIL: (France - Government.GovernmentalJurisdiction.Government - ?) Ours: (France - Location.Country.FormOfGovernment - ?)

Table 5: Comparison of selected logical forms for some examples. Logical forms are simplified and canonicalized into (subject - predicate - object) format for better readability, where ? corresponds to answer nodes.

date logical form with the answer type Location. This shows that through question abstraction, a better logical form is selected for this question.

To exemplify another benefit of question abstraction, consider the question "where does [marta] play soccer?" The top 3 entity linkings via Freebase Search API for "marta" are MetropolitanAtlantaRapidTransitAuthority, Marta, and SantaMarta, where the correct entity is the second one. Our question abstraction system returns FootballPlayer as the top topic entity type prediction that is indeed corresponding to the correct entity. Utilizing the context via question abstraction we are able to recover useful information when the entity linking is uncertain.

Table 6 also shows that the conversion to statement form also helps, especially together with *Abstraction*. In the above example, the model without *Conversion* (Base + Abs) predicts the answer type for "where does [football player] play soccer" as SportsFacility, whereas the full model, considering Conversion as well, finds the answer type for "[football player] play soccer where" as SportsTeam which is the better type in this case.

7.5 Error Analysis

We present a further analysis of our approach by classifying the type inference errors made on randomly sampled 100 questions. 9% of the errors are due to inference at incorrect granularity (e.g., City instead of Location). 12% of the errors are the result of incorrect answer labels (hence incorrect answer types) or question ambiguity (e.g., "where is dwight howard now?"). 11% of them are incorrect, but acceptable inferences, e.g., BookWrittenWork instead of BookEdition for question "what dawkins book to read first?" 39% of the errors are due to the sparsity problem: They are made on questions whose answer type appears less than 5 times in the training data (e.g., DayOfYear). The remaining 29% of them are due to incorrect question abstraction. In most of the question abstraction errors, the predicted topic entity type is semantically close to the correct type. In other cases such as "what did joey jordison play in slipknot?" where we predict FilmActor as the topic entity type while Musician is the correct one. In these cases, the answer type inference is not able to correct the abstraction error. These 29% of errors also contain the entity linking errors.

8 Related Work

Freebase QA has been studied from two different perspectives: grounded QA systems that work directly on KBs and general purpose ungrounded QA systems. Kwiatkowski et al. (2013) generates KB agnostic intermediary CCG parses of questions which are grounded afterwards given a KB. Bordes et al. (2014) uses a vector space embedding approach to measure the semantic similarity between question and answers. Yao and Van Durme (2014), Bast and Haussmann (2015) and Yih et al. (2015) exploit a graph centric approach where a grounded subgraph query is generated from question and then executed against a KB. In this work, we propose a neural answer type inference method that can be incorporated in existing grounded semantic parsers as a complementary feature to improve ranking of the candidate logical forms.

Berant and Liang (2015) uses lambda DCS logical language with predicates from Freebase. In their approach, types are included as a part of unary lexicon for building the logical forms from natural language questions. However, no explicit type inference is exploited. We show that such information could indeed be useful for selecting logical forms.

There have been a series of studies investigating the expected answer type of a question in different contexts such as Li and Roth (2002), Lally et al. (2012), and Balog and Neumayer (2012). Most of these approaches classify the questions into a small set of types. Even when the set of classes is more fine-grained, e.g., 50 classes in Li and Roth (2002), they cannot be used for our purpose as it would require nontrivial mapping between these categories and a much larger number of KB types. Furthermore, these methods often rely on a rich set of hand crafted features and external resources.

Sun et al. (2015) uses Freebase types to learn the relevance of candidate answers to a given question via an association model. Their model directly ranks the answer candidates by utilizing types, whereas ours ranks the logical forms via predicting answer type. In this sense, we are able to take advantage of both logical form and type inference. Su et al. (2015) exploits answer typing to facilitate knowledge graph search, but their input is graph query instead of natural language question. They predict answer types using additional relevance feedback for graph queries, while our algorithm directly infers answer types from input questions. On the question abstraction side, our work is related to a recent study (Dong et al., 2015a) which classifies entity mentions into 22 types derived from DBpedia. They use a multilayer perceptron over a fixed size window and a recurrent neural network for the representations of context and entity mention, respectively. Instead, we use a bidirectional LSTM network to exploit the full context more flexibly.

9 Conclusion

In this paper, we present a question answer type inference framework and leverage it to improve semantic parsing. We define the notion of abstract question as the class of questions that can be answered by type instead of entity. Question answer type inference is then reduced to "question abstraction" and "abstract question answering", both of which are formulated as classification problems. Question abstraction is performed by exploiting the topic entity and its context in question via an LSTM network . A separate neural network is trained to exploit the abstraction to make the final question answer type inference. Our method improves the ranking of logical forms returned by AgendaIL on the WEBQUESTIONS dataset. In the future, we would like to investigate how the abstraction and explicit type inference can be incorporated in the early stage of semantic parsing for generating better candidate logical forms.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments, and Huan Sun for fruitful discussions. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053, NSF IIS 1528175, and NSF CCF 1548848. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference (ISWC)*.

Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *ArXiv*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *ArXiv*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *ACM International Conference on Web Search and Data mining (WSDM)*.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015a. A hybrid neural model for type classification of entity mentions. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015b. Question answering over freebase with multi-column convolutional neural networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alex Graves, 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer Berlin Heidelberg.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Adam Lally, John M Prager, Michael C McCord, BK Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *International Conference on Computational Linguistics (COLING)*.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Marta Recasens, Marie catherine De Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics (TACL)*.

Yu Su, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Sue Kase, Michelle Vanni, and Xifeng Yan. 2015. Exploiting relevance feedback in knowledge graph search. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *World Wide Web (WWW)*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *World Wide Web (WWW)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *ArXiv*.

Tijmen Tieleman and Geoffrey E. Hinton. 2012. Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. *Technical Report*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structure learning algorithms applied to entity linking. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledged-based question answering. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wen-tau Yih, MingWei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Semantic Parsing to Probabilistic Programs for Situated Question Answering

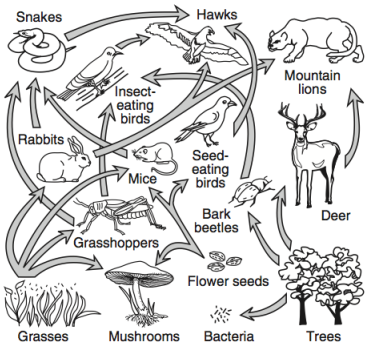
Jayant Krishnamurthy and **Oyvind Tafjord** and **Aniruddha Kembhavi**
Allen Institute for Artificial Intelligence
jayantk,oyvindt,anik@allenai.org

Abstract

Situated question answering is the problem of answering questions about an environment such as an image or diagram. This problem requires jointly interpreting a question and an environment using background knowledge to select the correct answer. We present Parsing to Probabilistic Programs (P^3), a novel situated question answering model that can use background knowledge and global features of the question/environment interpretation while retaining efficient approximate inference. Our key insight is to treat semantic parses as probabilistic programs that execute nondeterministically and whose possible executions represent environmental uncertainty. We evaluate our approach on a new, publicly-released data set of 5000 science diagram questions, outperforming several competitive classical and neural baselines.

1 Introduction

Situated question answering is a challenging problem that requires reasoning about uncertain interpretations of both a question and an environment together with background knowledge to determine the answer. To illustrate these challenges, consider the 8th grade science diagram questions in Figure 1, which are motivated by the Aristo project (Clark and Etzioni, 2016). These questions require both computer vision to interpret the diagram and compositional question understanding. These components, being imperfect, introduce uncertainty that must be jointly reasoned about to avoid implausible interpretations. These uncertain interpretations must further be combined with background knowledge, such as the definition of a "predator," to determine the correct answer.



1. According to the given food chain, what is the number of organisms that eat deer? (A) 3 (B) 2 (C) 4 **(D) 1**

2. Which organism is both predator and prey? (A) Bark Beetles **(B) Insect-eating birds** (C) Deer (D) Hawks

3. Based on the given food web, what would happen if there were no insect-eating birds? **(A) The grasshopper population would increase.** (B) The grasshopper population would decrease. (C) There would be no change in grasshopper number.

Figure 1: Example food web questions. A food web depicts a collection of organisms in an ecosystem with an arrow from organism x to y indicating that y eats x. Questions may require counting (1), knowing animal roles (2) and reasoning about population changes (3).

The challenges of situated question answering have not been completely addressed by prior work. Early "possible worlds" models (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013; Malinowski and Fritz, 2014) were capable of compositional question understanding and using background knowledge, but did not jointly reason about environ-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 160–170,
Austin, Texas, November 1-5, 2016.

ment/question uncertainty. These models also used unscalable inference algorithms for reasoning about the environment, despite the lack of joint reasoning. More recent neural models (Antol et al., 2015; Malinowski et al., 2015; Yang et al., 2015) are incapable of using background knowledge and it remains unclear to what extent these models can represent compositionality in language.

We present Parsing to Probabilistic Programs (P^3), a novel approach to situated question answering that addresses these challenges. It is motivated by two observations: (1) situated question answering can be formulated as semantic parsing with an execution model that is a *learned function of the environment*, and (2) *probabilistic programming* is a natural and powerful method for specifying the space of permissible execution models and learning over it. In P^3, we define a domain theory for the task as a probabilistic program, then train a joint loglinear model to semantically parse questions to logical forms in this theory and execute them in an environment. Importantly, the model includes global features over parsing and execution that enable it to avoid unlikely joint configurations. P^3 leverages semantic parsing to represent compositionality in language and probabilistic programming to specify background knowledge and perform linear-time approximate inference over the environment.

We present an experimental evaluation of P^3 on a new data set of 5000 food web diagram questions (Figure 1). We compare our approach to several baselines, including possible worlds and neural network approaches, finding that P^3 outperforms both. An ablation study demonstrates that global features help the model achieve high accuracy. We also demonstrate that P^3 improves accuracy on a previously published data set. Finally, we have released our data and code to facilitate further research.

2 Prior Work

Situated question answering is often formulated in terms of parsing both the question and environment into a common meaning representation where they can be combined to select the answer. This general approach has been implemented using different meaning representations:

Possible world models use a logical meaning representation defined by a knowledge base schema. These models train a semantic parser to map questions to queries and an environment model to map environments to knowledge bases in this schema. Executing the queries against the knowledge bases produces answers. These models assume that the parser and environment model are independent and furthermore that the knowledge base consists of independent predicate instances (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013; Malinowski and Fritz, 2014). Despite these strong independence assumptions, these models have intractable inference. An exception is Seo et al., (2015) who incorporate hard constraints on the joint question/environment interpretation; however, this approach does not generalize to soft constraints or arbitrary logical forms. In some work only the environment model is learned (Kollar et al., 2010; Tellex et al., 2011; Howard et al., 2014b; Howard et al., 2014a; Berant et al., 2014; Krishnamurthy and Mitchell, 2015).

Neural networks use a vector meaning representation that encodes both the question and environment as vectors. These networks have mostly been applied to visual question answering (Antol et al., 2015), where many architectures have been proposed (Malinowski et al., 2015; Yang et al., 2015; Fukui et al., 2016). It is unclear to what extent these networks can represent compositionality in language using their vector encodings. Dynamic Neural Module Networks (Andreas et al., 2016a; Andreas et al., 2016b) are the exception to the above generalization. This approach constructs a neural network to represent the meaning of the question via semantic parsing, then executes this network against the image to produce an answer. Our approach is similar except that we construct and execute a probabilistic program. Advantages of our approach are that it naturally represents the discrete structure of food webs and can use background knowledge.

Preliminaries for our work are semantic parsing and probabilistic programming. Semantic parsing translates natural language questions into executable logical forms and has been used in applications such as question answering against a knowledge base (Zelle and Mooney, 1993; Zettlemoyer and Collins, 2005; Liang et al., 2011; Kwiatkowski et al., 2013; Berant et al., 2013; Reddy et al., 2014; Yih et al., 2015; Xu et al., 2016), direction following

(Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013), and information extraction (Krishnamurthy and Mitchell, 2012; Choi et al., 2015). Semantic parsing alone is insufficient for situated question answering because it does not interpret the environment; many of the above approaches use semantic parsing as a component in a larger model.

Probabilistic programming languages extend programming languages with primitives for nondeterministic choice (McCarthy, 1963; Goodman and Stuhlmüller, 2014). We express logical forms in a probabilistic variant of Scheme similar to Church (Goodman et al., 2008); however, this paper uses Python-like pseudocode for clarity. The language has a single choice primitive called `choose` that nondeterministically returns one of its arguments. For example `choose(1,2,3)` can execute three ways, returning either 1, 2, or 3. Multiple calls to `choose` can be combined. For example, `choose(1,2)+choose(1,2)` adds two nondeterministically chosen values, and therefore has four executions that return 2, 3, 3 and 4. Each execution also has a probability; in our case, these probabilities are assigned by a trained model given the environment and not explicitly specified in the program.

3 Parsing to Probabilistic Programs (P^3)

The P^3 model is motivated by two observations. The first is that situated question answering can be formulated as semantic parsing with an execution model that is a learned function of the environment. Consider the first question in Figure 1. The meaning of this question could be represented by a logical form such as $\text{COUNT}(\lambda x.\text{EATS}(x, \text{DEER}))$, which we could train a semantic parser to predict given a suitable *domain theory* of functions such as COUNT and EATS. However, the information required to execute this logical form and answer the question must be extracted from the diagram. Specifically, $\text{EATS}(x, y)$ depends on whether an arrow is present between x and y, which we must train a vision model to determine. Thus, EATS should be a *learned function of the environment*.

This first observation suggests a need for a formalism for representing uncertainty and performing learning over the domain theory's functions. Our second observation is that probabilistic programming is a natural fit for this task. In this paradigm, the domain theory is a probabilistic program that defines the information to be extracted from the environment by using `choose`. To a first approximation, the diagram question theory includes:

```
def eats(x, y)
  choose(true, false)
```

Logical forms are then probabilistic programs, each of whose possible executions represents a different interpretation of the environment. For example, executing $\text{EATS}(\text{LION}, \text{DEER})$ hits the `choose` in the above definition, resulting in two executions where the lion either eats or does not eat the deer. In the COUNT example above, each execution represents a different set of animals that eat the deer. To learn the correct environment interpretation, we train an execution model to assign a probability to each execution given features of the environment. Using probabilistic programming enables us to combine learned functions, such as EATS, with background knowledge functions, such as COUNT, and also facilitates inference.

According to these observations, applying P^3 has two steps. The first step is to define an appropriate domain theory. This theory is the main design decision in instantiating P^3 and provides a powerful way to encode domain knowledge. The second step is to train a loglinear model consisting of a semantic parser and an execution model. This model learns to semantically parse questions into logical forms in the theory and execute them in the environment to answer questions correctly. We defer discussion of the diagram question domain theory to Section 4 and focus on the loglinear model in this section.

3.1 Model Overview

The input to the P^3 model is a question and an environment and its output is a denotation, which is a formal answer to the question. P^3 is a loglinear model with two factors: a semantic parser and an execution model. The semantic parser scores syntactic parses and logical forms for the question. These logical forms are probabilistic programs with multiple possible executions (specified by the domain theory), each of which may return a different denotation. The execution model assigns a score to each of these executions given the environment. Formally,

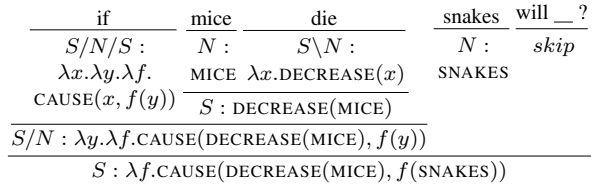


Figure 2: Example CCG parse of a question as predicted by the semantic parser f_{p}. The logical form ℓ for the question is shown on the bottom line.

the model predicts a denotation γ for a question q in an environment v using three latent variables:

$$P(\gamma|v,q;\theta) = \sum_{e,\ell,t} P(e,\ell,t|v,q;\theta)\mathbf{1}(\text{ret}(e)=\gamma)$$

$$P(e,\ell,t|v,q;\theta) = \frac{1}{Z_{q,v}} f_{\text{ex}}(e,\ell,v;\theta_{\text{ex}}) f_{\text{p}}(\ell,t,q;\theta_{\text{p}})$$

The model is composed of two factors. f_{p} represents the semantic parser that scores logical forms ℓ and syntactic parse trees t given question q and parameters θ_{p}. f_{ex} represents the execution model. Given parameters θ_{ex}, this factor assigns a score to a logical form ℓ and its execution e in environment v. The denotation γ, i.e., the formal answer to the question, is simply the value returned by e. $Z_{q,v}$ represents the model's partition function. The following sections describe these factors in more detail.

3.2 Semantic Parser

The factor f_{p} represents a Combinatory Categorial Grammar (CCG) semantic parser (Zettlemoyer and Collins, 2005) that scores logical forms for a question. Given a lexicon[1] mapping words to syntactic categories and logical forms, CCG defines a set of possible syntactic parses t and logical forms ℓ for a question q. Figure 3.2 shows an example CCG parse. f_{p} is a loglinear model over parses (ℓ, t):

$$f_{\text{p}}(\ell,t,q;\theta_{\text{p}}) = \exp\{\theta_{\text{p}}^T \phi(\ell,t,q)\}$$

The function ϕ maps parses to feature vectors. We use a rich set of features similar to those for syntactic CCG parsing (Clark and Curran, 2007); a full description is provided in an online appendix.

3.3 Execution Model

The factor f_{ex} is a loglinear model over the executions of a logical form given an environment. Logical forms in P^3 are probabilistic programs with a *set* of possible executions, where each execution e is a sequence, $e = [e_0, e_1, e_2, ..., e_n]$. e_0 is the program's starting state, e_i represents the state immediately after the ith call to `choose`, and e_n is the state at termination. The score of an execution is:

$$f_{\text{ex}}(e,\ell,v;\theta_{\text{ex}}) = \prod_{i=1}^{n} \exp\{\theta_{\text{ex}}^T \phi(e_{i-1}, e_i, \ell, v)\}$$

In the above equation, θ_{ex} represents the model's parameters and ϕ represents a feature function that produces a feature vector for the difference between sequential program states e_{i-1} and e_i given environment v and logical form ℓ. ϕ can include arbitrary features of the execution, logical form and environment, which is important, for example, to detect cycles in a food web (Section 4.3).

3.4 Inference

P^3 is designed to rely on approximate inference: our goal is to use rich features to accurately make local decisions, as in linear-time parsers (Nivre et al., 2006). We perform approximate inference using a two-stage beam search. Given a question q, the first stage performs a beam search over CCG parses to produce a list of logical forms scored by f_{p}. This step is performed by using a CKY-style chart parsing algorithm then marginalizing out the syntactic parses. The second stage performs a beam search over executions of each logical form. The space of possible executions of a logical form is a tree (Figure 4.2) where each internal node represents a partial execution up to a `choose` call. The search maintains a beam of partial executions at the same depth, and each iteration advances the beam to the next depth, discarding the lowest-scoring executions according to f_{ex} to maintain a fixed size beam. This procedure runs in time linear to the number of `choose` calls. We implement the search by rewriting the probabilistic program into continuation-passing style, which allows `choose` to be implemented as a function that adds multiple continuations to the search queue; we refer the reader to Goodman and Stuhlmüller (2014) for details. Our experiments use a beam size of 100 in the semantic parser, executing each of the 10 highest-scoring logical forms with a beam of 100 executions.

[1] In our experiments, we automatically learn the lexicon in a preprocessing step. See Section 5.2 for details.

3.5 Training

P^3 is trained by maximizing loglikelihood with stochastic gradient ascent. The training data $\{(q^i, v^i, c^i)\}_{i=1}^n$ is a collection of questions q^i and environments v^i paired with *supervision oracles* c^i. $c^i(e) = 1$ for a correct execution e and $c^i(e) = 0$ otherwise. The oracle c^i can implement various kinds of supervision, including: (1) labeled denotations, by verifying the value returned by e and (2) labeled environments, by verifying each choice made by e. The oracle for diagram question answering combines both forms of supervision (Section 4.5).

The objective function O is the loglikelihood of predicting a correct execution:

$$O(\theta) = \sum_{i=1}^{n} \log \sum_{e,l,t} c^i(e) P(e, \ell, t | q^i, v^i; \theta)$$

We optimize this objective function using stochastic gradient ascent, using the approximate inference algorithm from Section 3.4 to estimate the necessary marginals. When computing the marginal distribution over correct executions, we filter each step of the beam search using the supervision oracle c^i to improve the approximation.

4 Diagram Question Answering with P^3

As a case study, we apply P^3 to the task of answering food web diagram questions from an 8th grade science domain. A few steps are required to apply P^3. First, we create a domain theory of food webs that represents extracted information from the diagram and background knowledge for the domain. Second, we define the features of the execution model that are used to learn how programs in the domain theory execute given a diagram. Third, we define a component to select a multiple-choice answer given a denotation. Finally, we define the supervision oracle used for training.

4.1 Food Web Diagram Questions

We consider the task of answering food web diagram questions. The input consists of a diagram depicting a food web, a natural language question and a list of natural language answer options (Figure 1). The goal is to select the correct answer option. This task has many regularities that require global features: for example, food webs are usually acyclic and certain animals usually have certain roles (e.g., mice are herbivores). We have collected and released a data set for this task (Section 5.1).

We preprocess the diagrams in the data set using a computer vision system that identifies candidate diagram elements (Kembhavi et al., 2016). This system extracts a collection of text labels (via OCR), arrows, arrowheads and objects, each with corresponding scores. It also extracts a collection of scored linkages between these elements. These extractions are noisy and contain many discrepancies such as overlapping text labels and spurious linkages. We use these extractions to define a set of candidate organisms (using the text labels), and also to define features of the execution model.

4.2 Domain Theory

The domain theory is a probabilistic program encoding the information to extract from the environment as well as background knowledge about food webs. It represents the structure of a food web using two functions. These functions are predicates that invoke `choose` to return either true or false. The execution model learns to predict which of these values is correct for each set of arguments given the diagram. It furthermore has a collection of deterministic functions that encode domain knowledge, including definitions of animal roles such as HERBIVORE and a model of population change causation.

Figure 4.2 shows pseudocode for a portion of the domain theory. Food webs are represented using two functions over the extracted text labels: ORGANISM(x) indicates whether the label x is an organism (as opposed to, e.g., the diagram title); and EATS(x, y). The definitions of these functions invoke `choose` while remembering previously chosen values to avoid double counting probabilities when executing logical forms such as ORGANISM(DEER) $\wedge$ ORGANISM(DEER). The remembered values are stored in a global variable that is also used to implement the supervision oracle. Deterministic functions such as CAUSE are defined in terms of these learned functions.

The uses of `choose` in the domain theory create a tree of possible executions for every logical form. Figure 4.2 illustrates this tree for the logical form λf.CAUSE(DECREASE(MICE), f(SNAKES)), which

```
# initialize predicate instance variables
# from text labels in environment
world = {"mice": undef,
         ("mice", "snakes"): undef, ...}
def organism(name)
  if (world[name] == undef)
    world[name] = choose(true, false)
  return world[name]

def eats(x, y)
  # same as organism but with pairs.

# entities referenced in the logical form
# must be organisms. choose() represents
# failure; it returns no values.
def getOrganism(x)
  if (organism(x)) return x else choose()

# change events are direction/
# text label tuples
def decrease(x)
  return ("decrease", x)

def cause(e1, e2)
  e12 = eats(e1[1], e2[1])
  e21 = eats(e2[1], e1[1])
  # deterministic model with cases. e.g.
  # if eats(y, x) then (cause (decrease x)
  # (decrease y)) -> true
  return doCause(e1[0], e2[0], e12, e21)
```

Figure 3: Domain theory pseudocode for diagram question answering.

corresponds to the question "what happens to the snakes when the mice decrease?" This logical form is shorthand for the following program:

```
filter(lambda f.cause(
    decrease(getOrganism("mice")),
    f(getOrganism("snakes"))),
  set(decrease, increase, unchanged))
```

Specifically, entities such as MICE are created by calling `getOrganism` and logical forms with functional types implicitly represent filters over the appropriate argument type. Executing this program first applies the filter predicate to `decrease`. Next, it evaluates `getOrganism("mice")`, which calls `organism` and encounters the first call to `choose`. This call is shown as the first branch of the tree in Figure 4.2. The successful branch proceeds to evaluate `getOrganism("snakes")`, shown as the second branch. Finally, the successful branch evaluates `cause`, which calls `eats` twice, resulting in the final two branches. The value returned by each branch is determined by the causation model which performs

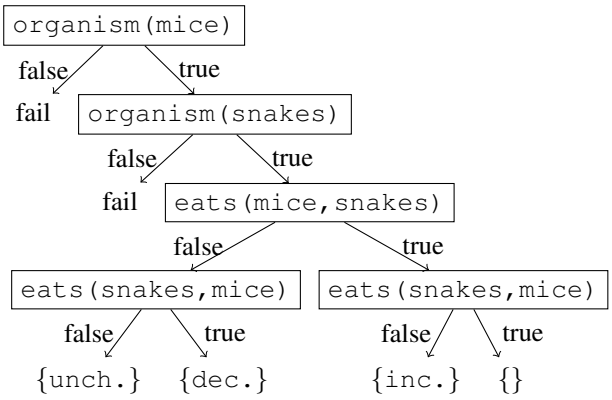


Figure 4: Tree of possible executions for the logical form $\lambda f.\text{CAUSE}(\text{DECREASE}(\text{MICE}), f(\text{SNAKES}))$. Each path from root to leaf represents a single execution that returns the indicated denotation or fails, and each internal node represents a nondeterministic choice made with `choose`.

some deterministic logic on the truth values of the two eats relations.

4.3 Execution Features

The execution model uses three sets of features: instance features, predicate features, and denotation features. Instance features treat each predicate instance independently, while the remainder are global features of multiple predicate instances and the logical form. We provide a complete listing of features in an online appendix.

Instance features fire whenever an execution chooses a truth value for a predicate instance. These features are similar to the per-predicate-instance features used in prior work to produce a distribution over possible worlds. For $\text{ORGANISM}(x)$, our features are the vision model's extraction score for x and indicator features for the number of tokens in x. For $\text{EATS}(x, y)$, our features are various combinations of the vision model's scores for arrows that may connect the text labels x and y.

Predicate features fire based on the global assignment of truth values to all instances of a single predicate. The features for ORGANISM count occurrences of overlapping text labels among true instances. The features for EATS include cycle count features for various cycle lengths and arrow reuse features. The cycle count features help the model learn that food webs are typically, but not always, acyclic and the arrow reuse features aim to prevent the model from predicting two different EATS instances on the basis of a single arrow.

Denotation features fire on the return value of an execution. There are two kinds of denotation features: size features that count the number of entities in denotations of various types and denotation element features for specific logical forms. The second kind of feature can be used to learn that the denotation of $\lambda x.\text{HERBIVORE}(x)$ is likely to contain MOUSE, but unlikely to contain WOLF.

4.4 Answer Selection

P^3 predicts a distribution over denotations for each question, which for our problem must be mapped to a distribution over multiple choice answers. Answer selection performs this task using string match heuristics and an LSTM (Hochreiter and Schmidhuber, 1997). The string match heuristics score each answer option given a denotation then select the highest scoring answer, abstaining in the case of a tie. The score computation depends on the denotation's type. If the denotation is a set of entities, the score is an approximate count of the number of entities in the denotation mentioned in the answer using a fuzzy string match. If the denotation is a set of change events, the score is a fuzzy match of both the change direction and the animal name. If the denotation is a number, string matching is straightforward. Applying these heuristics and marginalizing out denotations yields a distribution over answer options.

A limitation of the above approach is that it does not directly incorporate linguistic prior knowledge about likely answers. For example, "snake" is usually a good answer to "what eats mice?" regardless of the diagram. Such knowledge is known to be essential for visual question answering (Antol et al., 2015; Andreas et al., 2016b) and important in our task as well. We incorporate this knowledge in a standard way, by training a neural network on question/answer pairs (without the diagram) and combining its predictions with the string match heuristics above. The network is a sequence LSTM that is applied to the question concatenated with each answer option a to produce a 50-dimensional vector v_a for each answer. The distribution over answers is the softmax of the inner product of these vectors with a learned parameter vector w. For simplicity, we combine these two components using a 50/50 mix of their answer distributions.

4.5 Supervision Oracle

The supervision oracle for diagram question answering combines supervision of both answers and environment interpretations. We assume that each diagram has been labeled with a food web. An execution is correct if and only if (1) all of the chosen values in the global variable encoding the food web are consistent with the labeled food web, and (2) string match answer selection applied to its denotation chooses the correct answer. The first constraint guarantees that every logical form has at most one correct execution for any given diagram.

5 Evaluation

Our evaluation compares P^3 to both possible worlds and neural network approaches on our data set of food web diagram questions. An ablation study demonstrates that both sets of global features improve accuracy. Finally, we demonstrate P^3's generality by applying it to a previously-published data set, obtaining state-of-the-art results.

Code, data and supplementary material for this paper are available at: `http://www.allenai.org/paper-appendix/emnlp2016-p3`

5.1 FOODWEBS Data Set

FOODWEBS consists of ∼500 food web diagrams and ∼5000 questions designed to imitate actual questions encountered on 8th grade science exams. The train/validation/test sets contain ∼300/100/100 diagrams and their corresponding questions. The data set has three kinds of annotations in addition to the correct answer for each question. First, each diagram is annotated with the food web that it depicts using ORGANISM and EATS. Second, each diagram has predictions from a vision system for various diagram elements such as arrows and text labels (Kembhavi et al., 2016). These are noisy predictions, not ground truth. Finally, each question is annotated by the authors with a logical form (or null if its meaning is not representable in the domain theory). These logical forms are not used to train P^3 but are useful to measure per-component error.

We collected FOODWEBS by using a crowdsourcing process to expand a collection of real exam questions. First, we collected 89 questions from 4th and 8th grade exams and 500 food web diagrams us-

ing an image search engine. Second, we generated questions for these diagrams using Mechanical Turk. Workers were shown a diagram and a real question for inspiration and asked to write a new question and its answer options. We validated each generated question by asking 3 workers to answer it, discarding questions where at least 2 did not choose the correct answer. We also manually corrected any ambiguous (e.g., two answer options are correct) and poorly-formatted (e.g., two answer options have the same letter) questions. The final data set has high quality: a human domain expert correctly answered 95 out of 100 randomly-sampled questions.

5.2 Baseline Comparison

Our first experiment compares P^3 with several baselines for situated question answering. The first baseline, WORLDS, is a possible worlds model based on Malinowski and Fritz (2014). This baseline learns a semantic parser $P(\ell, t|q)$ and a distribution over food webs $P(w|v)$, then evaluates ℓ on w to produce a distribution over denotations. This model is implemented by independently training P^3's CCG parser (on question/answer pairs and labeled food webs) and a possible-worlds execution model (on labeled food webs). The CCG lexicon for both P^3 and WORLDS was generated by applying PAL (Krishnamurthy, 2016) to the same data. Both models select answers as described in Section 4.4.

We also compared P^3 to several neural network baselines. The first baseline, LSTM, is the text-only answer selection model described in Section 4.4. The second baseline, VQA, is a neural network for visual question answering. This model represents each image as a vector by using the final layer of a pre-trained VGG19 model (Simonyan and Zisserman, 2014) and applying a single fully-connected layer. It scores answer options by using the answer selection LSTM to encode question/answer pairs, then computing a dot product between the text and image vectors. This model is somewhat limited because VGG features are unlikely to encode important diagram structure, such as the content of text labels. Our third baseline, DQA, is a neural network that rectifies this limitation (Kembhavi et al., 2016). It encodes the diagram predictions from the vision system as vectors and attends to them using the LSTM-encoded question vector to select an answer. This model is trained with question/answer pairs and diagram parses, which is roughly comparable to the supervision used to train P^3.

Model	Accuracy	Accuracy (Unseen Organisms)
P^3	**69.1**	**57.7**
WORLDS	63.6	50.8
LSTM	60.3	34.7
VQA	56.5	36.8
DQA	59.3	33.0
Random	25.2	25.2

Table 1: Accuracy of P^3 and several baselines on the FOODWEBS test set and a modified test set with unseen organisms.

Model	Accuracy	Δ
P^3	69.1	
-LSTM	59.8	-9.3
-LSTM -denotation	55.8	-13.3
-LSTM -denotation -predicate	52.4	-16.7

Table 2: Test set accuracy of P^3 removing LSTM answer selection (Section 4.4), denotation features and predicate features (Section 4.3).

Table 5.2 compares the accuracy of P^3 to these baselines. Accuracy is the fraction of questions answered correctly. LSTM performs well on this data set, suggesting that many questions can be answered without using the image. This result is consistent with results on visual question answering (Antol et al., 2015). The other neural network models have similar performance to LSTM, whereas both WORLDS and P^3 outperform it. We also find that P^3 outperforms WORLDS likely due to its global features, which we investigate in the next section.

Given these results, we hypothesized that the neural models were largely memorizing common patterns in the text and were not able to interpret the diagram. We tested this hypothesis by running each model on a test set with unseen organisms created by reversing the organism names in every question and diagram (Table 5.2, right column). As expected, the accuracy of LSTM is considerably reduced on this data set. VQA and DQA again perform similarly to LSTM, which is consistent with our hypothesis. In contrast, we find that the accuracies of WORLDS and P^3 are only slightly reduced, which is consistent with superior diagram interpretation abilities but ineffective LSTM answer selection.

5.3 Ablation Study

We performed an ablation study to further understand the impact of LSTM answer selection and global features. Table 5.2 shows the accuracy of P^3 trained without these components. We find that LSTM answer selection improves accuracy by 9 points, as expected due to the importance of linguistic prior knowledge. Global features improve accuracy by 7 points, which is roughly comparable to the delta between P^3 and WORLDS in Table 5.2.

5.4 Component Error Analysis

Our third experiment analyses sources of error by training and evaluating P^3 while providing the gold logical form, food web, or both as input. Table 5.5 shows the accuracy of these three models. The final entry shows the maximum accuracy possible given our domain theory and answer selection. The larger accuracy improvement with gold food webs suggests that the execution model is responsible for more error than semantic parsing, though both components contribute.

5.5 SCENE Experiments

Our final experiment applies P^3 to the SCENE data set of Krishnamurthy and Kollar (2013). In this data set, the input is a natural language expression, such as "blue mug to the left of the monitor," and the output is the set of objects in an image that the expression denotes. The images are annotated with a bounding box for each candidate object. The data set includes a domain theory that was automatically generated by creating a category and/or relation per word based on its part of speech. It also includes a CCG lexicon and image features. We use these resources, adding predicate and denotation features.

Table 5.5 compares P^3 to prior work on SCENE. The evaluation metric is exact match accuracy between the predicted and labeled sets of objects. We consider three supervision conditions: QA trains with question/answer pairs, QA+E further includes labeled environments, and QA+E+LF further includes labeled logical forms. We trained P^3 in the first two conditions, while prior work trained in the first and third conditions. KK2013 is a possible worlds model with a max-margin training objective. P^3 slightly outperforms in the QA condition and P^3 trained with labeled environments outperforms prior work trained with additional logical form labels.

Model	Accuracy	Δ
P^3	69.1	
+ gold logical form	75.1	+6.0
+ gold food web	82.3	+13.2
+ both	91.6	+22.5

Table 3: Accuracy of P^3 when trained and evaluated with labeled logical forms, food webs, or both.

	Supervision		
Model	QA	QA+E	QA+E+LF
P^3	68	75	–
KK2013	67	–	70

Table 4: Accuracy on the SCENE data set. KK2013 results are from Krishnamurthy and Kollar (2013).

6 Conclusion

Parsing to Probabilistic Programs (P^3) is a novel model for situated question answering that jointly reasons about question and environment interpretations using background knowledge to produce answers. P^3 uses a domain theory – a probabilistic program – to define the information to be extracted from the environment and background knowledge. A semantic parser maps questions to logical forms in this theory, which are probabilistic programs whose possible executions represent possible interpretations of the environment. An execution model scores these executions given features of the environment. Both the semantic parser and execution model are jointly trained in a loglinear model, which thereby learns to both parse questions and interpret environments. Importantly, the model includes global features of the logical form and executions, which help the model avoid implausible interpretations. We demonstrate P^3 on a challenging new data set of 5000 science diagram questions, where it outperforms several competitive baselines.

Acknowledgments

We gratefully acknowledge Minjoon Seo, Mike Salvato and Eric Kolve for their implementation help, Isaac Cowhey and Carissa Schoenick for their help with the data, and Oren Etzioni, Peter Clark, Matt Gardner, Hannaneh Hajishirzi, Mike Lewis, and Jonghyun Choi for their comments.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Deep compositional question answering with neural module networks. In *CVPR*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Learning to compose neural networks for question answering. In *NAACL*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 2015 Association for Computational Linguistics*.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37:5–12.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv:1606.01847*.

Noah D Goodman and Andreas Stuhlmüller. 2014. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Accessed: 2016-2-25.

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. 2008. Church: A language for generative models. In *Uncertainty in Artificial Intelligence*.

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

Thomas M. Howard, Istvan Chung, Oron Propp, Matthew R. Walter, and Nicholas Roy. 2014a. Efficient natural language interfaces for assistive robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on Rehabilitation and Assistive Robotics*, September.

Thomas M Howard, Stefanie Tellex, and Nicholas Roy. 2014b. A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association of Computational Linguistics – Volume 1*.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In *NAACL*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to

answering questions about images. In *International Conference on Computer Vision*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning*.

John McCarthy. 1963. A basis for a mathematical theory of computation. In *Computer Programming and Formal Systems*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the Association for Computational Linguistics (ACL 2016)*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

John M. Zelle and Raymond J. Mooney. 1993. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Event participant modelling with neural networks

Ottokar Tilk
Institute of Cybernetics
Tallinn University of Technology
12618 Tallinn, Estonia
ottokar.tilk@phon.ioc.ee

Vera Demberg and **Asad Sayeed**
and **Dietrich Klakow** and **Stefan Thater**
Saarland University
66123 Saarbrücken, Germany
{vera,asayeed,stth}
@coli.uni-sb.de;
dietrich.klakow@lsv.uni-sb.de

Abstract

A common problem in cognitive modelling is lack of access to accurate broad-coverage models of event-level surprisal. As shown in, e.g., Bicknell et al. (2010), event-level knowledge does affect human expectations for verbal arguments. For example, the model should be able to predict that *mechanics* are likely to check *tires*, while *journalists* are more likely to check *typos*. Similarly, we would like to predict what locations are likely for *playing football* or *playing flute* in order to estimate the surprisal of actually-encountered locations. Furthermore, such a model can be used to provide a probability distribution over fillers for a thematic role which is not mentioned in the text at all.

To this end, we train two neural network models (an incremental one and a non-incremental one) on large amounts of automatically role-labelled text. Our models are probabilistic and can handle several roles at once, which also enables them to learn interactions between different role fillers. Evaluation shows a drastic improvement over current state-of-the-art systems on modelling human thematic fit judgements, and we demonstrate via a sentence similarity task that the system learns highly useful embeddings.

1 Introduction

Our goals in this paper are to learn a representation of events and their thematic roles based on large quantities of automatically role-labelled text and to be able to calculate probability distributions over the possible role fillers of specific missing roles. In this sense, the task is closely related to work on selectional preference acquisition (Van de Cruys, 2014). We focus here on the roles *agent*, *patient*, *location*, *time*, *manner* and the *predicate* itself. The model we develop is trained to represent the event-relevant context and hence systematically captures long-range dependencies. This has been previously shown to be beneficial also for more general language modelling tasks (e.g., Chelba and Jelinek, 1998; Tan et al., 2012).

This type of modelling is potentially relevant to a wide range of tasks, for instance for performing thematic fit judgment tasks, detecting anomalous events (Dasigi and Hovy, 2014), or predicting event structure that is not explicitly present in the text. The latter could be useful for inferring missing information in entailment tasks or improving identification of thematic roles outside the sentence containing the predicate. Potential applications also include predicate prediction based on arguments and roles, which has been noted to be relevant for simultaneous machine translation for a verb-final to a verb-medial source language (Grissom II et al., 2014). Within cognitive modelling, our model could help to more accurately estimate *semantic surprisal* for broad-coverage texts, when used in combination with an incremental role labeller (e.g., Konstas and Keller, 2015), or to provide surprisal estimates for content words as a control variable for psycholinguistic experimental materials.

In this work, we focus on the predictability of verbs and nouns, and we suggest that the predictability of these words depends to a large extent on the relationship of these words to other nouns and

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 171–182,
Austin, Texas, November 1-5, 2016.

verbs, especially those connected via the same event. We choose a neural network (NN) model because we found that results from existing related models, e.g. Baroni and Lenci's Distributional Memory, depend heavily on how exactly the distributional space is defined, while having no principled way of optimizing the space. A crucial advantage of a neural network-based approach is thus that the model can be trained to optimize the distributional representation for the task.

Our model is trained specifically to predict missing semantic role-fillers based on the predicate and other available role-fillers of that predicate. The model can also predict the predicate based on the semantic roles and their fillers. In our model, there is no difference in how the semantic roles or the predicate are treated. Thus, when we refer here to *roles*, we usually mean both semantic roles and the predicate, unless otherwise explicitly stated.

Our model is compositional in that it has access to several role-fillers (including the verb) at the same time, and can thus represent interdependencies between participants of an event and predict from a combined representation. Consider, for example, the predicate *serve*, whose likely patients include e.g., *drinks*. If we had the agent *robber*, we would like to be able to predict a patient like *sentence*, in the sense of "the robber will serve his sentence..." This task is related to modelling thematic fit. In this paper, we evaluate our model on a variety of thematic fit rating datasets as well as on a sentence similarity dataset that tests for successful compositionality in our model's representations.

This paper makes the following contributions:

- We compare two novel NN models for generating a probability distribution over selectional preferences given one or more roles and fillers.
- We show that our technique outperforms state of the art thematic fit models on many datasets.
- We show that the embeddings thus obtained are effective in measuring sentence similarity.

1.1 Neural networks

Neural networks have proven themselves to be very well suited for language modeling. By learning distributed representations of words (Bengio et al., 2003), they are able to generalize to new contexts that were not observed word-by-word in the training corpus. They can also use a relatively large number of context words in order to make predictions about the upcoming word. In fact, the recurrent neural network (RNN) LM (Mikolov et al., 2010) does not explicitly fix the context size at all but is potentially able to compress the relevant information about the entire context in its recurrent layer. These are the properties that we would like to see in our role-filler prediction model as well.

Neural networks have also been used for selectional preference acquisition, as in Van de Cruys (2014). His selectional preference model differs from our model in several aspects. First, unlike our model it is limited to a fixed number of inputs. Another difference is that his model uses separate embeddings for all input words, while ours enables partial parameter sharing. Finally and crucially for role-filler prediction, selectional preference models score the inputs, while our model gives a probability distribution over all words for the queried target role.

We discuss the components necessary for our model in more detail in section 3.

2 Data source

Our source of training data is the ukWaC corpus, which is part of the WaCky project, as well as the British National Corpus. The corpus consists of web pages crawled from the `.uk` web domain, containing approximately 138 million sentences.

These sentences were run through a semantic role labeller and head words were extracted as described in Sayeed et al. (2015). The semantic role labeller used, SENNA (Collobert and Weston, 2007), generates PropBank-style role labels. While PropBank argument positions (ARG0, ARG1, etc.) are primarily designed to be verb-specific, rather than directly representing "classical" thematic roles (agent, patient, etc.), in the majority of cases, ARG0 lines up with agent roles and ARG1 lines up with patient roles. PropBank-style roles have been used in other recent efforts in thematic fit modelling (e.g., Baroni et al., 2014; Vandekerckhove et al., 2009),

For processing purposes, the corpus was divided into 3500 segments. Fourteen segments (approx 500 thousand sentences) each were used for development and testing, and the rest were used for training.

In order to construct our incremental model and compare it to n-gram language models, we needed a precise mapping between the lemmatized argument words and their positions in the original sentence. This required aligning the SENNA tokenization and the original ukWaC tokenization used for MaltParser. Because of the heterogeneous nature of web data, this alignment was not always achievable—we skipped a small number of sentences in this case. In the development and testing portions of the data set, we filtered sentences containing predicates where there were multiple role-assignees with the same role for the same predicate.

3 Model design and implementation

Our model is a neural network with a single nonlinear hidden layer and a *Softmax* output layer. All inputs are one-hot encoded—i.e., represented as a binary vector with size equal to the number of possible input values, where all entries are zero except the entry at the index corresponding to the current input value.

3.1 Two-part view of the model

The parameters of a neural network classifier with a single hidden layer and one-hot encoded inputs can be viewed as serving two distinct purposes: moving from inputs towards outputs, the first weight matrix that we encounter is responsible for learning distributed representations (or embeddings) of the inputs; the second weight matrix represents the parameters of a maximum entropy classifier that uses the learned embeddings as inputs.

Considering the task of role-filler prediction, we would want these two sets of parameters to have the following properties:

- The **classifier layer** should be different for each target role, because the suitable filler given the context can clearly be very different depending on the role (e.g., verb vs. agent).
- The **embedding layer** should also be different depending on the role of context word. Otherwise, the network would not have any information about the role of the context word. For example, the suitable verb filler for context word *dog* in an agent role is probably very different from what it would be, were it in a patient role (e.g. *bark* vs. *feed*).

We now briefly describe some incrementally improved intermediate approaches that we also considered as they help to understand the steps that led to our final solution for achieving the desired properties of the embedding and classifier layer.

A naive way to accomplish the aspired properties would be to have a separate model for each input role and target role pair. This approach has several drawbacks. For a start, there is no obvious way to model interactions of different input roles and fillers in order to make predictions based on multiple input role-word pairs simultaneously. Another problem is that the parameters are trained only on a fraction of available training data—e.g., verb embedding weights are trained independently for each target role classifier. Finally, given that we have chosen to distinguish between n different roles, it would require us to train and tune hyper-parameters for n^2 models.

One of these problems (data under-utilization) can be alleviated by sharing role-specific embedding and classifier weights across different models. For example, the verb embedding matrix would be shared across all models that predict different role fillers based on input verbs. Other problems remain, and training the large number of models becomes even more difficult because of parameter synchronization, but this is a step towards the next improvement.

Shared role-specific embedding and classifier weights enable us to combine all input-target role pair models into a single model. This can be done by stacking role-specific embedding matrices to form a 3-way embedding tensor and building a classifier parameter tensor analogously. Having a single model saves us from tuning multiple models and makes modelling interactions between inputs possible.

Despite these advantages, having two tensors in our model has a drawback of rapidly growing the number of parameters as vocabulary size, number of roles, and hidden layer size increase. This may lead to over-fitting and increases training time.

A more subtle weakness is the fact that this kind of model lacks parameter sharing across role-specific embedding weight matrices. It is clear that some characteristics of words (e.g., semantics) usu-

ally remain the same across different roles. Thus it is practical to share some information across role-specific weights so that the embeddings can benefit from more data and learn better semantic representations while leaving room for role-specific traits.

For these reasons we replace the tensors with their factored form in our models.

3.2 Factored parameter tensors

Factoring classifier and embedding tensors helps to alleviate both the efficiency and parameter sharing problems brought out in Section 3.1.

Given vocabulary size $|V|$, number of roles $|R|$ and hidden layer size H, each tensor T would require $|V| \times |R| \times H$ parameters. The number of parameters can be reduced by expressing the tensor as a sum of F rank-one tensors (Hitchcock, 1927). This technique enables us to replace the tensor T with three factor matrices A, B and C. Each tensor element $T[i, j, k]$ can then be written as:

$$T[i, j, k] = \sum_{f=1}^{F} A[i, f]B[j, f]C[f, k] \quad (1)$$

Assuming lateral slices of T represent role-specific weight matrices (index j denotes roles), we write each role specific weight matrix W as:

$$W = A \,\mathrm{diag}(rB)C \quad (2)$$

where r is a one-hot encoded role vector and $diag$ is a function that returns a square matrix with the argument vector on the main diagonal and zeros elsewhere. For example, with a vocabulary of 50000 words, 7 roles and number of factors and hidden units equal to 512, the factorization reduces the number of parameters from 179M to 26M and greatly improves training speed. Factorization also enables parameter sharing, since factor matrices A and C are shared across all roles.

Factored tensors have been used in different neural network models before. Starting with restricted Boltzmann machines, Memisevic and Hinton (2010) used a factored 3-way interaction tensor in their image transformation model. Sutskever et al. (2011) created a character level RNN LM that was efficiently able to use input character specific recurrent weights by using a factored tensor. Alumäe (2013) used a factored tensor in a multi-domain LM to be able to use a domain-specific hidden layer weight matrix that would take into account the differences while exploiting similarities between domains. A multi-modal LM by Kiros et al. (2014) uses a factored tensor to change the effective output layer weights based on image features.

It has been noticed before, that training models with factored tensors as parameters using gradient descent is difficult (Sutskever et al., 2011; Kiros et al., 2014). As explained by Sutskever et al. (2011), this is caused by the fact that each tensor element is represented as a product of three parameters, which may cause disproportionate updates if these three factors have magnitudes that are too different. Another problem is that if the factor matrix B happens to have too small or too large values, then this might also cause instabilities in the lower layers as the back-propagated gradients are scaled by role-specific row of B in our model. This situation is magnified in our models, since we have not one, but two factored layers.

To solve this problem, Sutskever et al. (2011) suggest using 2nd order methods instead of gradient descent. Alumäe (2013) has alleviated the problem of shrinking back-propagated gradients by adding a bias (initialized with ones) to the domain-specific factor vector. We found that using AdaGrad (Duchi et al., 2011) to update the parameters is very effective. The method provides parameter-specific learning rates that depend on the historic magnitudes of the gradients of these parameters. This seems to neutralize the effect of vanishing or exploding gradients by reducing the step size for parameters that tend to have large gradients and allow a bigger learning rate for parameters with smaller gradients.

3.3 General structure of the model

Our general approach, common to both role-filler models, is shown in Figure 1. First, role-specific word embedding vector e is computed by implicitly taking a fiber (word indexed row of a role indexed slice) from the factored embedding tensor:

$$e = wA_e \,\mathrm{diag}(rB_e)C_e \quad (3)$$

$$h = \mathrm{PReLU}(e + b_h) \quad (4)$$

where w and r are one-hot encoded word and role vectors respectively, b_h is hidden layer bias, and A_e,

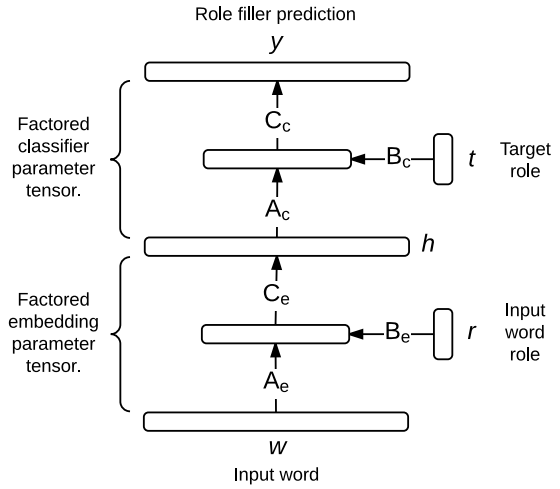


Figure 1: General structure of role-filler models.

B_e and C_e represent the factor matrices that the embedding tensor is factored into. Next, we apply a parametric rectifier (PReLU; He et al., 2015) nonlinearity to the role-specific word embedding to obtain the hidden activation vector h.

The hidden layer activation vector h is fed to the *Softmax* output layer through a target role specific classifier weight matrix (a target role-indexed slice of the classifier parameter tensor):

$$c = hA_c \,\text{diag}(tB_c)C_c \quad (5)$$

$$y = \text{Softmax}(c + b_y) \quad (6)$$

where t is a one-hot encoded target role vector, b_y is output layer bias, and y is the output of the model representing the probability distribution over the output vocabulary.

3.4 Modeling input interactions

The general approach described in Section 3.3 also allows us to model interactions between different input role-word pairs. If we know the order in which the inputs were introduced, then we can add a recurrent connection to the hidden layer to implement an incremental role filler predictor. When word order is unknown, then input role-word pair representations can be added together to compose the representation of the entire predicate context[1]. We chose addition over concatenation (often preferred in language models) because the non-incremental model does not need to preserve information about word order, and addition also enables using a variable number of inputs.

The incremental model adds information about the previous hidden state h_{t-1} to the current input word role-specific embedding e_t through recurrent weights W_r. So, Equation 4 is replaced with:

$$h_t = \text{PReLU}(e_t + h_{t-1}W_r + p_tW_p + b_h) \quad (7)$$

where p_t is a binary predicate boundary indicator that informs the model about the start of a new predicate and equals 1 when the target word belongs to a new predicate and 0 otherwise. The predicate boundary input p_t is connected to the network through parameter vector W_p. The hidden state h_0 is initialized to zeros.

The non-incremental model adds role-specific embedding vectors of all input words together to form the representation of the entire predicate context and replaces Equation 4 with:

$$h = \text{PReLU}(\sum_{i=1}^{N} e_i + b_h) \quad (8)$$

where N is the number of input role-word pairs.

3.5 Training details

First, we give details that are common to both the RNN and NN models. The models are trained with mini-batches of 128 samples. The hidden layer consists of 256 PReLU units; embedding and classifier tensor factorization layer sizes are 256 and 512 respectively. The input and output vocabularies are the same, consisting of 50,000 most frequent lemmatized words in the training corpus. The role vocabulary consists of 5 argument roles (ARG0, ARG1, ARGM-LOC, ARGM-TMP and ARGM-MNR), the verb is treated as the sixth role, and all the other roles are mapped to a shared OTHER label. Parameters are updated using AdaGrad (Duchi et al., 2011) with a learning rate of 0.1. All models are implemented using Theano (Bastien et al., 2012; Bergstra et al., 2010) and trained on GPUs for 8 days.

RNN model gradients are computed using backpropagation through time (Rumelhart et al., 1986)

[1] In applications like natural language generation, for example, where role-fillers need to be predicted, it is not necessarily always the case that the order will be known in advance or that the thematic fit model will be used to generate the full sentence in correct word order.

Model Name	Dev	Test
3-gram LM	450.1 ± 2.6	438.9 ± 2.6
3-gram CWM	859.6 ± 4.6	834.9 ± 4.5
RNN CWM	485.8 ± 2.7	473.2 ± 2.6
RNN RF	244.6 ± 1.4	237.8 ± 1.4
NN RF	248.2 ± 1.4	241.9 ± 1.4

Table 1: Perplexities on dev/test dataset.

over 3 time steps. The NN model is trained on mini-batches of 128 samples that are randomly drawn with replacement from the training set.

3.6 Model comparison

Perplexity allows us to compare all our models in similar terms, and evaluate the extent to which access to thematic roles helps the model to predict missing role fillers. For comparability, the perplexities of all models are computed only on content word probabilities (i.e., predicates and their arguments). We also report the 95% confidence interval for perplexity, which is computed according to Klakow and Peters (2002). All models are trained on exactly the same sentences of lemmatized words. Probability mass is distributed across the vocabulary of the 50,000 most frequent content words in the training corpus.

3.6.1 Models

First, we compare our model to a conventional 3-gram language model **3-gram LM**, conditioning on the previous context containing the immediately preceding context of content and function words. All n-grams are discounted with Kneser-Ney smoothing, and n-gram probability estimates are interpolated with lower order estimates. Sentence onset in all models is padded with a special sentence onset tag. The vocabulary of context words for this model consists of all words from the training corpus.

As a second model, we train a 3-gram content word model **3-gram CWM**, which is an N-gram LM that is trained only on content words.

Next, we have **RNN CWM**—an RNN LM (Mikolov et al., 2010) trained on content words only. The context size of this model is not explicitly defined and the model can potentially utilize more context words than 3-gram CWM (even from outside the sentence boundary).

Our incremental role-filler **RNN RF** is similar to RNN CWM, except for using role-specific embedding and classifier weights (slices of factored tensor). It thus has additional information about the content word roles[2].

Finally, the non-incremental role-filler **NN RF** loses the information about word order and the ability to use information outside predicate boundaries and trades it for the ability to see the future (i.e., the context includes both the preceding and the following content words and their roles).

3.6.2 Results

The results of content word perplexity evaluation are summarized in Table 1. The thematic-role informed models outperform all other models by a very large margin, cutting perplexity almost in half. The incremental model achieves a slightly lower perplexity than the non-incremental one (237.8 vs. 241.9), hinting that the content word order and out-of-predicate role-word pairs can be even more informative than a preview of upcoming role-word pairs.

The difference between normal LM and the CWM can be explained by the loss of information from function words, combined with additional sparsity in the model because content word sequences are much sparser than sequences of content and function words.

This also explains why using a neural network-based RNN CWM model improves the performance so much (perplexity drops from 834.9 to 473.2), as neural network based language models are well known for their ability to generalize well to unseen contexts by learning distributed representations of words (Bengio et al., 2003).

4 Evaluation on thematic fit ratings

In order to see whether our model accurately represents events and their typical thematic role fillers, we evaluate our model on a range of existing datasets containing human thematic fit ratings. This evaluation also allows us to compare our model to existing models that have been used on this task.

[2] A reviewer kindly points out, as a matter of historical interest, that the high-level architecture of the RNN RF model bears some resemblance to the parallel distributed processing model in McClelland et al. (1989) and St. John and McClelland (1990).

Data source	# ratings	Roles	NN RF	BL2010	GSD2015	BDK2014
Pado (agent, patient)	414	ARG0, ARG1, ARG2	0.52 (8)	**0.53** (0)	**0.53** (0)	0.41
McRae (agent, patient)	1444	ARG0, ARG1	**0.38** (20)	0.32 (70)	0.36 (70)	0.28
Ferretti (location)	274	ARGM-LOC	**0.44** (3)	0.23 (3)	0.29 (3)	-
Ferretti (instrument)	248	ARGM-MNR	**0.45** (6)	0.36 (17)	0.42 (17)	-
Greenberg (patient)	720	ARG1	**0.61** (8)	0.46 (18)	0.48 (18)	-
Pado+McRae+Ferretti	2380		**0.41** (37)	0.35 (90)	0.38 (90)	-

Table 2: Thematic fit evaluation scores, consisting of Spearman's ρ correlations between average human judgements and model output, with numbers of missing values (due to missing vocabulary entries) in brackets. The baseline scores come from the TypeDM (Baroni and Lenci, 2010) model, further developed and evaluated in Greenberg et al. (2015a,b) and the neural network *predict* model described in Baroni et al. (2014). NN RF is the non-incremental model presented in this article. Our model maps ARG2 in Pado to OTHER role. Significances were calculated using paired two-tailed significance tests for correlations (Steiger, 1980). NN RF was significantly better than both of the other models on the Greenberg and Ferretti location datasets and significantly better than BL2010 but not GSD2015 on McRae and Pado+McRae+Ferretti; differences were not statistically significant for Pado and Ferretti instruments.

4.1 Related work

State-of-the-art computational models of thematic fit quantify the similarity between a role filler of a verb and the proto-typical filler for that role for the verb based on distributional vector space models. For example, the thematic fit of *grass* as a patient for the verb *eat* would be determined by the cosine of a distributional vector representation of *grass* and a prototypical patient of *eat*. The proto-typical patient is in turn obtained from averaging representations of words that typically occur as a patient of *eat* (e.g., Erk, 2007; Baroni and Lenci, 2010; Sayeed and Demberg, 2014; Greenberg et al., 2015b). For more than one role, information from both the agent and the predicate can be used to jointly to predict a patient (e.g., Lenci, 2011).

4.2 Data

Previous studies obtained thematic fit ratings from humans by asking experimental participants to rate how common, plausible, typical, or appropriate some test role-fillers are for given verbs on a scale from 1 (least plausible) to 7 (most plausible) (McRae et al., 1998; Ferretti et al., 2001; Binder et al., 2001; Padó, 2007; Padó et al., 2009; Vandekerckhove et al., 2009; Greenberg et al., 2015a). The datasets include agent, patient, location and instrument roles. For example, in the Padó et al. (2009) dataset, the noun *sound* has a very low rating of 1.1 as the subject of *hear* and a very high rating of 6.8 as the object of *hear*. Each of the verb-role-noun triples was rated by several humans, and our evaluations are done against the average human score. The datasets differ from one another in size (as shown in Table 2), choice of verb-noun pairs, and in how exactly the question was asked of human raters.

4.3 Methods

A major difference between what the state-of-the-art models do and what our model does is that our model distributes a probability mass of one across the vocabulary, while the thematic fit models have no such overall constraint; they will assign a high number to all words that are similar to the prototypical vector, without having to distribute probability mass. Specifically, this implies that two synonymous fillers, one of which is a frequent word like *fire*, and the other of which is an infrequent word, e.g., *blaze*, will get similar ratings by the distributional similarity models, but quite different ratings by the neural network model, as the more frequent word will have higher probability. Greenberg et al. (2015a) showed that human ratings are insensitive to noun frequency. Hence, we report results that adjust for frequency effects by setting the output layer bias of the neural network model to zero. Since the output unit biases of the neural network model are independent from the inputs, they correlate strongly ($r_s = 0.74, p = 0.0$) with training corpus word frequencies after being trained. Therefore, setting the learned output layer bias vector to a zero-vector is a simple way to reduce the effect of word frequencies on the model's output probability distribution.

Role	# ratings	ρ (# NaN)
ARG0	924	0.38 (14)
ARG1	1615	0.51 (22)
ARG2	39	0.59 (0)
ARGM-MNR	248	0.45 (6)
ARGM-LOC	274	0.44 (3)
ALL	3100	0.45 (45)

Table 3: Per role thematic-fit evaluation scores in terms of Spearmans ρ correlations between average human judgements and model output.

4.4 Results

We can see that the neural network model outperforms the baselines on all the datasets except the Pado dataset. An error analysis on the role filler probabilities generated by the neural net points to the effect of level of constraint of the verb on the estimates. For a relatively non-constraining verb, the neural net model will have to distribute the probability mass across many different suitable fillers, while the semantic similarity models do not suffer from this. This implies that filler fit is not directly comparable across verbs in the NN model (only filler predictability is comparable).

Per role results are shown in Table 3. Surprisingly, the model output has the highest correlation with the averaged human judgements for the target role ARG2, despite the fact that ARG2 is mapped to OTHER along with several other roles. The model struggles the most when it comes to predicting fillers for ARG0. There is no noticeable correlation between the role-specific performance and the role occurrence frequency in the samples of our training set. This implies that parameter sharing between roles does indeed help when it comes to balancing the performance between rare and ubiquitous roles as discussed in section 3.1.

4.5 Compositionality

The above thematic role fit data sets only assess the fit between two words. Our model can however also model the interaction between different roles; see Figure 2 for an example of model predictions. We are only aware of one small dataset that can be used to systematically test the effectiveness of the compositionality for this task. The Bicknell et al. (2010) dataset contains triples like *journalist check spelling* vs. *mechanic check spelling* and *journalist check tires* vs. *mechanic check tires* together with human congruity judgments.

Model	NN RF	Lenci 2011
Accuracy 1	**0.687**	0.671
Accuracy 2	0.828	**0.844**

Table 4: Accuracies on the Bicknell evaluation task.

The goal in this task is for the model to reproduce the human judgments on the 64 sentence pairs. Lenci (2011), which we compare against in Table 4, proposed a first compositional model based on TypeDM to evaluate on this task.

We use two accuracy scores for the evaluation, which we call "Accuracy 1" and "Accuracy 2". "Accuracy 1" counts a hit iff the model assigns the composed subject-verb combination a higher score when we test a human-rated better-fitting object in contrast with when we test a worse-fitting one; in other words, a hit is achieved when *journalist check spelling* should be better than *journalist check tires*, if we give the model *journalist check* as the predicate to test against different objects. (The result from Lenci for this task was transmitted by private communication.)

"Accuracy 2" counts a hit iff, given an object, the composed subject-verb combination gives a higher score when the subject is better fitting. That is, a hit is achieved when *journalist check spelling* has a higher score than *mechanic check spelling*, setting the query to the model as *journalist check* and *mechanic check* and finding a score for *spelling* in that context. This accuracy metric is proposed and evaluated in Lenci (2011).

Evaluation shows that our model performs similarly to that of Lenci, although only limited conclusions can be drawn due to the small data set size.

5 Evaluation of event representations: sentence similarity

To show that our model learns to represent input words and their roles in a useful way that reflects the meaning and interactions between inputs, we evaluate our non-incremental model on a sentence similarity task from Grefenstette and Sadrzadeh (2015).

We assign similarity scores to sentence pairs by computing representations for each sentence by tak-

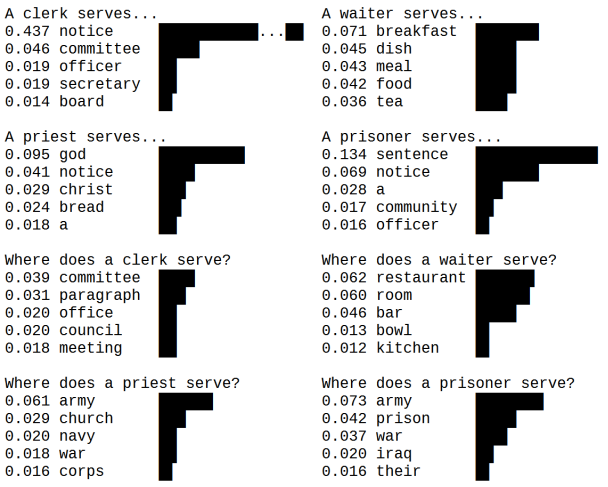


Figure 2: Examples of model predictions for the verb *serve* with different agents and target roles *patient* and *location*.

ing the hidden layer state (Equation 8) of the non-incremental model given the words in the sentence and their corresponding roles. Sentence similarity is then rated with the cosine similarity between the representations of the two sentences.

Spearman's rank correlation between the cosine similarities produced by our model and human ratings are shown in Table 5. Our model achieves much higher correlation with human ratings than the best result reported by Grefenstette and Sadrzadeh (2015), showing our model's ability to compose meaningful representations of multiple input words and their roles.

We also compare our model with another NN word representation model baseline that does not embed role information; by this comparison, we can determine the size of the improvement brought by our role-specific embeddings. The baseline sentence representations are constructed by element-wise addition of pre-trained word2vec (Mikolov et al., 2013) word embeddings[3]. Scores are again computed by using cosine similarity. The large gap between our model's and word2vec baseline's performance illustrates the importance of embedding role information in word representations.

6 Conclusions

In this paper we proposed two neural network architectures for learning proto-typical event representations. These models were trained to generate probability distributions over role fillers for a given semantic role. In our perplexity evaluation, we demonstrated that giving the model access to thematic role information substantially improved prediction performance. We also compared the performance of our model to the performance of current state-of-the-art models in predicting human thematic fit ratings and showed that our model outperforms the existing models by a large margin. Finally, we also showed that the event representations from the hidden layer of our model are highly effective in a sentence similarity task. In future work, we intend to test the potential contribution of this model when applied to larger tasks such as entailment and inference tasks as well as semantic surprisal-based prediction tasks.

[3] https://code.google.com/p/word2vec/

# ratings	NN RF	Kronecker	W2V	Humans
199	**0.34**	0.26	0.13	0.62

Table 5: Sentence similarity evaluation scores on GS2013 dataset (Grefenstette and Sadrzadeh, 2015), consisting of Spearman's ρ correlations between human judgements and model output. Kronecker is the best performing model from Grefenstette and Sadrzadeh (2015). NN RF is the non-incremental model presented in this article, and W2V is the word2vec baseline. Human performance (inter-annotator agreement) shows the upper bound.

7 Acknowledgements

This research was funded by the German Research Foundation (DFG) as part of SFB 1102: "Information Density and Linguistic Encoding" as well as the Cluster of Excellence "Multimodal Computing and Interaction" (MMCI). Also, the authors wish to thank the anonymous reviewers whose valuable ideas contributed to this paper.

References

Alumäe, T. (2013). Multi-domain neural network language model. In *INTERSPEECH*, pages 2182–2186. Citeseer.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673–721.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Bicknell, K., Elman, J. L., Hare, M., McRae, K., and Kutas, M. (2010). Effects of event knowledge in processing verbal arguments. *Journal of Memory and Language*, 63(4):489–505.

Binder, K. S., Duffy, S. A., and Rayner, K. (2001). The effects of thematic fit and discourse context on syntactic ambiguity resolution. *Journal of Memory and Language*, 44(2):297–324.

Chelba, C. and Jelinek, F. (1998). Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 225–231. Association for Computational Linguistics.

Collobert, R. and Weston, J. (2007). Fast semantic extraction using a novel neural network architecture. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 560–567, Prague, Czech Republic. Association for Computational Linguistics.

Dasigi, P. and Hovy, E. H. (2014). Modeling newswire events using neural networks for anomaly detection. In *COLING*, pages 1414–1422.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Erk, K. (2007). A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 216–223, Prague, Czech Republic. Association for Computational Linguistics.

Ferretti, T. R., McRae, K., and Hatherell, A. (2001). Integrating verbs, situation schemas, and thematic role concepts. *Journal of Memory and Language*, 44(4):516–547.

Greenberg, C., Demberg, V., and Sayeed, A. (2015a). Verb polysemy and frequency effects in thematic fit modeling. In *Proceedings of the 6th Workshop on Cognitive Modeling and Computational Linguistics*, pages 48–57, Denver, Colorado. Association for Computational Linguistics.

Greenberg, C., Sayeed, A., and Demberg, V. (2015b). Improving unsupervised vector-space thematic fit evaluation via role-filler prototype clustering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT)*.

Grefenstette, E. and Sadrzadeh, M. (2015). Concrete models and empirical evaluations for the categorical compositional distributional model of meaning. *Computational Linguistics*.

Grissom II, A. C., Boyd-Graber, J., He, H., Morgan, J., and Daumé III, H. (2014). Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, (6):164–189.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014).

Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603.

Klakow, D. and Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1):19–28.

Konstas, I. and Keller, F. (2015). Semantic role labeling improves incremental parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1191–1201, Beijing, China. Association for Computational Linguistics.

Lenci, A. (2011). Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2Nd Workshop on Cognitive Modeling and Computational Linguistics*, CMCL '11, pages 58–66, Stroudsburg, PA, USA. Association for Computational Linguistics.

McClelland, J. L., St. John, M., and Taraban, R. (1989). Sentence comprehension: A parallel distributed processing approach. *Language and cognitive processes*, 4(3-4):SI287–SI335.

McRae, K., Spivey-Knowlton, M. J., and Tanenhaus, M. K. (1998). Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38(3):283–312.

Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–1492.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Padó, U. (2007). *The integration of syntax and semantic plausibility in a wide-coverage model of human sentence processing*. PhD thesis, Saarland University.

Padó, U., Crocker, M. W., and Keller, F. (2009). A probabilistic model of semantic plausibility in sentence processing. *Cognitive Science*, 33(5):794–838.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *NATURE*, 323:9.

Sayeed, A. and Demberg, V. (2014). Combining unsupervised syntactic and semantic models of thematic fit. In *Proceedings of the first Italian Conference on Computational Linguistics (CLiC-it 2014)*.

Sayeed, A., Demberg, V., and Shkadzko, P. (2015). An exploration of semantic features in an unsupervised thematic fit evaluation framework. In *IJCoL vol. 1, n. 1 december 2015: Emerging Topics at the First Italian Conference on Computational Linguistics*, pages 25–40. Accademia University Press.

St. John, M. F. and McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46(1-2):217–257.

Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Tan, M., Zhou, W., Zheng, L., and Wang, S. (2012). A scalable distributed syntactic, semantic, and lexical language model. *Computational Linguistics*, 38(3):631–671.

Van de Cruys, T. (2014). A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35.

Vandekerckhove, B., Sandra, D., and Daelemans, W. (2009). A robust and extensible exemplar-based model of thematic fit. In *EACL 2009, 12th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Athens, Greece, March 30 - April 3, 2009*, pages 826–834.

Context-Dependent Sense Embedding*

Lin Qiu[†] and **Kewei Tu**[‡] and **Yong Yu**[†]
[†] Shanghai Jiao Tong University, Shanghai, China, {lqiu,yyu}@apex.sjtu.edu.cn
[‡] ShanghaiTech University, Shanghai, China, tukw@shanghaitech.edu.cn

Abstract

Word embedding has been widely studied and proven helpful in solving many natural language processing tasks. However, the ambiguity of natural language is always a problem on learning high quality word embeddings. A possible solution is sense embedding which trains embedding for each sense of words instead of each word. Some recent work on sense embedding uses context clustering methods to determine the senses of words, which is heuristic in nature. Other work creates a probabilistic model and performs word sense disambiguation and sense embedding iteratively. However, most of the previous work has the problems of learning sense embeddings based on imperfect word embeddings as well as ignoring the dependency between sense choices of neighboring words. In this paper, we propose a novel probabilistic model for sense embedding that is not based on problematic word embedding of polysemous words and takes into account the dependency between sense choices. Based on our model, we derive a dynamic programming inference algorithm and an Expectation-Maximization style unsupervised learning algorithm. The empirical studies show that our model outperforms the state-of-the-art model on a word sense induction task by a 13% relative gain.

1 Introduction

Distributed representation of words (aka word embedding) aims to learn continuous-valued vectors to represent words based on their context in a large corpus. They can serve as input features for algorithms of natural language processing (NLP) tasks. High quality word embeddings have been proven helpful in many NLP tasks (Collobert and Weston, 2008; Turian et al., 2010; Collobert et al., 2011; Maas et al., 2011; Chen and Manning, 2014). Recently, with the development of deep learning, many novel neural network architectures are proposed for training high quality word embeddings (Mikolov et al., 2013a; Mikolov et al., 2013b).

However, since natural language is intrinsically ambiguous, learning one vector for each word may not cover all the senses of the word. In the case of a multi-sense word, the learned vector will be around the average of all the senses of the word in the embedding space, and therefore may not be a good representation of any of the senses. A possible solution is sense embedding which trains a vector for each sense of a word. There are two key steps in training sense embeddings. First, we need to perform word sense disambiguation (WSD) or word sense induction (WSI) to determine the senses of words in the training corpus. Then, we need to train embedding vectors for word senses according to their contexts.

Early work on sense embedding (Reisinger and Mooney, 2010; Huang et al., 2012; Chen et al., 2014; Neelakantan et al., 2014; Kageback et al., 2015; Li and Jurafsky, 2015) proposes context clustering methods which determine the sense of a word by clustering aggregated embeddings of words in its context. This kind of methods is heuristic in nature and relies on external knowledge from lexicon like WordNet (Miller, 1995).

*The second author was supported by the National Natural Science Foundation of China (61503248).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 183–191,
Austin, Texas, November 1-5, 2016.

Recently, sense embedding methods based on complete probabilistic models and well-defined learning objective functions (Tian et al., 2014; Bartunov et al., 2016; Jauhar et al., 2015) become more popular. These methods regard the choice of senses of the words in a sentence as hidden variables. Learning is therefore done with expectation-maximization style algorithms, which alternate between inferring word sense choices in the training corpus and learning sense embeddings.

A common problem with these methods is that they model the sense embedding of each center word dependent on the word embeddings of its context words. As we previously explained, word embedding of a polysemous word is not a good representation and may negatively influence the quality of inference and learning. Furthermore, these methods choose the sense of each word in a sentence independently, ignoring the dependency that may exist between the sense choices of neighboring words. We argue that such dependency is important in word sense disambiguation and therefore helpful in learning sense embeddings. For example, consider the sentence "He cashed a check at the bank". Both "check" and "bank" are ambiguous here. Although the two words hint at banking related senses, the hint is not decisive (as an alternative interpretation, they may represent a check mark at a river bank). Fortunately, "cashed" is not ambiguous and it can help disambiguate "check". However, if we consider a small context window in sense embedding, then "cashed" cannot directly help disambiguate "bank". We need to rely on the dependency between the sense choices of "check" and "bank" to disambiguate "bank".

In this paper, we propose a novel probabilistic model for sense embedding that takes into account the dependency between sense choices of neighboring words. We do not learn any word embeddings in our model and hence avoid the problem with embedding polysemous words discussed above. Our model has a similar structure to a high-order hidden Markov model. It contains a sequence of observable words and latent senses and models the dependency between each word-sense pair and between neighboring senses in the sequence. The energy of neighboring senses can be modeled using existing word embedding approaches such as CBOW and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b). Given the model and a sentence, we can perform exact inference using dynamic programming and get the optimal sense sequence of the sentence. Our model can be learned from an unannotated corpus by optimizing a max-margin objective using an algorithm similar to hard-EM.

Our main contributions are the following:

1. We propose a complete probabilistic model for sense embedding. Unlike previous work, we model the dependency between sense choices of neighboring words and do not learn sense embeddings dependent on problematic word embeddings of polysemous words.

2. Based on our proposed model, we derive an exact inference algorithm and a max-margin learning algorithm which do not rely on external knowledge from any knowledge base or lexicon (except that we determine the numbers of senses of polysemous words according to an existing sense inventory).

3. The performance of our model on contextual word similarity task is competitive with previous work and we obtain a 13% relative gain compared with previous state-of-the-art methods on the word sense induction task of SemEval-2013.

The rest of this paper is organized as follows. We introduce related work in section 2. Section 3 describes our models and algorithms in detail. We present our experiments and results in section 4. In section 5, a conclusion is given.

2 Related Work

Distributed representation of words (aka word embedding) was proposed in 1986 (Hinton, 1986; Rumelhart et al., 1986). In 2003, Bengio et al. (2003) proposed a neural network architecture to train language models which produced word embeddings in the neural network. Mnih and Hinton (2007) replaced the global normalization layer of Bengio's model with a tree-structure to accelerate the training process. Collobert and Weston (2008) introduced a max-margin objective function

to replace the most computationally expensive max-likelihood objective function. Recently proposed Skip-gram model, CBOW model and GloVe model (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014) were more efficient than traditional models by introducing a log-linear layer and making it possible to train word embeddings with a large scale corpus. With the development of neural network and deep learning techniques, there have been a lot of work based on neural network models to obtain word embedding (Turian et al., 2010; Collobert et al., 2011; Maas et al., 2011; Chen and Manning, 2014). All of them have proven that word embedding is helpful in NLP tasks.

However, the models above assumed that one word has only one vector as its representation which is problematic for polysemous words. Reisinger and Mooney (2010) proposed a method for constructing multiple sense-specific representation vectors for one word by performing word sense disambiguation with context clustering. Huang et al. (2012) further extended this context clustering method and incorporated global context to learn multi-prototype representation vectors. Chen et al. (2014) extended the context clustering method and performed word sense disambiguation according to sense glosses from WordNet (Miller, 1995). Neelakantan et al. (2014) proposed an extension of the Skip-gram model combined with context clustering to estimate the number of senses for each word as well as learn sense embedding vectors. Instead of performing word sense disambiguation tasks, Kageback et al. (2015) proposed the instance-context embedding method based on context clustering to perform word sense induction tasks. Li and Jurafsky (2015) introduced a multi-sense embedding model based on the Chinese Restaurant Process and applied it to several natural language understanding tasks.

Since the context clustering based models are heuristic in nature and rely on external knowledge, recent work tends to create probabilistic models for learning sense embeddings. Tian et al. (2014) proposed a multi-prototype Skip-gram model and designed an Expectation-Maximization (EM) algorithm to do word sense disambiguation and learn sense embedding vectors iteratively. Jauhar et al. (2015) extended the EM training framework and retrofitted embedding vectors to the ontology of WordNet. Bartunov et al. (2016) proposed a non-parametric Bayesian extension of Skip-gram to automatically learn the required numbers of representations for all words and perform word sense induction tasks.

3 Context-Dependent Sense Embedding Model

We propose the context-dependent sense embedding model for training high quality sense embeddings which takes into account the dependency between sense choices of neighboring words. Unlike pervious work, we do not learn any word embeddings in our model and hence avoid the problem with embedding polysemous words discussed previously. In this section, we will introduce our model and describe our inference and learning algorithms.

3.1 Model

We begin with the notation in our model. In a sentence, let w_i be the i^{th} word of the sentence and s_i be the sense of the i^{th} word. $S(w)$ denotes the set of all the senses of word w. We assume that the sets of senses of different words do not overlap. Therefore, in this paper a word sense can be seen as a lexeme of the word (Rothe and Schutze, 2015).

Our model can be represented as a Markov network shown in Figure 1. It is similar to a high-order hidden Markov model. The model contains a sequence of observable words $(w_1, w_2, \ldots)$ and latent senses $(s_1, s_2, \ldots)$. It models the dependency between each word-sense pair and between neighboring senses in the sequence. The energy function is formulated as follows:

$$E(\mathbf{w}, \mathbf{s}) = \sum_i \Big(E_1(w_i, s_i) + E_2(s_{i-k}, \ldots, s_{i+k}) \Big) \tag{1}$$

Here $\mathbf{w} = \{w_i | 1 \leq i \leq l\}$ is the set of words in a sentence with length l and $\mathbf{s} = \{s_i | 1 \leq i \leq l\}$ is the set of senses. The function E_1 models the dependency between a word-sense pair. As we assume that the sets of senses of different words do not overlap, we can formulate E_1 as follows:

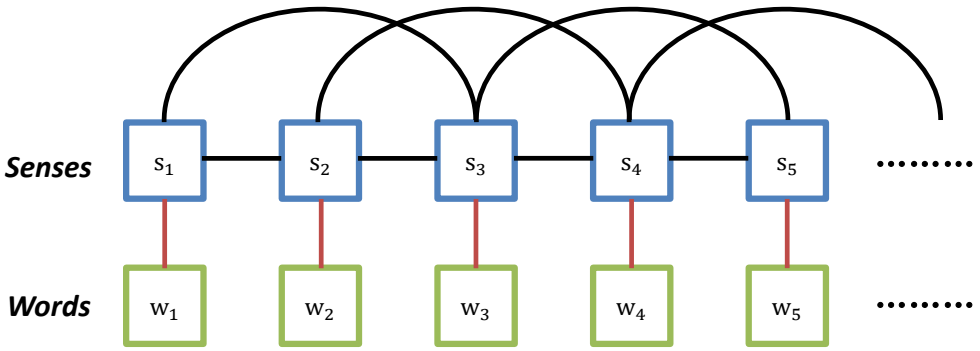


Figure 1: Context-Dependent Sense Embedding Model with window size $k = 1$

$$E_1(w_i, s_i) = \begin{cases} 0 & s_i \in S(w_i) \\ +\infty & s_i \notin S(w_i) \end{cases} \quad (2)$$

Here we assume that all the matched word-sense pairs have the same energy, but it would also be interesting to model the degrees of matching with different energy values in E_1. In Equation 1, the function E_2 models the compatibility of neighboring senses in a context window with fixed size k. Existing embedding approaches like CBOW and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b) can be used here to define E_2. The formulation using CBOW is as follows:

$$E_2(s_{i-k}, \ldots, s_{i+k}) = -\sigma\left(\sum_{i-k \le j \le i+k, j \neq i} V^T(s_j) V'(s_i)\right) \quad (3)$$

Here $V(s)$ and $V'(s)$ are the input and output embedding vectors of sense s. The function σ is an activation function and we use the sigmoid function here in our model. The formulation using Skip-gram can be defined in a similar way:

$$E_2(s_{i-k}, \ldots, s_{i+k}) = -\sum_{i-k \le j \le i+k, j \neq i} \sigma\left(V^T(s_j) V'(s_i)\right) \quad (4)$$

3.2 Inference

In this section, we introduce our inference algorithm. Given the model and a sentence **w**, we want to infer the most likely values of the hidden variables (i.e. the optimal sense sequence of the sentence) that minimize the energy function in Equation 1:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} E(\mathbf{w}, \mathbf{s}) \quad (5)$$

We use dynamic programming to do inference which is similar to the Viterbi algorithm of the hidden Markov model. Specifically, for every valid assignment $A_{i-2k}, \ldots, A_{i-1}$ of every subsequence of senses $s_{i-2k}, \ldots, s_{i-1}$, we define $m(A_{i-2k}, \ldots, A_{i-1})$ as the energy of the best sense sequence up to position $i-1$ that is consistent with the assignment $A_{i-2k}, \ldots, A_{i-1}$. We start with $m(A_1, \ldots, A_{2k}) = 0$ and then recursively compute m in a left-to-right forward process based on the update formula:

$$m(A_{i-2k+1}, \ldots, A_i) = \min_{A_{i-2k}} \Big(m(A_{i-2k}, \ldots, A_{i-1}) + E_1(w_i, A_i) + E_2(A_{i-2k}, \ldots, A_i) \Big) \quad (6)$$

Once we finish the forward process, we can retrieve the best sense sequence with a backward process. The time complexity of the algorithm is $O(n^{4k}l)$ where n is the maximal number of senses of a word. Because most words in a typical sentence have either a single sense or far less than n senses, the actual running time of the algorithm is very fast.

3.3 Learning

In this section, we introduce our unsupervised learning algorithm. In learning, we want to learn all the

input and output sense embedding vectors that optimize the following max-margin objective function:

$$\Theta^* = \arg\min_{\Theta} \sum_{\mathbf{w}\in C} \min_{\mathbf{s}} \sum_{i=1}^{\|\mathbf{w}\|} \sum_{s_{neg}\in S_{neg}(w_i)} \max\Big(1 + E_1(w_i, s_i) + E_2(s_{i-k}, \ldots, s_{i+k}) - E_2(s_{i-k}, \ldots, s_{i-1}, s_{neg}, s_{i+1}, \ldots, s_{i+k}), 0\Big) \quad (7)$$

Here Θ is the set of all the parameters including V and V' for all the senses. C is the set of training sentences. Our learning objective is similar to the negative sampling and max-margin objective proposed for word embedding (Collobert and Weston, 2008). $S_{neg}(w_i)$ denotes the set of negative samples of senses of word w_i which is defined with the following strategy. For a polysemous word w_i, $S_{neg}(w_i) = S(w_i)\backslash\{s_i\}$. For the other words with a single sense, $S_{neg}(w_i)$ is a set of randomly selected senses of a fixed size.

The objective in Equation 7 can be optimized by coordinate descent which in our case is equivalent to the hard Expectation-Maximization algorithm. In the hard E step, we run the inference algorithm using the current model parameters to get the optimal sense sequences of the training sentences. In the M step, with the sense sequences **s** of all the sentences fixed, we learn sense embedding vectors. Assume we use the CBOW model for E_2 (Equation 3), then the M-step objective function is as follows:

$$\Theta^* = \arg\min_{\Theta} \sum_{\mathbf{w}\in C} \sum_{i=1}^{\|\mathbf{w}\|} \sum_{s_{neg}\in S_{neg}(w_i)} \max\Big(1 - \sigma(\sum_{i-k\le j\le i+k, j\neq i} V(s_j)^T V'(s_i)) + \sigma(\sum_{i-k\le j\le i+k, j\neq i} V(s_j)^T V'(s_{neg})), 0\Big) \quad (8)$$

Here E_1 is omitted because the sense sequences produced from the E-step always have zero E_1 value. Similarly, if we use the Skip-gram model for E_2 (Equation 4), then the M-step objective function is:

$$\Theta^* = \arg\min_{\Theta} \sum_{\mathbf{w}\in C} \sum_{i=1}^{\|\mathbf{w}\|} \sum_{i-k\le j\le i+k, j\neq i} \sum_{s_{neg}\in S_{neg}(w_i)} \max\Big(1 - \sigma(V(s_j)^T V'(s_i)) + \sigma(V(s_j)^T V'(s_{neg})), 0\Big) \quad (9)$$

We optimize the M-step objective function using stochastic gradient descent.

We use a mini batch version of the hard EM algorithm. For each sentence in the training corpus, we run E-step to infer its sense sequence and then immediately run M-step (for 1 iteration of stochastic gradient descent) to update the model parameters based on the senses in the sentence. Therefore, the batch size of our algorithm depends on the length of each sentence.

The advantage of using mini batch is twofold. First, while our learning objective is highly non-convex (Tian et al., 2014), the randomness in mini batch hard EM may help us avoid trapping into local optima. Second, the model parameters are updated more frequently in mini batch hard EM, resulting in faster convergence.

Note that before running hard-EM, we need to determine, for each word w, the size of $S(w)$. In our experiments, we used the sense inventory provided by Coarse-Grained English All-Words Task of SemEval-2007 Task 07 (Navigli et al., 2007) to determine the number of senses for each word. The sense inventory is a coarse version of WordNet sense inventory. We do not use the WordNet sense inventory because the senses in WordNet are too fine-grained and are difficult to recognize even for human annotators (Edmonds and Kilgarriff, 2002). Since we do not link our learned senses with external sense inventories, our approach can be seen as performing WSI instead of WSD.

4 Experiments

This section presents our experiments and results. First, we describe our experimental setup including the training corpus and the model configuration.

Word	Nearest Neigbors
bank_1	banking, lender, loan
bank_2	river, canal, basin
bank_3	slope, tilted, slant
apple_1	macintosh, imac, blackberry
apple_2	peach, cherry, pie
date_1	birthdate, birth, day
date_2	appointment, meet, dinner
fox_1	cbs, abc, nbc
fox_2	wolf, deer, rabbit

Table 1: The nearest neighbors of senses of polysemous words

Then, we perform a qualitative evaluation on our model by presenting the nearest neighbors of senses of some polysemous words. Finally, we introduce two different tasks and show the experimental results on these tasks respectively.

4.1 Experimental Setup

4.1.1 Training Corpus

Our training corpus is the commonly used Wikipedia corpus. We dumped the October 2015 snapshot of the Wikipedia corpus which contains 3.6 million articles. In our experiments, we removed the infrequent words with less than 20 occurrences and the training corpus contains 1.3 billion tokens.

4.1.2 Configuration

In our experiments, we set the context window size to 5 (5 words before and after the center word). The embedding vector size is set to 300. The size of negative sample sets of single-sense words is set to 5. We trained our model using AdaGrad stochastic gradient decent (Duchi et al., 2010) with initial learning rate set to 0.025. Our configuration is similar to that of previous work.

Similar to Word2vec, we initialized our model by randomizing the sense embedding vectors. The number of senses of all the words is determined with the sense inventory provided by Coarse-Grained English All-Words Task of SemEval-2007 Task 07 (Navigli et al., 2007) as we explained in section 3.3.

4.2 Case Study

In this section, we give a qualitative evaluation of our model by presenting the nearest neighbors of the senses of some polysemous words. Table 1 shows the results of our qualitative evaluation. We list several polysemous words in the table, and for each word, some typical senses of the word are picked. The nearest neighbors of each sense are listed aside. We used the cosine distance to calculate the distance between sense embedding vectors and find the nearest neighbors.

In Table 1, we can observe that our model produces good senses for polysemous words. For example, the word "bank" can be seen to have three different sense embedding vectors. The first one means the financial institution. The second one means the sloping land beside water. The third one means the action of tipping laterally.

4.3 Word Similarity in Context

This section gives a quantitative evaluation of our model on word similarity tasks. Word similarity tasks evaluate a model's performance with the Spearman's rank correlation between the similarity scores of pairs of words given by the model and the manual labels. However, traditional word similarity tasks like Wordsim-353 (Finkelstein et al., 2001) are not suitable for evaluating sense embedding models because these datasets do not include enough ambiguous words and there is no context information for the models to infer and disambiguate the senses of the words. To overcome this issue, Huang et al. (2012) released a new dataset named Stanford's Contextual Word Similarities (SCWS) dataset. The dataset consists of 2003 pairs of words along with human labelled similarity scores and the sentences containing these words.

Given a pair of words and their contexts, we can perform inference using our model to disambiguate the questioned words. A similarity score can be calculated with the cosine distance between the two embedding vectors of the inferred senses of the questioned words. We also propose another method for calculating similarity scores. In the inference process, we compute the energy of each sense choice of the questioned word and consider the negative energy as the confidence of the sense choice. Then we calculate the cosine similarity between all pairs of senses of the questioned words and compute the average of similarity weighted by the confidence of the senses. The first method is named HardSim and the

Model	Similarity Metrics	$\rho \times 100$
Huang	AvgSim	62.8
Huang	AvgSimC	65.7
Chen	AvgSim	66.2
Chen	AvgSimC	68.9
Neelakantan	AvgSim	67.2
Neelakantan	AvgSimC	69.2
Li		**69.7**
Tian	Model_M	63.6
Tian	Model_W	65.4
Bartunov	AvgSimC	61.2
Ours + CBOW	HardSim	64.3
Ours + CBOW	SoftSim	65.6
Ours + Skip-gram	HardSim	64.9
Ours + Skip-gram	SoftSim	**66.1**

Table 2: Spearman's rank correlation results on the SCWS dataset

second method is named SoftSim.

Table 2 shows the results of our context-dependent sense embedding models on the SCWS dataset. In this table, ρ refers to the Spearman's rank correlation and a higher value of ρ indicates better performance. The baseline performances are from Huang et al. (2012), Chen et al. (2014), Neelakantan et al. (2014), Li and Jurafsky (2015), Tian et al. (2014) and Bartunov et al. (2016). Here Ours + CBOW denotes our model with a CBOW based energy function and Ours + Skip-gram denotes our model with a Skip-gram based energy function. The results above the thick line are the models based on context clustering methods and the results below the thick line are the probabilistic models including ours. The similarity metrics of context clustering based models are AvgSim and AvgSimC proposed by Reisinger and Mooney (2010). Tian et al. (2014) propose two metrics Model_M and Model_W which are similar to our HardSim and SoftSim metrics.

From Table 2, we can observe that our model outperforms the other probabilistic models and is not as good as the best context clustering based model. The context clustering based models are overall better than the probabilistic models on this task. A possible reason is that most context clustering based methods make use of more external knowledge than probabilistic models. However, note that Faruqui et al. (2016) presented several problems associated with the evaluation of word vectors on word similarity datasets and pointed out that the use of word similarity tasks for evaluation of word vectors is not sustainable. Bartunov et al. (2016) also suggest that SCWS should be of limited use for evaluating word representation models. Therefore, the results on this task shall be taken with caution. We consider that more realistic natural language processing tasks like word sense induction are better for evaluating sense embedding models.

4.4 Word Sense Induction

In this section, we present an evaluation of our model on the word sense induction (WSI) tasks. The WSI task aims to discover the different meanings for words used in sentences. Unlike a word sense disambiguation (WSD) system, a WSI system does not link the sense annotation results to an existing sense inventory. Instead, it produces its own sense inventory and links the sense annotation results to this sense inventory. Our model can be seen as a WSI system, so we can evaluate our model with WSI tasks.

We used the dataset from task 13 of SemEval-2013 as our evaluation set (Jurgens and Klapaftis, 2013). The dataset contains 4664 instances inflected from one of the 50 lemmas. Both single-sense instances and instances with a graded mixture of senses are included in the dataset. In this paper, we only consider the single sense instances. Jurgens and Klapaftis (2013) propose two fuzzy measures named Fuzzy B-Cubed (FBC) and Fuzzy Normalized Mutual Information (FNMI) for comparing fuzzy sense assignments from WSI systems. the FBC measure summarizes the performance per instance while the FNMI measure is based on sense clusters rather than instances.

Table 3 shows the results of our context-dependent sense embedding models on this dataset. Here HM is the harmonic mean of FBC and FNMI. The result of AI-KU is from Baskaya et al. (2013), MSSG is from Neelakantan et al. (2014), ICE-online and ICE-kmeans are from Kageback et al. (2015). Our models are denoted in the same way as in the previous section.

From Table 3, we can observe that our models

Model	FBC(%)	FNMI(%)	HM
AI-KU	35.1	4.5	8.0
MSSG	45.9	3.7	6.8
ICE-online	48.7	5.5	9.9
ICE-kmeans	51.1	5.9	10.6
Ours + CBOW	53.8	6.3	11.3
Ours + Skip-gram	56.9	6.7	**12.0**

Table 3: Results of single-sense instances on task 13 of SemEval-2013

outperform the previous state-of-the-art models and achieve a 13% relative gain. It shows that our models can beat context clustering based models on realistic natural language processing tasks.

5 Conclusion

In this paper we propose a novel probabilistic model for learning sense embeddings. Unlike previous work, we do not learn sense embeddings dependent on word embeddings and hence avoid the problem with inaccurate embeddings of polysemous words. Furthermore, we model the dependency between sense choices of neighboring words which can help us disambiguate multiple ambiguous words in a sentence. Based on our model, we derive a dynamic programming inference algorithm and an EM-style unsupervised learning algorithm which do not rely on external knowledge from any knowledge base or lexicon except that we determine the number of senses of polysemous words according to an existing sense inventory. We evaluate our model both qualitatively by case studying and quantitatively with the word similarity task and the word sense induction task. Our model is competitive with previous work on the word similarity task. On the word sense induction task, our model outperforms the state-of-the-art model and achieves a 13% relative gain.

For the future work, we plan to try learning our model with soft EM. Besides, we plan to use shared senses instead of lexemes in our model to improve the generality of our model. Also, we will study unsupervised methods to link the learned senses to existing inventories and to automatically determine the numbers of senses. Finally, we plan to evaluate our model with more NLP tasks.

References

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 300–306.

Yoshua Bengio, Holger Schwenk, Jean Sbastien Sencal, Frderic Morin, and Jean Luc Gauvain. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(6):1137–1155.

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

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):257–269.

Philip Edmonds and Adam Kilgarriff. 2002. Introduction to the special issue on evaluating word sense disambiguation systems. *Natural Language Engineering*, 8(4):279–291.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proceedings of international conference on World Wide Web*, pages 406–414.

G. E. Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

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

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299.

Mikael Kageback, Fredrik Johansson, Richard Johansson, and Devdatt Dubhashi. 2015. Neural context embeddings for automatic discovery of word senses. In *Proceedings of NAACL-HLT*, pages 25–32.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *EMNLP*, pages 1722–1732. Association for Computational Linguistics.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

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

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 641–648.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: coarse-grained english all-words task. In *International Workshop on Semantic Evaluations*, pages 30–35.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, pages 1059–1069. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schutze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1793–1803. Association for Computational Linguistics.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representation by back-propagating errors. *Nature*, 323(6088):533–536.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Jointly Embedding Knowledge Graphs and Logical Rules

Shu Guo[†‡], **Quan Wang**[†‡*], **Lihong Wang**[§], **Bin Wang**[†‡], **Li Guo**[†‡]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[‡]University of Chinese Academy of Sciences, Beijing 100049, China
{guoshu,wangquan,wangbin,guoli}@iie.ac.cn
[§]National Computer Network Emergency Response Technical Team
Coordination Center of China, Beijing 100029, China
wlh@isc.org.cn

Abstract

Embedding knowledge graphs into continuous vector spaces has recently attracted increasing interest. Most existing methods perform the embedding task using only fact triples. Logical rules, although containing rich background information, have not been well studied in this task. This paper proposes a novel method of jointly embedding knowledge graphs and logical rules. The key idea is to represent and model triples and rules in a unified framework. Specifically, triples are represented as atomic formulae and modeled by the translation assumption, while rules represented as complex formulae and modeled by t-norm fuzzy logics. Embedding then amounts to minimizing a global loss over both atomic and complex formulae. In this manner, we learn embeddings compatible not only with triples but also with rules, which will certainly be more predictive for knowledge acquisition and inference. We evaluate our method with link prediction and triple classification tasks. Experimental results show that joint embedding brings significant and consistent improvements over state-of-the-art methods. Particularly, it enhances the prediction of new facts which cannot even be directly inferred by pure logical inference, demonstrating the capability of our method to learn more predictive embeddings.

1 Introduction

Knowledge graphs (KGs) provide rich structured information and have become extremely useful resources for many NLP related applications like word sense disambiguation (Wasserman-Pritsker et al., 2015) and information extraction (Hoffmann et al., 2011). A typical KG represents knowledge as multi-relational data, stored in triples of the form (*head entity*, *relation*, *tail entity*), e.g., (`Paris`, `Capital-Of`, `France`). Although powerful in representing structured data, the symbolic nature of such triples makes KGs, especially large-scale KGs, hard to manipulate.

Recently, a promising approach, namely knowledge graph embedding, has been proposed and successfully applied to various KGs (Nickel et al., 2012; Socher et al., 2013; Bordes et al., 2014). The key idea is to embed components of a KG including entities and relations into a continuous vector space, so as to simplify the manipulation while preserving the inherent structure of the KG. The embeddings contain rich semantic information about entities and relations, and can significantly enhance knowledge acquisition and inference (Weston et al., 2013).

Most existing methods perform the embedding task based solely on fact triples (Bordes et al., 2013; Wang et al., 2014; Nickel et al., 2016). The only requirement is that the learned embeddings should be compatible with those facts. While logical rules contain rich background information and are extremely useful for knowledge acquisition and inference (Jiang et al., 2012; Pujara et al., 2013), they have not been well studied in this task. Wang et al. (2015) and Wei et al. (2015) tried to leverage both embedding methods and logical rules for KG completion. In their work, however, rules are modeled separately from embedding methods, serving as post-processing steps, and thus will not help to obtain

*Corresponding author: Quan Wang.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 192–202,
Austin, Texas, November 1-5, 2016.

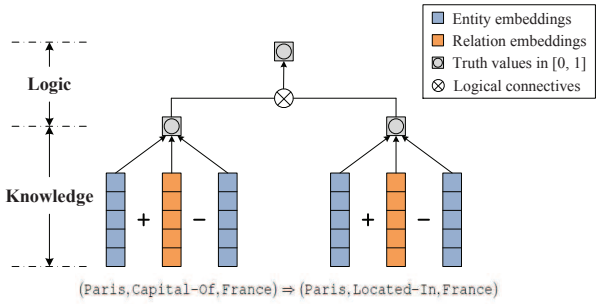


Figure 1: Simple illustration of KALE.

better embeddings. Rocktäschel et al. (2015) recently proposed a joint model which injects first-order logic into embeddings. But it focuses on the relation extraction task, and creates vector embeddings for entity pairs rather than individual entities. Since entities do not have their own embeddings, relations between unpaired entities cannot be effectively discovered (Chang et al., 2014).

In this paper we introduce KALE, a new approach that learns entity and relation Embeddings by jointly modeling Knowledge And Logic. Knowledge triples are taken as atoms and modeled by the translation assumption, i.e., relations act as translations between head and tail entities (Bordes et al., 2013). A triple (e_i, r_k, e_j) is scored by $\|\mathbf{e}_i + \mathbf{r}_k - \mathbf{e}_j\|_1$, where $\mathbf{e}_i$, $\mathbf{r}_k$, and $\mathbf{e}_j$ are the vector embeddings for entities and relations. The score is then mapped to the unit interval $[0, 1]$ to indicate the truth value of that triple. Logical rules are taken as complex formulae constructed by combining atoms with logical connectives (e.g., $\wedge$ and $\Rightarrow$), and modeled by t-norm fuzzy logics (Hájek, 1998). The truth value of a rule is a composition of the truth values of the constituent atoms, defined by specific logical connectives. In this way, KALE represents triples and rules in a unified framework, as atomic and complex formulae respectively. Figure 1 gives a simple illustration of the framework. After unifying triples and rules, KALE minimizes a global loss involving both of them to obtain entity and relation embeddings. The learned embeddings are therefore compatible not only with triples but also with rules, which will definitely be more predictive for knowledge acquisition and inference.

The main contributions of this paper are summarized as follows. (i) We devise a unified framework that jointly models triples and rules to obtain more predictive entity and relation embeddings. The new framework KALE is general enough to handle any type of rules that can be represented as first-order logic formulae. (ii) We evaluate KALE with link prediction and triple classification tasks on WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008). Experimental results show significant and consistent improvements over state-of-the-art methods. Particularly, joint embedding enhances the prediction of new facts which cannot even be directly inferred by pure logical inference, demonstrating the capability of KALE to learn more predictive embeddings.

2 Related Work

Recent years have seen rapid growth in KG embedding methods. Given a KG, such methods aim to encode its entities and relations into a continuous vector space, by using neural network architectures (Socher et al., 2013; Bordes et al., 2013; Bordes et al., 2014), matrix/tensor factorization techniques (Nickel et al., 2011; Riedel et al., 2013; Chang et al., 2014), or Bayesian clustering strategies (Kemp et al., 2006; Xu et al., 2006; Sutskever et al., 2009). Among these methods, TransE (Bordes et al., 2013), which models relations as translating operations, achieves a good trade-off between prediction accuracy and computational efficiency. Various extensions like TransH (Wang et al., 2014) and TransR (Lin et al., 2015b) are later proposed to further enhance the prediction accuracy of TransE. Most existing methods perform the embedding task based solely on triples contained in a KG. Some recent work tries to further incorporate other types of information available, e.g., relation paths (Neelakantan et al., 2015; Lin et al., 2015a; Luo et al., 2015), relation type-constraints (Krompaßet al., 2015), entity types (Guo et al., 2015), and entity descriptions (Zhong et al., 2015), to learn better embeddings.

Logical rules have been widely studied in knowledge acquisition and inference, usually on the basis of Markov logic networks (Richardson and Domingos, 2006; Bröcheler et al., 2010; Pujara et al., 2013; Beltagy and Mooney, 2014). Recently, there has been growing interest in combining logical rules and embedding models. Wang et al. (2015) and Wei et al. (2015) tried to utilize rules to refine predictions

made by embedding models, via integer linear programming or Markov logic networks. In their work, however, rules are modeled separately from embedding models, and will not help obtain better embeddings. Rocktäschel et al. (2015) proposed a joint model that injects first-order logic into embeddings. But their work focuses on relation extraction, creating vector embeddings for entity pairs, and hence fails to discover relations between unpaired entities. This paper, in contrast, aims at learning more predictive embeddings by jointly modeling knowledge and logic. Since each entity has its own embedding, our approach can successfully make predictions between unpaired entities, providing greater flexibility for knowledge acquisition and inference.

3 Jointly Embedding Knowledge and Logic

We first describe the formulation of joint embedding. We are given a KG containing a set of triples $\mathcal{K} = \{(e_i, r_k, e_j)\}$, with each triple composed of two entities $e_i, e_j \in \mathcal{E}$ and their relation $r_k \in \mathcal{R}$. Here $\mathcal{E}$ is the entity vocabulary and $\mathcal{R}$ the relation set. Besides the triples, we are given a set of logical rules $\mathcal{L}$, either specified manually or extracted automatically. A logical rule is encoded, for example, in the form of $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, stating that any two entities linked by relation r_s should also be linked by relation r_t. Entities and relations are associated with vector embeddings, denoted by $\mathbf{e}, \mathbf{r} \in \mathbb{R}^d$, representing their latent semantics. The proposed method, KALE, aims to learn these embeddings by jointly modeling knowledge triples $\mathcal{K}$ and logical rules $\mathcal{L}$.

3.1 Overview

To enable joint embedding, a key ingredient of KALE is to unify triples and rules, in terms of first-order logic (Rocktäschel et al., 2014; Rocktäschel et al., 2015). A triple (e_i, r_k, e_j) is taken as a ground atom which applies a relation r_k to a pair of entities e_i and e_j. Given a logical rule, it is first instantiated with concrete entities in the vocabulary $\mathcal{E}$, resulting in a set of ground rules. For example, a universally quantified rule $\forall x, y : (x, \texttt{Capital-Of}, y) \Rightarrow (x, \texttt{Located-In}, y)$ might be instantiated with the concrete entities of `Paris` and `France`, giving the ground rule $(\texttt{Paris}, \texttt{Capital-Of}, \texttt{France}) \Rightarrow (\texttt{Paris}, \texttt{Located-In}, \texttt{France})$.[1] A ground rule can then be interpreted as a complex formula, constructed by combining ground atoms with logical connectives (e.g. $\wedge$ and $\Rightarrow$).

Let $\mathcal{F}$ denote the set of training formulae, both atomic (triples) and complex (ground rules). KALE further employs a truth function $I : \mathcal{F} \rightarrow [0, 1]$ to assign a soft truth value to each formula, indicating how likely a triple holds or to what degree a ground rule is satisfied. The truth value of a triple is determined by the corresponding entity and relation embeddings. The truth value of a ground rule is determined by the truth values of the constituent triples, via specific logical connectives. In this way, KALE models triples and rules in a unified framework. See Figure 1 for an overview. Finally, KALE minimizes a global loss over the training formulae $\mathcal{F}$ to learn entity and relation embeddings compatible with both triples and rules. In what follows, we describe the key components of KALE, including triple modeling, rule modeling, and joint learning.

3.2 Triple Modeling

To model triples we follow TransE (Bordes et al., 2013), as it is simple and efficient while achieving state-of-the-art predictive performance. Specifically, given a triple (e_i, r_k, e_j), we model the relation embedding $\mathbf{r}_k$ as a translation between the entity embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$, i.e., we want $\mathbf{e}_i + \mathbf{r}_k \approx \mathbf{e}_j$ when the triple holds. The intuition here originates from linguistic regularities such as $\texttt{France} - \texttt{Paris} = \texttt{Germany} - \texttt{Berlin}$ (Mikolov et al., 2013). In relational data, such analogy holds because of the certain relation `Capital-Of`, through which we will get $\texttt{Paris} + \texttt{Capital-Of} = \texttt{France}$ and $\texttt{Berlin} + \texttt{Capital-Of} = \texttt{Germany}$. Then, we score each triple on the basis of $\|\mathbf{e}_i + \mathbf{r}_k - \mathbf{e}_j\|_1$, and define its soft truth value as

$$I(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \|\mathbf{e}_i + \mathbf{r}_k - \mathbf{e}_j\|_1, \quad (1)$$

where d is the dimension of the embedding space. It is easy to see that $I(e_i, r_k, e_j) \in [0, 1]$ with the constraints $\|\mathbf{e}_i\|_2 \leq 1$, $\|\mathbf{e}_j\|_2 \leq 1$, and $\|\mathbf{r}_k\|_2 \leq$

[1] Our approach actually takes as input rules represented in first-order logic, i.e., those with quantifiers such as $\forall$. But it could be hard to deal with quantifiers, so we use ground rules, i.e., propositional statements during learning.

1.[2] $I(e_i, r_k, e_j)$ is expected to be large if the triple holds, and small otherwise.

3.3 Rule Modeling

To model rules we use t-norm fuzzy logics (Hájek, 1998), which define the truth value of a complex formula as a composition of the truth values of its constituents, through specific t-norm based logical connectives. We follow Rocktäschel et al. (2015) and use the product t-norm. The compositions associated with logical conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$) are defined as follow:

$$\begin{aligned} I(f_1 \wedge f_2) &= I(f_1) \cdot I(f_2), \\ I(f_1 \vee f_2) &= I(f_1) + I(f_2) - I(f_1) \cdot I(f_2), \\ I(\neg f_1) &= 1 - I(f_1), \end{aligned}$$

where f_1 and f_2 are two constituent formulae, either atomic or complex. Given these compositions, the truth value of any complex formula can be calculated recursively, e.g.,

$$\begin{aligned} I(\neg f_1 \wedge f_2) &= I(f_2) - I(f_1) \cdot I(f_2), \\ I(f_1 \Rightarrow f_2) &= I(f_1) \cdot I(f_2) - I(f_1) + 1. \end{aligned}$$

This paper considers two types of rules. The first type is $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$. Given a ground rule $f \triangleq (e_m, r_s, e_n) \Rightarrow (e_m, r_t, e_n)$, the truth value is calculated as:

$$\begin{aligned} I(f) = & I(e_m, r_s, e_n) \cdot I(e_m, r_t, e_n) \\ & - I(e_m, r_s, e_n) + 1, \end{aligned} \tag{2}$$

where $I(\cdot,\cdot,\cdot)$ is the truth value of a constituent triple, defined by Eq. (1). The second type is $\forall x, y, z : (x, r_{s_1}, y) \wedge (y, r_{s_2}, z) \Rightarrow (x, r_t, z)$. Given a ground rule $f \triangleq (e_\ell, r_{s_1}, e_m) \wedge (e_m, r_{s_2}, e_n) \Rightarrow (e_\ell, r_t, e_n)$, the truth value is:

$$\begin{aligned} I(f) = & I(e_\ell, r_{s_1}, e_m) \cdot I(e_m, r_{s_2}, e_n) \cdot I(e_\ell, r_t, e_n) \\ & - I(e_\ell, r_{s_1}, e_m) \cdot I(e_m, r_{s_2}, e_n) + 1. \end{aligned} \tag{3}$$

The larger the truth values are, the better the ground rules are satisfied. It is easy to see that besides these two types of rules, the KALE framework is general enough to handle any rules that can be represented as first-order logic formulae. The investigation of other types of rules will be left for future work.

[2]Note that $0 \leq \|\mathbf{e}_i + \mathbf{r}_k - \mathbf{e}_j\|_1 \leq \|\mathbf{e}_i\|_1 + \|\mathbf{r}_k\|_1 + \|\mathbf{e}_j\|_1 \leq 3\sqrt{d}$, where the last inequality holds because $\|\mathbf{x}\|_1 = \sum_i |x_i| \leq \sqrt{d \sum_i x_i^2} = \sqrt{d}\,\|\mathbf{x}\|_2$ for any $\mathbf{x} \in \mathbb{R}^d$, according to the Cauchy-Schwarz inequality.

3.4 Joint Learning

After unifying triples and rules as atomic and complex formulae, we minimize a global loss over this general representation to learn entity and relation embeddings. We first construct a training set $\mathcal{F}$ containing all positive formulae, including (i) observed triples, and (ii) ground rules in which at least one constituent triple is observed. Then we minimize a margin-based ranking loss, enforcing positive formulae to have larger truth values than negative ones:

$$\begin{aligned} \min_{\{\mathbf{e}\},\{\mathbf{r}\}} & \sum_{f^+ \in \mathcal{F}} \sum_{f^- \in \mathcal{N}_{f^+}} \left[\gamma - I(f^+) + I(f^-)\right]_+, \\ \text{s.t.} \quad & \|\mathbf{e}\|_2 \leq 1, \forall e \in \mathcal{E}; \quad \|\mathbf{r}\|_2 \leq 1, \forall r \in \mathcal{R}. \end{aligned} \tag{4}$$

Here $f^+ \in \mathcal{F}$ is a positive formula, $f^- \in \mathcal{N}_{f^+}$ a negative one constructed for f^+, γ a margin separating positive and negative formulae, and $[x]_+ \triangleq \max\{0, x\}$. If $f^+ \triangleq (e_i, r_k, e_j)$ is a triple, we construct f^- by replacing either e_i or e_j with a random entity $e \in \mathcal{E}$, and calculate its truth value according to Eq. (1). For example, we might generate a negative instance $(\texttt{Paris}, \texttt{Capital-Of}, \texttt{Germany})$ for the triple $(\texttt{Paris}, \texttt{Capital-Of}, \texttt{France})$. If $f^+ \triangleq (e_m, r_s, e_n) \Rightarrow (e_m, r_t, e_n)$ or $(e_\ell, r_{s_1}, e_m) \wedge (e_m, r_{s_2}, e_n) \Rightarrow (e_\ell, r_t, e_n)$ is a ground rule, we construct f^- by replacing r_t in the consequent with a random relation $r \in \mathcal{R}$, and calculate its truth value according to Eq. (2) or Eq. (3). For example, given a ground rule $(\texttt{Paris}, \texttt{Capital-Of}, \texttt{France}) \Rightarrow (\texttt{Paris}, \texttt{Located-In}, \texttt{France})$, a possible negative instance $(\texttt{Paris}, \texttt{Capital-Of}, \texttt{France}) \Rightarrow (\texttt{Paris}, \texttt{Has-Spouse}, \texttt{France})$ could be generated. We believe that most instances (both triples and ground rules) generated in this way are truly negative. Stochastic gradient descent in mini-batch mode is used to carry out the minimization. To satisfy the ℓ_2-constraints, $\mathbf{e}$ and $\mathbf{r}$ are projected to the unit ℓ_2-ball before each mini-batch. Embeddings learned in this way are required to be compatible with not only triples but also rules.

3.5 Discussions

Complexity. We compare KALE with several state-of-the-art embedding methods in space complexity and time complexity (per iteration) during learning. Table 1 shows the results, where d is the dimension

Method	Complexity (Space/Time)	
SE (Bordes et al., 2011)	$n_e d+2n_r d^2$	$O(n_t d^2)$
LFM (Jenatton et al., 2012)	$n_e d+n_r d^2$	$O(n_t d^2)$
TransE (Bordes et al., 2013)	$n_e d+n_r d$	$O(n_t d)$
TransH (Wang et al., 2014)	$n_e d+2n_r d$	$O(n_t d)$
TransR (Lin et al., 2015b)	$n_e d+n_r(d^2+d)$	$O(n_t d^2)$
KALE (this paper)	$n_e d+n_r d$	$O(n_t d+n_g d)$

Table 1: Complexity of different embedding methods.

of the embedding space, and $n_e/n_r/n_t/n_g$ is the number of entities/relations/triples/ground rules. The results indicate that incorporating additional rules will not significantly increase the space or time complexity of KALE, keeping the model complexity almost the same as that of TransE (optimal among the methods listed in the table). But please note that KALE needs to ground universally quantified rules before learning, which further requires $O(n_u n_t/n_r)$ in time complexity. Here, n_u is the number of universally quantified rules, and n_t/n_r is the averaged number of observed triples per relation. During grounding, we select those ground rules with at least one triple observed. Grounding is required only once before learning, and is not included during the iterations.

Extensions. Actually, our approach is quite general. (i) Besides TransE, a variety of embedding methods, e.g., those listed in Table 1, can be used for triple modeling (Section 3.2), as long as we further define a mapping $f : \mathbb{R} \to [0, 1]$ to map original scores to soft truth values. (ii) Besides the two types of rules introduced in Section 3.3, other types of rules can also be handled as long as they can be represented as first-order logic formulae. (iii) Besides the product t-norm, other types of t-norm based fuzzy logics can be used for rule modeling (Section 3.3), e.g., the Łukasiewicz t-norm used in probabilistic soft logic (Bröcheler et al., 2010) and the minimum t-norm used in fuzzy description logic (Stoilos et al., 2007). (iv) Besides the pairwise ranking loss, other types of loss functions can be designed for joint learning (Section 3.4), e.g., the pointwise squared loss or the logarithmic loss (Rocktäschel et al., 2014; Rocktäschel et al., 2015).

4 Experiments

We empirically evaluate KALE with two tasks: (i) link prediction and (ii) triple classification.

Dataset	# Ent	# Rel	# Train/Valid/Test-I/Test-II				# Rule
FB122	9,738	122	91,638	9,595	5,057	6,186	78,488
WN18	40,943	18	141,442	5,000	1,394	3,606	119,222

Table 3: Statistics of datasets.

4.1 Experimental Setup

Datasets. We use two datasets: WN18 and FB122. WN18 is a subgraph of WordNet containing 18 relations. FB122 is composed of 122 Freebase relations regarding the topics of "people", "location", and "sports", extracted from FB15K. Both WN18 and FB15K are released by Bordes et al. (2013)[3]. Triples on each dataset are split into training/validation/test sets, used for model training, parameter tuning, and evaluation respectively. For WN18 we use the original data split, and for FB122 we extract triples associated with the 122 relations from the training, validation, and test sets of FB15K.

We further create logical rules for each dataset, in the form of $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$ or $\forall x, y, z : (x, r_{s_1}, y) \wedge (y, r_{s_2}, z) \Rightarrow (x, r_t, z)$. To do so, we first run TransE to get entity and relation embeddings, and calculate the truth value for each of such rules according to Eq. (2) or Eq. (3). Then we rank all such rules by their truth values and manually filter those ranked at the top. We finally create 47 rules on FB122, and 14 on WN18 (see Table 2 for examples). The rules are then instantiated with concrete entities (grounding). Ground rules in which at least one constituent triple is observed in the *training* set are used in joint learning.

Note that some of the test triples can be inferred by directly applying these rules on the training set (pure logical inference). On each dataset, we further split the test set into two parts, test-I and test-II. The former contains triples that *cannot* be directly inferred by pure logical inference, and the latter the remaining test triples. Table 3 gives some statistics of the datasets, including the number of entities, relations, triples in training/validation/test-I/test-II set, and ground rules.

Comparison settings. As baselines we take the embedding techniques of TransE, TransH, and TransR. TransE models relation embeddings as translation operations between entity embeddings. TransH

[3] https://everest.hds.utc.fr/doku.php?id=en:smemlj12

$\forall x, y$: /sports/athlete/team$(x, y) \Rightarrow$ /sports/sports_team/player(y, x)
$\forall x, y$: /location/country/capital$(x, y) \Rightarrow$ /location/location/contains(x, y)
$\forall x, y, z$: /people/person/nationality$(x, y) \wedge$ /location/country/official_language$(y, z) \Rightarrow$ /people/person/languages(x, z)
$\forall x, y, z$: /country/administrative_divisions$(x, y) \wedge$ /administrative_division/capital$(y, z) \Rightarrow$ /country/second_level_divisions(x, z)
$\forall x, y$: _hypernym$(x, y) \Rightarrow$ _hyponym(y, x)
$\forall x, y$: _instance_hypernym$(x, y) \Rightarrow$ _instance_hyponym(y, x)
$\forall x, y$: _synset_domain_topic_of$(x, y) \Rightarrow$ _member_of_domain_topic(y, x)

Table 2: Examples of rules created.

and TransR are extensions of TransE. They further allow entities to have distinct embeddings when involved in different relations, by introducing relation-specific hyperplanes and projection matrices respectively. All the three methods have been demonstrated to perform well on WordNet and Freebase data.

We further test our approach in three different scenarios. (i) KALE-Trip uses triples alone to perform the embedding task, i.e., only the training triples are included in the optimization Eq. (4). It is a linearly transformed version of TransE. The only difference is that relation embeddings are normalized in KALE-Trip, but not in TransE. (ii) KALE-Pre first repeats pure logical inference on the training set and adds inferred triples as additional training data, until no further triples can be inferred. Both original and inferred triples are then included in the optimization. For example, given a logical rule $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, a new triple (e_i, r_t, e_j) can be inferred if (e_i, r_s, e_j) is observed in the training set, and both triples will be used as training instances for embedding. (iii) KALE-Joint is the joint learning scenario, which considers both training triples and ground rules in the optimization. In the aforementioned example, training triple (e_i, r_s, e_j) and ground rule $(e_i, r_s, e_j) \Rightarrow (e_i, r_t, e_j)$ will be used in the training process of KALE-Joint, *without* explicitly incorporating triple (e_i, r_t, e_j). Among the methods, TransE/TransH/TransR and KALE-Trip use only triples, while KALE-Pre/KALE-Joint further incorporates rules, before or during embedding.

Implementation details. We use the code provided by Bordes et al. (2013) for TransE[4], and the code provided by Lin et al. (2015b) for TransH and TransR[5]. KALE is implemented in Java. Note that Lin et al. (2015b) initialized TransR with the results of TransE. However, to ensure fair comparison, we randomly initialize all the methods in our experiments. For all the methods, we create 100 mini-batches on each dataset, and tune the embedding dimension d in $\{20, 50, 100\}$. For TransE, TransH, and TransR which score a triple by a distance in $\mathbb{R}^+$, we tune the learning rate η in $\{0.001, 0.01, 0.1\}$, and the margin γ in $\{1, 2, 3, 4\}$. For KALE which scores a triple (as well as a ground rule) by a soft truth value in the unit interval $[0, 1]$, we set the learning rate η in $\{0.01, 0.02, 0.05, 0.1\}$, and the margin γ in $\{0.1, 0.12, 0.15, 0.2\}$. KALE allows triples and rules to have different weights, with the former fixed to 1, and the latter (denoted by λ) selected in $\{0.001, 0.01, 0.1, 1\}$.

4.2 Link Prediction

This task is to complete a triple (e_i, r_k, e_j) with e_i or e_j missing, i.e., predict e_i given (r_k, e_j) or predict e_j given (e_i, r_k).

Evaluation protocol. We follow the same evaluation protocol used in TransE (Bordes et al., 2013). For each test triple (e_i, r_k, e_j), we replace the head entity e_i by every entity e'_i in the dictionary, and calculate the truth value (or distance) for the corrupted triple (e'_i, r_k, e_j). Ranking the truth values in descending order (or the distances in ascending order), we get the rank of the correct entity e_i. Similarly, we can get another rank by corrupting the tail entity e_j. Aggregated over all the test triples, we report three metrics: (i) the mean reciprocal rank (MRR), (ii) the median of the ranks (MED), and (iii) the proportion of ranks no larger than n (HITS@N). We do not report the averaged rank (i.e., the "Mean Rank" metric used by Bordes et al. (2013)), since it is usually sensitive to outliers (Nickel et al., 2016).

Note that a corrupted triple may exist in KGs, which should also be taken as a valid triple. Consider a test triple (`Paris`, `Located-In`, `France`)

[4]https://github.com/glorotxa/SME

[5]https://github.com/mrlyk423/relation_extraction

		Test-I					Test-II					Test-ALL				
		MRR	MED	HITS@N (%)			MRR	MED	HITS@N (%)			MRR	MED	HITS@N (%)		
				3	5	10			3	5	10			3	5	10
FB122	TransE	0.220	29.0	25.7	32.4	40.6	0.296	5.0	40.0	50.8	57.8	0.262	10.0	33.6	42.5	50.0
	TransH	0.218	29.0	25.0	31.3	39.2	0.297	6.0	37.5	48.5	56.3	0.249	12.0	31.9	40.7	48.6
	TransR	0.219	31.0	24.7	30.8	38.9	0.273	9.0	32.4	42.8	51.6	0.261	15.0	28.9	37.4	45.9
	KALE-Trip	0.201	25.0	23.9	31.6	40.1	0.309	5.0	40.9	51.3	58.0	0.261	11.0	33.3	42.4	50.0
	KALE-Pre	0.203	25.0	24.1	31.7	40.2	**0.368**	**4.0**	**47.3**	**55.4**	**61.4**	**0.294**	**9.0**	**36.9**	**44.8**	**51.9**
	KALE-Joint	**0.229**	**21.0**	**26.3**	**33.8**	**42.2**	0.357	**4.0**	44.0	53.0	59.3	0.299	**9.0**	36.1	44.3	51.6
WN18	TransE	0.248	**4.0**	40.9	60.6	77.0	0.363	3.0	59.4	70.8	81.4	0.331	3.0	54.3	67.9	80.2
	TransH	0.242	**4.0**	39.2	60.1	75.9	0.482	**2.0**	63.5	70.8	79.3	0.415	3.0	56.7	67.8	78.3
	TransR	0.240	**4.0**	40.1	57.7	71.6	0.449	3.0	55.7	64.5	74.3	0.391	3.0	51.3	62.6	73.5
	KALE-Trip	0.250	**4.0**	40.6	62.3	78.1	0.393	**2.0**	61.9	71.2	80.6	0.353	3.0	56.0	68.7	79.9
	KALE-Pre	0.248	**4.0**	40.4	61.5	78.2	0.451	3.0	**69.6**	**77.5**	**85.3**	0.395	3.0	**61.4**	**73.0**	**83.3**
	KALE-Joint	**0.260**	**4.0**	**43.6**	**64.1**	**79.2**	**0.563**	**2.0**	67.6	73.8	81.0	**0.478**	**2.0**	60.9	71.1	80.5

Table 4: Link prediction results on the test-I, test-II, and test-all sets of FB122 and WN18 (raw setting).

		Test-I					Test-II					Test-ALL				
		MRR	MED	HITS@N (%)			MRR	MED	HITS@N (%)			MRR	MED	HITS@N (%)		
				3	5	10			3	5	10			3	5	10
FB122	TransE	0.296	13.0	36.0	41.5	48.1	0.630	2.0	77.5	82.8	88.4	0.480	**2.0**	58.9	64.2	70.2
	TransH	0.280	15.0	33.6	39.1	46.4	0.606	2.0	70.1	75.4	82.0	0.460	3.0	53.7	59.1	66.0
	TransR	0.283	16.0	33.4	39.2	46.0	0.499	2.0	57.0	63.2	70.1	0.401	5.0	46.4	52.4	59.3
	KALE-Trip	0.299	10.0	36.6	42.9	50.2	0.650	2.0	79.0	83.4	88.7	0.492	**2.0**	59.9	65.2	71.4
	KALE-Pre	0.291	11.0	35.8	41.9	49.8	**0.713**	**1.0**	**82.9**	**86.1**	**89.9**	**0.523**	**2.0**	**61.7**	66.2	71.8
	KALE-Joint	**0.325**	**9.0**	**38.4**	**44.7**	**52.2**	0.684	**1.0**	79.7	84.1	89.6	**0.523**	**2.0**	61.2	**66.4**	**72.8**
WN18	TransE	0.306	**3.0**	57.4	72.3	80.1	0.511	2.0	87.5	95.6	98.7	0.453	**2.0**	79.1	89.1	93.6
	TransH	0.318	**3.0**	61.7	72.4	78.2	0.653	2.0	87.1	91.4	94.6	0.560	**2.0**	80.0	86.1	90.0
	TransR	0.299	**3.0**	56.1	66.7	74.5	0.597	2.0	75.0	81.7	88.0	0.514	**2.0**	69.7	77.5	84.3
	KALE-Trip	0.322	**3.0**	61.0	73.9	80.8	0.555	2.0	90.6	96.3	98.8	0.490	**2.0**	82.3	90.1	93.8
	KALE-Pre	0.322	**3.0**	60.6	74.5	81.1	0.612	2.0	**96.4**	**98.6**	**99.6**	0.532	**2.0**	**86.4**	**91.9**	**94.4**
	KALE-Joint	**0.338**	**3.0**	**65.5**	**76.3**	**82.1**	**0.787**	**1.0**	93.3	95.4	97.2	**0.662**	**2.0**	85.5	90.1	93.0

Table 5: Link prediction results on the test-I, test-II, and test-all sets of FB122 and WN18 (filtered setting).

and a possible corruption (`Lyon`, `Located-In`, `France`). Both triples are valid. In this case, ranking `Lyon` before the correct answer `Paris` should not be counted as an error. To avoid such phenomena, we follow Bordes et al. (2013) and remove those corrupted triples which exist in either the training, validation, or test set before getting the ranks. That means, we remove `Lyon` from the candidate list before getting the rank of `Paris` in the aforementioned example. We call the original setting "raw" and the new setting "filtered".

Optimal configurations. For each of the methods to be compared, we tune its hyperparameters in the ranges specified in Section 4.1, and select a best model that leads to the highest filtered MRR score on the validation set (with a total of 500 epochs over the training data). The optimal configurations for KALE are: $d = 100$, $\eta = 0.05$, $\gamma = 0.12$, and $\lambda = 1$ on FB122; $d = 50$, $\eta = 0.05$, $\gamma = 0.2$, and $\lambda = 0.1$ on WN18. To better see and understand the effects of rules, we use the same configuration for KALE-Trip, KALE-Pre, and KALE-Joint on each dataset.

Results. Table 4 and Table 5 show the results in the raw setting and filtered setting respectively. On each dataset we report the metrics on three sets: test-I, test-II, and the whole test set (denoted by test-all). Test-I contains test triples that cannot be directly inferred by performing pure logical inference on the training set, and hence might be intrinsically more difficult for the rules. The remaining test triples (i.e., the directly inferable ones) are included in Test-II. These triples have either been used directly as train-

		Raw						Filtered					
		Test-Incl			Test-Excl			Test-Incl			Test-Excl		
		MEAN	MED	HITS@10	MEAN	MED	HITS@10	MEAN	MED	HITS@10	MEAN	MED	HITS@10
FB122	KALE-Trip	0.150	49.0	34.2	0.235	17.0	44.1	0.267	14.0	46.2	0.321	8.0	52.9
	KALE-Joint	**0.175**	**36.0**	**36.6**	**0.265**	**15.0**	**45.9**	**0.290**	**11.0**	**49.3**	**0.349**	**7.0**	**54.2**
WN18	KALE-Trip	0.062	239.0	15.1	0.285	**4.0**	90.0	0.072	186.0	17.3	0.369	**2.0**	92.9
	KALE-Joint	**0.093**	**186.0**	**19.6**	**0.291**	**4.0**	**90.5**	**0.113**	**136.0**	**24.0**	**0.381**	**2.0**	**93.2**

Table 6: Comparison between KALE-Trip and KALE-Joint on Test-Incl and Test-Excl of FB122 and WN18.

ing instances in KALE-Pre, or encoded explicitly in training ground rules in KALE-Joint, making this set trivial for the rules to some extent. From the results, we can see that in both settings: (i) KALE-Pre and KALE-Joint outperform (or at least perform as well as) the other methods which use triples alone on almost all the test sets, demonstrating the superiority of incorporating logical rules. (ii) On the test-I sets which contain triples beyond the scope of pure logical inference, KALE-Joint performs significantly better than KALE-Pre. On these sets KALE-Joint can still beat all the baselines by a significant margin in most cases, while KALE-Pre can hardly outperform KALE-Trip. It demonstrates the capability of the joint embedding scenario to learn more predictive embeddings, through which we can make better predictions even beyond the scope of pure logical inference. (iii) On the test-II sets which contain directly inferable triples, KALE-Pre can easily beat all the baselines (even KALE-Joint). That means, for triples covered by pure logical inference, it is trivial to improve the performance by directly incorporating them as training instances.

To better understand how the joint embedding scenario can learn more predictive embeddings, on each dataset we further split the test-I set into two parts. Given a triple (e_i, r_k, e_j) in the test-I set, we assign it to the first part if relation r_k is covered by the rules, and the second part otherwise. We call the two parts Test-Incl and Test-Excl respectively. Table 6 compares the performance of KALE-Trip and KALE-Joint on the two parts. The results show that KALE-Joint outperforms KALE-Trip on both parts, but the improvements on Test-Incl are much more significant than those on Test-Excl. Take the filtered setting on WN18 as an example. On Test-Incl, KALE-Joint increases the metric MRR by 55.7%, decreases the metric MED by 26.9%, and increases the metric HITS@10 by 38.2%. On Test-Excl, however, MRR rises by 3.1%, MED remains the same, and HITS@10 rises by only 0.3%. This observation indicates that jointly embedding triples and rules helps to learn more predictive embeddings, especially for those relations that are used to construct the rules. This might be the main reason that KALE-Joint can make better predictions even beyond the scope of pure logical inference.

4.3 Triple Classification

This task is to verify whether an unobserved triple (e_i, r_k, e_j) is correct or not.

Evaluation protocol. We take the following evaluation protocol similar to that used in TransH (Wang et al., 2014). We first create labeled data for evaluation. For each triple in the test or validation set (i.e., a positive triple), we construct 10 negative triples for it by randomly corrupting the entities, 5 at the head position and the other 5 at the tail position.[6] To make the negative triples as difficult as possible, we corrupt a position using only entities that have appeared in that position, and further ensure that the corrupted triples do not exist in either the training, validation, or test set. We simply use the truth values (or distances) to classify triples. Triples with large truth values (or small distances) tend to be predicted as positive. To evaluate, we first rank the triples associated with each specific relation (in descending order according to their truth values, or in ascending order according to the distances), and calculate the average precision for that relation. We then report on the test sets the mean average precision (MAP)

[6] Previous work typically constructs only a single negative case for each positive one. We empirically found such a balanced classification task too simple for our datasets. So we consider a highly unbalanced setting, with a positive-to-negative ratio of 1:10, for which the previously used metric accuracy is no longer suitable.

	FB122			WN18		
	MAP (Test-I/II/ALL)			MAP (Test-I/II/ALL)		
TransE	0.552	0.852	0.634	0.592	0.993	0.958
TransH	0.576	0.758	0.641	0.604	0.978	0.947
TransR	0.572	0.699	0.619	0.412	0.854	0.836
KALE-Trip	0.578	0.829	0.652	0.618	0.995	0.953
KALE-Pre	0.575	**0.916**	0.668	0.620	**0.997**	**0.964**
KALE-Joint	**0.599**	0.870	**0.677**	**0.627**	**0.997**	0.961

Table 7: Triple classification results on the test-I, test-II, and test-all sets of FB122 and WN18.

aggregated over different relations.

Optimal configurations. The hyperparameters of each method are again tuned in the ranges specified in Section 4.1, and the best models are selected by maximizing MAP on the validation set. The optimal configurations for KALE are: $d = 100$, $\eta = 0.1$, $\gamma = 0.2$, and $\lambda = 0.1$ on FB122; $d = 100$, $\eta = 0.1$, $\gamma = 0.2$, and $\lambda = 0.001$ on WN18. Again, we use the same configuration for KALE-Trip, KALE-Pre, and KALE-Joint on each dataset.

Results. Table 7 shows the results on the test-I, test-II, and test-all sets of our datasets. From the results, we can see that: (i) KALE-Pre and KALE-Joint outperform the other methods which use triples alone on almost all the test sets, demonstrating the superiority of incorporating logical rules. (ii) KALE-Joint performs better than KALE-Pre on the test-I sets, i.e., triples that cannot be directly inferred by performing pure logical inference on the training set. This observation is similar to that observed in the link prediction task, demonstrating that the joint embedding scenario can learn more predictive embeddings and make predictions beyond the capability of pure logical inference.

5 Conclusion and Future Work

In this paper, we propose a new method for jointly embedding knowledge graphs and logical rules, referred to as KALE. The key idea is to represent and model triples and rules in a unified framework. Specifically, triples are represented as atomic formulae and modeled by the translation assumption, while rules as complex formulae and by the t-norm fuzzy logics. A global loss on both atomic and complex formulae is then minimized to perform the embedding task. Embeddings learned in this way are compatible not only with triples but also with rules, which are certainly more useful for knowledge acquisition and inference. We evaluate KALE with the link prediction and triple classification tasks on WordNet and Freebase data. Experimental results show that joint embedding brings significant and consistent improvements over state-of-the-art methods. More importantly, it can obtain more predictive embeddings and make better predictions even beyond the scope of pure logical inference.

For future work, we would like to (i) Investigate the efficacy of incorporating other types of logical rules such as $\forall x, y, z : (x, \texttt{Capital-Of}, y) \Rightarrow \neg(x, \texttt{Capital-Of}, z)$. (ii) Investigate the possibility of modeling logical rules using only relation embeddings as suggested by Demeester et al. (2016), e.g., modeling the above rule using only the embedding associated with `Capital-Of`. This avoids grounding, which might be time and space inefficient especially for complicated rules. (iii) Investigate the use of automatically extracted rules which are no longer hard rules and tolerant of uncertainty.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by the National Natural Science Foundation of China (grant No. 61402465) and the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200).

References

Islam Beltagy and Raymond J. Mooney. 2014. Efficient markov logic inference for natural language semantics. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence - Workshop on Statistical Relational Artificial Intelligence*, pages 9–14.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Matthias Bröcheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic similarity logic. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 73–82.

Kai-wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1579.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Regularizing relation representations by first-order implications. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Workshop on Automated Knowledge Base Construction*, pages 75–80.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 84–94.

Petr Hájek. 1998. *The metamathematics of fuzzy logic*. Kluwer.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R. Obozinski. 2012. A latent factor model for highly multi-relational data. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pages 3167–3175.

Shangpu Jiang, Daniel Lowd, and Dejing Dou. 2012. Learning to refine an automatically extracted knowledge base using markov logic. In *Proceedings of 12th IEEE International Conference on Data Mining*, pages 912–917.

Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. 2006. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, pages 381–388.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of the 14th International Semantic Web Conference*, pages 640–655.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Yuanfei Luo, Quan Wang, Bin Wang, and Li Guo. 2015. Context-dependent knowledge graph embedding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1656–1661.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 156–166.

Maximilian Nickel, Volker Tresp, and Hans P. Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Maximilian Nickel, Volker Tresp, and Hans P. Kriegel. 2012. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 271–280.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *Proceed-*

ings of the 12th International Semantic Web Conference, pages 542–557.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference on North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Tim Rocktäschel, Matko Bošnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics - Workshop on Semantic Parsing*, pages 45–49.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 926–934.

Giorgos Stoilos, Giorgos B. Stamou, Jeff Z. Pan, Vassilis Tzouvaras, and Ian Horrocks. 2007. Reasoning with very expressive fuzzy description logics. *Journal of Artificial Intelligence Research*, 30:273–320.

Ilya Sutskever, Joshua B. Tenenbaum, and Ruslan R. Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pages 1821–1828.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1859–1865.

Evgenia Wasserman-Pritsker, William W. Cohen, and Einat Minkov. 2015. Learning to identify the best contexts for knowledge-based wsd. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1662–1667.

Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale knowledge base completion: inferring via grounding network sampling over selected instances. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1331–1340.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371.

Zhao Xu, Volker Tresp, Kai Yu, and Hanspeter Kriegel. 2006. Infinite hidden relational models. In *Proceedings of Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 544–551.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 267–272.

Learning Connective-based Word Representations for Implicit Discourse Relation Identification

Chloé Braud
CoAStaL, Dep of Computer Science
University of Copenhagen
University Park 5, 2100 Copenhagen, Denmark
braud@di.ku.dk

Pascal Denis
Magnet Team, INRIA Lille – Nord Europe
59650 Villeneuve dAscq, France
pascal.denis@inria.fr

Abstract

We introduce a simple semi-supervised approach to improve implicit discourse relation identification. This approach harnesses large amounts of automatically extracted discourse connectives along with their arguments to construct new distributional word representations. Specifically, we represent words in the space of discourse connectives as a way to directly encode their rhetorical function. Experiments on the Penn Discourse Treebank demonstrate the effectiveness of these task-tailored representations in predicting implicit discourse relations. Our results indeed show that, despite their simplicity, these connective-based representations outperform various off-the-shelf word embeddings, and achieve state-of-the-art performance on this problem.

1 Introduction

A natural distinction is often made between explicit and implicit discourse relations depending on whether they are lexicalized by a connective or not, respectively. To illustrate, the *Contrast* relation in example (1a) is triggered by the connective *but*, while it is not overtly marked in example (1b).[1] Given the lack of strong explicit cues, the identification of implicit relations is a much more challenging and still open problem. The typically low performance scores for this task also hinder the development of text-level discourse parsers (Lin et al., 2010; Xue et al., 2015): implicit discourse relations account for around half of the data for different genres and languages (Prasad et al., 2008; Sporleder and Lascarides, 2008; Taboada, 2006; Subba and Di Eugenio, 2009; Soria and Ferrari, 1998; Versley and Gastel, 2013).

(1) a. The house has voted to raise the ceiling to $3.1 trillion, *but* the Senate isn't expected to act until next week at the earliest.

b. That's not to say that the nutty plot of "A Wild Sheep Chase" is rooted in reality. It's imaginative and often funny.

The difficulty of this task lies in its dependence on a wide variety of linguistic factors, ranging from syntax, lexical semantics and also world knowledge (Asher and Lascarides, 2003). In order to deal with this issue, a common approach is to exploit hand-crafted resources to design features capturing lexical, temporal, modal, or syntactic information (Pitler et al., 2009; Park and Cardie, 2012). By contrast, more recent work show that using simple low-dimensional word-based representations, either cluster-based or distributed (aka word embeddings), yield comparable or better performance (Rutherford and Xue, 2014; Braud and Denis, 2015), while dispensing with feature engineering.

While standard low-dimensional word representations appear to encode relevant linguistic information, they have not been built with the specific rhetorical task in mind. A natural question is therefore whether one could improve implicit discourse relation identification by using word representations that are more directly related to the task. The

[1] These examples are taken from documents wsj_0008 and wsj_0037, respectively, of the PDTB.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 203–213,
Austin, Texas, November 1-5, 2016.

problem of learning good representation for discourse has been recently tackled by Ji and Eisenstein (2014) on the problem of text-level discourse parsing. Their approach uses two recursive neural networks to jointly learn the task and a transformation of the discourse segments to be attached. While this type of joint learning yields encouraging results, it is also computationally intensive, requiring long training times, and could be limited by the relatively small amount of manually annotated data available.

In this paper, we explore the possibility of learning a distributional word representation adapted to the task by selecting relevant rhetorical contexts, in this case discourse connectives, extracted from large amounts of automatically detected connectives along with their arguments. Informally, the assumption is that the estimated word-connective co-occurrence statistics will in effect give us an important insight to the rhetorical function of different words. The learning phase in this case is extremely simple, as it amounts to merely estimating co-occurrence frequencies, potentially combined with a reweighting scheme, between each word appearing in a discourse segment and its co-occurring connective. To assess the usefulness of these connective-based representations,[2] we compare them with pre-trained word representations, like Brown clusters and other word embeddings, on the task of implicit discourse relation identification. Our experiments on the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) show that these new representations deliver improvements over systems using these generic representations and yield state-of-the-art results, and this without the use of other hand-crafted features, thus also alleviating the need for external linguistic resources (like lexical databases). Thus, our approach could be easily extended to resource-poor languages as long as connectives can be reliably identified on raw texts.

Section 2 summarizes related work. In Section 3, we detail our connective-based distributional word representation approach. Section 4 presents the automatic annotation of the explicit examples used to build the word representation. In Section 5, we describe our comparative experiments on the PDTB.

[2]Available at `https://bitbucket.org/chloebt/discourse-data`.

2 Related Work

Implicit discourse relation identification has attracted growing attention since the release of the PDTB, the first discourse corpus to make the distinction between explicit and implicit examples. Within the large body of research on this problem, we identify two main strands directly relevant to our work.

2.1 Finding the Right Input Representation

The first work on this task (Marcu and Echihabi, 2002), which pre-dates the release of the PDTB, proposed a simple word-based representation: they use the Cartesian product of words appearing in the two segments. Given the knowledge-rich nature of the task, following studies attempted to exploit various hand-crafted resources and pre-processing systems to enrich their model with information on modality, polarity, tense, lexical semantics, and syntax, possibly combined with feature selection methods (Pitler et al., 2009; Lin et al., 2009; Park and Cardie, 2012; Biran and McKeown, 2013; Li and Nenkova, 2014). Interestingly, Park and Cardie (2012) concluded on the worthlessness of word-based features, as long as hand-crafted linguistic features were used. More recent studies however reversed this conclusion (Rutherford and Xue, 2014; Braud and Denis, 2015), demonstrating that word-based features can be effective provided they were not encoded using the sparse one-hot representation, but instead with a denser one (cluster based or distributed). This paper takes one step further by testing whether learning a simple task-specific, distributional word representation could lead to further improvements.

As noted, some previous work have also attempted to learn discourse-specific representation for the related problem of discourse parsing. Thus, Ji and Eisenstein (2014) reports improvements on the RST Discourse Treebank (Carlson et al., 2001), by jointly learning a combination of the discourse units, represented by bag-of-words in a one-hot encoding, along with the sequence of actions of their shift-reduce parser. Our approach is attractively simpler, since training reduces to collecting frequency counts, and it can easily generate representations for unseen words without having to retrain the whole system.

2.2 Leveraging Explicit Discourse Data

Another line of work, also initiated in (Marcu and Echihabi, 2002), propose to deal with the sparseness of the word pair representation by using additional data automatically annotated using discourse connectives. An appeal of this strategy is that one can easily identify explicit relations in raw data, as performance are high on this task (Pitler et al., 2009) and it is even possible to rely on simple heuristics (Marcu and Echihabi, 2002; Sporleder and Lascarides, 2005; Lan et al., 2013). It has been shown, however, that using explicit examples as additional data for training an implicit relation classifier degrades performance, due to important distribution differences (Sporleder and Lascarides, 2008).

Recent attempts to overcome this issue involve domain adaptation strategies (Braud and Denis, 2014; Ji et al., 2015), sample selection (Rutherford and Xue, 2015; Wang et al., 2012), or multi-task algorithms (Lan et al., 2013). However, it generally involves longer training time since models are built on a massive amount of data, the strategy requiring a large corpus of explicit examples to overcome the noise induced by the automatic annotation strategy. In this paper, we circumvent this problem by using explicit data only for learning our word representations and not for estimating the parameters of our implicit classification model. Some aspects of the present work are similar to Biran and McKeown (2013) in that they also exploit explicit data to compute co-occurrence statistics between word pairs and connectives. But the perspective is reversed, as they represent connectives in the contexts of co-occurring word pairs, with the aim of deriving similarity features between each implicit example and each connective. Furthermore, their approach did not outperform state-of-the-art systems.

3 The Connective Vector Space Model

Our discourse-based word representation model is a simple variant of the standard vector space model (Turney and Pantel, 2010): that is, it represents individual words in specific co-occurring contexts (in this case, discourse connectives) that define the dimensions of the underlying vector space. Our specific choice of contexts was guided by two main considerations. On the one hand, we aim at learning word representations that live in a relatively low-dimensional space, so as to make learning a classification function over that space feasible. The number of parameters of that function grows proportionally with that of the input size. Although there is often a lack of consensus among linguists as to the exact definition of discourse connectives, they nevertheless form a closed class. For English, the PDTB recognizes 100 distinct connectives. On the other hand, we want to learn a vectorial representation that captures relevant aspects of the problem, in this case the rhetorical contribution of words. Adapting Harris (1954)'s famous quote, we make the assumption that words occurring in similar *rhetorical* contexts tend to have similar *rhetorical* meanings. Discourse connectives are by definition strong rhetorical cues. As an illustration, Pitler et al. (2009) found that connectives alone unambiguously predict a single relation in 94% of the PDTB level 1 data. By using connectives as contexts, we are thus linking each word to a relation (or a small set of relations), namely those that can be triggered by this connective. Note that for level 2 relations in the PDTB, the connectives are much more ambiguous (86.77% reported in (Lin et al., 2010)), and it could be also the case if we expand the list of forms considered as connectives for English, or if we try to deal with other languages and domains. We however believe that the set of relations that can be triggered by a connective is limited (not all relations can be expressed by the same connective), and that one attractive feature of our strategy is precisely to keep this ambiguity.

Before turning to the details of how we construct our distributional connective-based model, note that we decided to learn a unique representation for any individual word, irrespective of its position (with)in a particular segment. That is, we represent both arguments of a connective as a single bag of words. Other designs are of course possible: we could directly learn distinct word representation for left and right segment words, or even the pair of words (Conrath et al., 2014), to take into account the fact that some relations are oriented (e.g. *Reason* contains the cause in the first argument and *Result* in the second one). An obvious drawback of these more expressive representations is that they would need much more data to compute a robust estimate of the frequency counts.

	but			*while*			*before*		
Word	Freq.	TF-IDF	PPMI-IDF	Freq.	TF-IDF	PPMI-IDF	Freq.	TF-IDF	PPMI-IDF
reality	12	0.0	0.0	13	0.0	0.0	10	0.0	0.0
not	142	0.37	0.36	201	0.18	0.06	0	0.0	0.0
week	0	0.0	0.0	110	0.10	0.04	90	0.12	0.12

Table 1: Illustrative example of association measures between connectives and words.

3.1 Building the Distributional Representation

Our discourse-based representations of words are obtained by computing a matrix of co-occurrence between the words and the chosen contexts. The frequency counts are then weighted in order to highlight relevant associations. More formally, we note $\mathcal{V}$ the set of the n words appearing in the arguments, and $\mathcal{C}$ the set of the m connective contexts. We build the matrix $\mathbf{F}$, of size $n \times m$, by computing the frequency of each element of $\mathcal{V}$ with each element of $\mathcal{C}$. We note $f_{i,j}$ the frequency of the word $w_i \in \mathcal{V}$ appearing in one argument of the connective $c_j \in \mathcal{C}$. We use two standard weighting functions on these raw frequencies: the normalized Term Frequency (TF), eq. (1), and the Positive Pointwise Mutual Information (PPMI), eq. (2), which is a version of the PMI where negative values are ignored (with $p_{i,j}$ the joint probability that the word w_i appears with connective c_j, and $p_{i,*}$ and $p_{*,j}$, relative frequency of resp. w_i and c_j). These two measures are then normalized by multiplying the value by the Inverse Document Frequency (IDF) for a word w_i, eq. (3), as in (Biran and McKeown, 2013). In the final matrices, the i^{th} row corresponds to the m-dimensional vector for the i^{th} word of $\mathcal{V}$. The j^{th} column is a vector corresponding to the j^{th} connective.

$$\text{TF}_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{n} f_{k,j}} \quad (1)$$

$$\text{PPMI}_{i,j} = \max(0, \log\left(\frac{p_{i,j}}{p_{i,*}\, p_{*,j}}\right)) \quad (2)$$

$$\text{IDF}_i = \log\left(\frac{m}{\sum_{k=1}^{m} f_{i,k}}\right) \quad (3)$$

Table 1 illustrates the weighting of the words using the TF and the PPMI normalized with IDF. For instance, the presence of the negation "not" is positively linked to *Contrast* through *but* and *while* whereas it receives a null or a very small weight with the temporal connective *before*. The final vector for this word, $< 0.37, 0.18, 0.0 >$ with TF-IDF or $< 0.36, 0.06, 0.0 >$ with PPMI-IDF, is intended to guide the implicit model toward a contrastive relation, thus potentially helping in identifying the relation in example (1b). In contrast, the word "week" is more likely to be found in the arguments of temporal relations that can be triggered by *before* but also *while*, an ambiguity kept in our representation whereas approaches based on using explicit examples as new training data generally choose to annotate them using the most frequent sense associated with the connective, often limiting themselves to the less ambiguous ones (Marcu and Echihabi, 2002; Sporleder and Lascarides, 2008; Lan et al., 2013; Braud and Denis, 2014; Rutherford and Xue, 2015). Finally, a word occuring with all connectives, not discriminant, such as "reality" is associated with a null weight for all dimensions: it thus has no impact on the model.

Since we have 100 connectives for the PDTB, the representation is already of quite low dimensionality. However, it has been shown (Turney and Pantel, 2010) that using a dimensionality reduction algorithm could help capturing the latent dimensions between the words and their contexts and reducing the noise. We thus also test versions with a reduction Components Analysis (PCA) (Jolliffe, 2002).

3.2 Using the Word-based Representation

So far, our distributional framework associates a word with a d-dimensional vector (where $d \leq m$). We now need to represent a pair of arguments (i.e., the spans of text linked by a relation), modeled here as a pair of bags of words. Following (Braud and Denis, 2015), we first sum all word vectors contained in each segment, thus obtaining a d-dimensional vector for each segment. We then combine the two segment vectors to build a composite vector representing the pair of arguments, by ei-

ther concatenating the two segment vectors (leading to a $2d$-dimensional vector) or by computing the Kronecker product between them (leading to a d^2-dimensional vector). Finally, these segment-pair representations will be normalized using the L_2 norm to avoid segment size effects. These will then be used as the input of a classification model, as described in Section 5. Given these combination schemes, it should be clear that despite the fact that each individual word receives a unique vectorial representation irrespective of its position, the parameters of the classification model associated with a given word are likely to be different depending of whether it appears in the left or right segment.

4 Automatic Annotation of Explicit Examples

In order to collect reliable word-connective co-occurrence frequencies, we need a large corpus where the connectives and their arguments have been identified. We therefore rely on automatic annotation of raw data, instead of using the relatively small amount of explicit examples manually annotated in the PDTB (roughly $18,000$ examples). Specifically, we used the *Bllip* corpus[3] composed of news articles from the *LA Times*, the *Washington Post*, the *New York Times* and *Reuters* and containing 310 millions of words automatically POS-tagged.

Identifying the Connectives and their Arguments We have two tasks to perform: identifying the connectives and extracting their arguments.[4] Rather than relying on manually defined patterns to annotate explicit examples (Marcu and Echihabi, 2002; Sporleder and Lascarides, 2008; Rutherford and Xue, 2015), we use two binary classification models inspired by previous works on the PDTB (Pitler and Nenkova, 2009; Lin et al., 2010): the first one identifies the connectives and the second one localizes the arguments between inter- and intra-sentential, an heuristic being then used to decide on the exact boundaries of the arguments.

Discourse connectives are words (e.g., *but*, *since*) or grammaticalized multi-word expressions (e.g., *as soon as*, *on the other hand*) that may trigger a discourse relation. However, these forms can also appear without any discourse reading, such as *because* in: *He can't sleep because of the deadline*. We thus need to disambiguate these forms between discourse and non discourse readings, a task that has proven to be quite easy on the PDTB (Pitler and Nenkova, 2009). This is the task performed by our first binary classifier: a pattern-matching is used to identify all potential connectives, and the model predicts if they have discourse reading in context.

We then need to extract the arguments of the identified connectives, that is the two spans of text linked by the connective. This latter task has proven to be extremely hard on the PDTB (Lin et al., 2010; Xue et al., 2015) because of some annotation principles that make the possible types of argument very diverse. As first proposed in (Lin et al., 2010), we thus split this task into two subtasks: identifying the relative positions of the arguments and delimiting their exact boundaries.

For an explicit example in the PDTB, one argument, called *Arg2*, is linked to the connective, and thus considered as easy to extract (Lin et al., 2010). The other argument, called *Arg1*, may be located at different places relative to *Arg2* (Prasad et al., 2008): we call intra-sentential the examples where *Arg1* is a clause within the same sentence as *Arg2* (60.9% of the explicit examples in the PDTB), and inter-sentential the other examples, that is *Arg1* is found in the previous sentence, in a non-adjacent previous sentence (9%) or in a following sentence (less than 0.1%). In this work, we build a localization model by only considering these two coarse cases – the example is either intra- or inter-sentential. Note that this distinction is similar to what has been done in (Lin et al., 2010): more precisely, these authors distinguish between "same-sentence" and "previous sentence" and ignore the cases where the *Arg1* is in a following sentence. We rather choose to include them as being also inter-sentential. When the position of *Arg1* has been predicted, an heuristic is in charge of finding the exact boundaries of the arguments.

Here, the problem is that in addition to the variety of locations, the annotators were almost free to choose any boundary for an argument in the PDTB:

[3] https://catalog.ldc.upenn.edu/LDC2008T13

[4] Note that contrary to studies using automatically annotated explicit examples as new training data, we do not need to annotate the relation triggered by the connective.

an argument can cover only a part of a sentence, an entire sentence or several sentences. Statistical approaches intended to solve this task lead for now to low performance even when complex sequential models are used, and they often rely on the syntactic configurations (Lin et al., 2010; Xue et al., 2015). We thus decided to define an heuristic to perform this task, following the simplifying assumptions also used in previous work since (Marcu and Echihabi, 2002). We assume that: (1) *Arg1* is either in the same sentence as *Arg2* or in the previous one, (2) an argument covers at most one sentence and (3) a sentence contains at most two arguments. As it can be deduced from (1), our final model ignores the finer distinctions one can make for the position of inter-sentential examples (i.e. we never extract *Arg1* from a non-adjacent previous sentence or a following one).

According to these assumptions, once a connective is identified, knowing its localization is almost enough to identify the boundaries of its arguments. More precisely, if a connective is predicted as inter-sentential, then our heuristic picks the entire preceding sentence as *Arg1*, *Arg2* being the sentence containing the connective, according to assumptions (1) and (2). If a connective is predicted as intra-sentential, then the sentence containing the connective is split into two segments – according to (3) –, more precisely, the sentence is split around the connective using the punctuation and making it necessary to have a verb in each argument.

Settings We thus built two models using the PDTB: one to identify the discourse markers (connective *vs* not connective), and one to identify the position of the arguments with respect to the connective (inter- *vs* intra-sentential). The PDTB contains $18,459$ explicit examples for 100 connectives. For both models, we use the same split of the data as in (Lin et al., 2014). The test set contains 923 positive instances of connectives and $2,075$ negative instances, and 546 inter-sentential and 377 intra-sentential examples. Both models are built using a logistic regression model optimized on the development set (see Section 5), and the same simple feature set (Lin et al., 2014; Johannsen and Sgaard, 2013) without syntactic information. With C the connective, F the following word and P the previous one, our features are: C, P+C, C+F, C-POS[5], P-POS, F-POS, P-POS+C-POS and C-POS+F-POS.

Results Our model identifies discourse connective with a micro-accuracy of 92.9% (macro-F_1 91.5%). These scores are slightly lower than the state-of-the-art in micro-accuracy, but high enough to rely on this annotation. When applying our model to the *Bllip* data, we found 4 connectives that correspond to no examples. We thus have examples for only 96 connectives. For distinguishing between inter- and intra-sentential examples, we get a micro-accuracy of 96.1% (macro-F_1 96.0), with an F_1 of 96.7 for the intra- and 95.3 for the inter-sentential class, again close enough to the state-of-the-art (Lin et al., 2014).

Coverage Using these models on *Bllip*, we are able to extract around 3 million connectives, along with their arguments. Our word representation has a large vocabulary (see Table 2) compared to existing off-the-shelf word vectors, with only $2,902$ out of vocabulary (OOV) tokens in set of implicit relations.[6]

	# words	# OOV
HLBL	$246,122$	$5,439$
CnW	$268,810$	$5,638$
Brown	$247,339$	$5,413$
H-PCA	$178,080$	$7,042$
Bllip	$\mathbf{422,199}$	$\mathbf{2,902}$

Table 2: Lexicon coverage for *Brown* clusters (Brown et al., 1992), Collobert and Weston (*CnW*) (Collobert and Weston, 2008) and hierarchical log-bilinear embeddings (*HLBL*) (Mnih and Hinton, 2007) using the implementation in (Turian et al., 2010), Hellinger PCA (*H-PCA*) (Lebret and Collobert, 2014) and our connective-based representation (*Bllip*).

5 Experiments

Our experiments investigate the relevance of our connective-based representations for implicit discourse relation identification, recast here as multi-class classification problem. That is, we aim at evaluating the usefulness of having a word representation linked to the task, compared to using generic

[5]The connective POS is either the node covering the connective, or the POS of its first word if no such node exists.

[6]Training and development sets, only.

Relation	Train	Dev	Test
Temporal	665	93	68
Contingency	3,281	628	276
Comparison	1,894	401	146
Expansion	6,792	1,253	556
Total	12,632	2,375	1,046

Table 3: Number of examples in train, dev, test.

word representations (either one-hot, cluster-based or distributed), and whether they encode all the information relevant to the task, thus comparing systems with or without additional hand-crafted features.

5.1 Data

The PDTB (Prasad et al., 2008) is the largest corpus annotated for discourse relations, formed by newspaper articles from the Wall Street Journal. It contains 16,053 pairs of spans of text annotated with one or more implicit relations. The relation set is organized in a three-level hierarchy. We focus on the level 1 coarse-grained relations and keep only the first relation annotated. We use the most spread split of the data, used in (Rutherford and Xue, 2014; Rutherford and Xue, 2015; Braud and Denis, 2015) among others, that is sections 2-20 for training and 21-22 for testing. The other sections are used for development. The number of examples per relation is reported in Table 3. It can be seen that the dataset is highly imbalanced, with the relation *Expansion* accounting for more than 50% of the examples.

5.2 Settings

Feature Set Our main features are based on the words occurring in the arguments. We test simple baselines using raw tokens. The first one uses the Cartesian product of the tokens, a feature template, generally called "Word pairs", used in most of the previous study for this task as in (Marcu and Echihabi, 2002; Pitler et al., 2009; Lin et al., 2011; Braud and Denis, 2015; Ji et al., 2015). It is the sparsest representation one can build from words, and it corresponds to using the combination scheme based on the Kronecker product to combine the one-hot vectors representing each word. We also report results with a less sparse version where the vectors are combined using concatenation.

We also compare our systems to previous approaches that make use of word based representations but not linked to the task. We implement the systems proposed in (Braud and Denis, 2015) in multiclass, that is using the *Brown* clusters (Brown et al., 1992), the Collobert and Weston (Collobert and Weston, 2008) and the hierarchical log-bilinear embeddings (Mnih and Hinton, 2007) using the implementation in (Turian et al., 2010)[7], and the *HPCA* (Lebret and Collobert, 2014)[8]. We use the combination schemes described in Section 3 to build vector representations for pairs of segments. For these systems and ours, using the connective-based representations, the dimensionality of the final model depends on the number of dimensions d of the representation used and on the combination scheme – the concatenation leading to $2d$ dimensions and the Kronecker product to d^2.

All the word representations used – the off-the-shelf representations as well as our connective-based representation (see Section 4) – are solely or mainly trained on newswire data, thus on the same domain as our evaluation data. The *CnW* embeddings we use in this paper, with the implementation in (Turian et al., 2010), as well as the *HLBL* embeddings have been obtained using the RCV1 corpus, that is one year of Reuters English newswire. The *H-PCA* have been built on the Wikipedia, the Reuters corpus and the Wall street Journal. We thus do not expect any out-of-domain issue when using these representations.

Finally, we experiment with additional features proposed in previous studies and well described in (Pitler et al., 2009; Park and Cardie, 2012): production rules[9], information on verbs (average verb phrases length and Levin classes), polarity (Wilson et al., 2005), General Inquirer tags (Stone and Kirsh, 1966), information about the presence of numbers and modals, and first, last and first three words. We concatenate these features to the ones built using word representations.

[7] http://metaoptimize.com/projects/wordreprs/

[8] http://lebret.ch/words/

[9] We use the gold standard parses provided in the Penn Treebank (Marcus et al., 1993).

Model We train a multinomial multiclass logistic regression model.[10] In order to deal with the class imbalance issue, we use a sample weighting scheme where each instance has a weight inversely proportional to the frequency of the class it belongs to.

Parameters We optimize the hyper-parameters of the algorithm, that is the regularization norm (L1 or L2), and the strength of the regularization $C \in \{0.001, 0.005, 0.01, 0.1, 0.5, 1, 5, 10, 100\}$. When using additional features or one-hot sparse encodings over the pairs of raw tokens, we also optimize a filter on the features by defining a frequency cutoff $t \in \{1, 2, 5, 10, 15, 20\}$. We evaluate the unsupervised representations with different number of dimensions. We test versions of the *Brown* clusters with 100, 320, 1,000 and 3,200 clusters, of the Collobert and Weston embeddings with 25, 50, 100 and 200 dimensions, of the hierarchical log-bilinear embeddings with 50 and 100 dimensions, and of the Hellinger PCA with 50, 100 and 200 dimensions. Finally, the distributional representations of words based on the connective are built using either no PCA – thus corresponding to 96 dimensions– , or a PCA[11] keeping the first k dimensions with $k \in \{2, 5, 10, 50\}$.[12] We optimize both the hyper-parameters of the algorithm and the number of dimensions of the unsupervised representation on the development set based on the macro-F_1 score, the most relevant measure to track when dealing with imbalanced data.

5.3 Results

Our results are summarized in Table 4. Using our connective-based word representation allows improvements of above 2% in macro-F_1 over the baseline systems based on raw tokens (*One-hot*), the competitive systems using pre-trained representations (*Brown* and *Embed.*) and the state-of-the-art results in terms of macro-F_1 (R&X 15). These improvements demonstrate the efficiency of the representation for this task.

We found that using an unsupervised word representation generally leads to improvements over the use of raw tokens (*One-hot*), a conclusion in line with the results reported in (Braud and Denis, 2015) for binary systems. However, contrary to their findings, in multiclass, the best results are not obtained using the *Brown* clusters, but rather the dense, real valued representations (*Embed.* and *Bllip*). Furthermore, concerning the combination schemes, the concatenation ($\oplus$) generally outperforms the Kronecker product ($\otimes$), in effect favoring lower dimensional models.

Representation	Macro-F_1	Acc.
One-hot $\otimes$	39.0	48.6
One-hot $\oplus$	40.2	50.2
Best *Brown* $\otimes$	37.5	50.6
Best *Brown* $\oplus$	40.6	51.2
Best *Embed.* $\otimes$	41.0	51.7
Best *Embed.* $\oplus$	41.6	50.1
Best dense + add feat.	40.8	51.2
Bllip TF-IDF $\otimes$	41.4	51.0
Bllip TF-IDF $\oplus$	40.1	50.0
Bllip PPMI-IDF $\otimes$	38.9	48.2
Bllip PPMI-IDF $\oplus$	**42.2***	52.5
Best *Bllip* + add feat.	**42.8***	51.7
R&X 15	40.5	57.1

Table 4: Results for multiclass experiments. R&X 15 are the scores reported in (Rutherford and Xue, 2015) ; *One-hot*: one-hot encoding of raw tokens ; *Brown* and *Embed.*: pre-trained representations ; *Bllip*: connective based representation. * $p \leq 0.1$ compared to *One-hot* $\otimes$ with t-test and Wilcoxon.

More importantly, the distributional representations based on connectives (*Bllip*) allows performance at least similar or even better than those obtained with the other dense representations unconnected to the task (*Embed.*). While simply based on weighted co-occurrence counts, thus really easy and fast to build, these representations generally outperform the ones learned using neural networks (see CnW and HLBL in Figure 1). Besides, our second best representation is also distributional, namely HPCA (see Figure 1). These result are thus in line with the conclusions in (Lebret and Collobert, 2014) for other NLP tasks: distributional representations, while simpler to obtain, may allow similar results than distributed ones.

[10] `http://scikit-learn.org/dev/index.html`.

[11] Implemented in scikit-learn, applied with default settings.

[12] Keeping resp. 11.3%, 36.6%, 56.2% or 95.3% of the variance of the data.

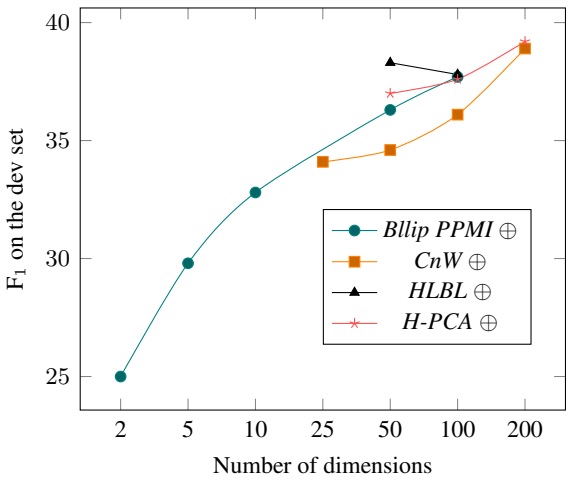


Figure 1: F_1 scores on dev against the number of dimensions.

Our best results with *Bllip* are obtained without the use of a dimensionality reduction method, thus keeping the 96 dimensions corresponding to the connectives identified in the raw data. Our new word representation like the other low-dimensional ones yield higher scores as one increases the number of dimensions (see Figure 1). This could be a limitation of our strategy, since the number of connectives in the PDTB is fixed. However, one could easily expand our model to include additional lexical elements that might have a rhetorical function such as modals or specific expressions such as *one reason is*.

We also tested the addition of hand-crafted features traditionally used for the task. We found that, either using a pre-trained word representation or our representation based on connectives, adding these features leads to small or even no improvements and suggest that these representations already encode the information provided by these features. This conclusion has however to be nuanced: when looking at the scores per relation reported in Table 5, the use of the connective based word representation alone allows the best performance for *Temporal* and *Contingency*, but the addition of new features dramatically increase the scores for *Comparison* showing that some information are missing for this relation. Moreover, this relation is the one taking the most advantage of the addition of explicit data in (Rutherford and Xue, 2015), demonstrating that these data could probably provide even more information than the ones we leverage through our representations.

Finally, our results are similar or even better than those reported in (Rutherford and Xue, 2015) in terms of macro-F_1. Our systems correspond however to a lower micro-accuracy. Looking at the scores per relation in Table 5, we found that we obtain better results for all the relations except *Expansion*, the most represented, which could explain the loss in accuracy. It is noteworthy that we generally obtain better results even without the additional features used in this work. Moreover, our systems requires lower training time (since we only train on implicit examples) and alleviate the need for the sample selection strategy used to deal with the distribution differences between the two types of data.

	Bllip PPMI-IDF ⊕		*Bllip* + *add feat*		R&X 15	
Rel	Prec	F_1	Prec	F_1	Prec	F_1
Temp	23.0	**29.9**	23.7	27.9	38.5	14.7
Cont	49.6	**47.1**	46.7	46.3	49.3	43.9
Comp	35.9	27.7	35.0	**34.3**	44.9	34.2
Exp	62.8	64.0	63.7	62.6	61.4	**69.1**

Table 5: Scores per relation for multiclass experiments, "R&X 15" are the scores reported in (Rutherford and Xue, 2015).

6 Conclusion

We presented a new approach to leverage information from explicit examples for implicit relation identification. We showed that building distributional representations linked to the task through connectives allows state-of-the-art performance and alleviates the need for additional features. Future work includes extending the representations to new contexts – such as the Alternative Lexicalization annotated in the PDTB, the modals or some adverbs – using more sophisticated weighting schemes (Lebret and Collobert, 2014) and testing this strategy for other languages and domains.

Acknowledgements

We thank the three anonymous reviewers for their comments. Chloé Braud was funded by the ERC Starting Grant LOWLANDS No. 313695. Pascal Denis was supported by ERC Grant STAC No. 269427, and by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of ACL*.

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *Proceedings of COLING*.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of EMNLP*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Juliette Conrath, Stergos Afantenos, Nicholas Asher, and Philippe Muller. 2014. Unsupervised extraction of semantic relations using discourse cues. In *Proceedings of Coling*.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of ACL*.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the gap: Domain adaptation from explicit to implicit discourse relations. In *Proceedings of EMNLP*.

Anders Johannsen and Anders Sgaard. 2013. Disambiguating explicit discourse connectives without oracles. In *Proceedings of IJCNLP*.

Ian Jolliffe. 2002. *Principal component analysis*. Wiley Online Library.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *Proceedings of ACL*.

Rémi Lebret and Ronan Collobert. 2014. Word emdeddings through Hellinger PCA. In *Proceedings of ACL*.

Junyi Jessy Li and Ani Nenkova. 2014. Reducing sparsity improves the recognition of implicit discourse relations. In *Proceedings of SIGDIAL*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of EMNLP*.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A PDTB-styled end-to-end discourse parser. Technical report, National University of Singapore.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of ACL-HLT*.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20:151–184.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of ACL*.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of SIGDIAL Conference*.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of ACL-IJCNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of LREC*.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through Brown cluster pair representation and coreference patterns. In *Proceedings of EACL*.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of NAACL-HLT*.

Claudia Soria and Giacomo Ferrari. 1998. Lexical marking of discourse relations - some experimental findings. In *Proceedings of the ACL Workshop on Discourse Relations and Discourse Markers*.

Caroline Sporleder and Alex Lascarides. 2005. Exploiting linguistic cues to classify rhetorical relations. In *Proceedings of RANLP-05*.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14:369–416.

Philip J. Stone and John Kirsh. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of ACL-HLT*.

Maite Taboada. 2006. Discourse markers as signals (or not) of rhetorical relations. *Journal of Pragmatics*, 38:567–592.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning : Vector space models of semantics. *Journal of Artificial Intelligence Research*, pages 141–188.

Yannick Versley and Anna Gastel. 2013. Linguistic tests for discourse relations in the TüBa-D/Z corpus of written German. *Dialogue & Discourse*, 4(2):142–173.

Xun Wang, Sujian Li, Jiwei Li, and Wenjie Li. 2012. Implicit discourse relation recognition by selecting typical training examples. In *Proceedings of COLING 2012: Technical Papers*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The CoNLL-2015 shared task on shallow discourse parsing. In *Proceedings of CoNLL*.

Aspect Level Sentiment Classification with Deep Memory Network

Duyu Tang, Bing Qin*, Ting Liu
Harbin Institute of Technology, Harbin, China
{dytang, qinb, tliu}@ir.hit.edu.cn

Abstract

We introduce a deep memory network for aspect level sentiment classification. Unlike feature-based SVM and sequential neural models such as LSTM, this approach explicitly captures the importance of each context word when inferring the sentiment polarity of an aspect. Such importance degree and text representation are calculated with multiple computational layers, each of which is a neural attention model over an external memory. Experiments on laptop and restaurant datasets demonstrate that our approach performs comparable to state-of-art feature based SVM system, and substantially better than LSTM and attention-based LSTM architectures. On both datasets we show that multiple computational layers could improve the performance. Moreover, our approach is also fast. The deep memory network with 9 layers is 15 times faster than LSTM with a CPU implementation.

1 Introduction

Aspect level sentiment classification is a fundamental task in the field of sentiment analysis (Pang and Lee, 2008; Liu, 2012; Pontiki et al., 2014). Given a sentence and an aspect occurring in the sentence, this task aims at inferring the sentiment polarity (e.g. positive, negative, neutral) of the aspect. For example, in sentence "*great food but the service was dreadful!*", the sentiment polarity of aspect "*food*" is positive while the polarity of aspect "*service*" is negative. Researchers typically use machine learning algorithms and build sentiment classifier in a supervised manner. Representative approaches in literature include feature based Support Vector Machine (Kiritchenko et al., 2014; Wagner et al., 2014) and neural network models (Dong et al., 2014; Lakkaraju et al., 2014; Vo and Zhang, 2015; Nguyen and Shirai, 2015; Tang et al., 2015a). Neural models are of growing interest for their capacity to learn text representation from data without careful engineering of features, and to capture semantic relations between aspect and context words in a more scalable way than feature based SVM.

Despite these advantages, conventional neural models like long short-term memory (LSTM) (Tang et al., 2015a) capture context information in an implicit way, and are incapable of explicitly exhibiting important context clues of an aspect. We believe that only some subset of context words are needed to infer the sentiment towards an aspect. For example, in sentence "*great food but the service was dreadful!*", "*dreadful*" is an important clue for the aspect "*service*" but "*great*" is not needed. Standard LSTM works in a sequential way and manipulates each context word with the same operation, so that it cannot explicitly reveal the importance of each context word. A desirable solution should be capable of explicitly capturing the importance of context words and using that information to build up features for the sentence after given an aspect word. Furthermore, a human asked to do this task will selectively focus on parts of the contexts, and acquire information where it is needed to build up an internal representation towards an aspect in his/her mind.

* Corresponding author.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 214–224,
Austin, Texas, November 1-5, 2016.

In pursuit of this goal, we develop deep memory network for aspect level sentiment classification, which is inspired by the recent success of computational models with attention mechanism and explicit memory (Graves et al., 2014; Bahdanau et al., 2015; Sukhbaatar et al., 2015). Our approach is data-driven, computationally efficient and does not rely on syntactic parser or sentiment lexicon. The approach consists of multiple computational layers with shared parameters. Each layer is a content- and location- based attention model, which first learns the importance/weight of each context word and then utilizes this information to calculate continuous text representation. The text representation in the last layer is regarded as the feature for sentiment classification. As every component is differentiable, the entire model could be efficiently trained end-to-end with gradient descent, where the loss function is the cross-entropy error of sentiment classification.

We apply the proposed approach to laptop and restaurant datasets from SemEval 2014 (Pontiki et al., 2014). Experimental results show that our approach performs comparable to a top system using feature-based SVM (Kiritchenko et al., 2014). On both datasets, our approach outperforms both LSTM and attention-based LSTM models (Tang et al., 2015a) in terms of classification accuracy and running speed. Lastly, we show that using multiple computational layers over external memory could achieve improved performance.

2 Background: Memory Network

Our approach is inspired by the recent success of memory network in question answering (Weston et al., 2014; Sukhbaatar et al., 2015). We describe the background on memory network in this part.

Memory network is a general machine learning framework introduced by Weston et al. (2014). Its central idea is inference with a long-term memory component, which could be read, written to, and jointly learned with the goal of using it for prediction. Formally, a memory network consists of a memory m and four components I, G, O and R, where m is an array of objects such as an array of vectors. Among these four components, I converts input to internal feature representation, G updates old memories with new input, O generates an output representation given a new input and the current memory state, R outputs a response based on the output representation.

Let us take question answering as an example to explain the work flow of memory network. Given a list of sentences and a question, the task aims to find evidences from these sentences and generate an answer, e.g. a word. During inference, I component reads one sentence s_i at a time and encodes it into a vector representation. Then G component updates a piece of memory m_i based on current sentence representation. After all sentences are processed, we get a memory matrix m which stores the semantics of these sentences, each row representing a sentence. Given a question q, memory network encodes it into vector representation e_q, and then O component uses e_q to select question related evidences from memory m and generates an output vector o. Finally, R component takes o as the input and outputs the final response. It is worth noting that O component could consist of one or more computational layers (hops). The intuition of utilizing multiple hops is that more abstractive evidences could be found based on previously extracted evidences. Sukhbaatar et al. (2015) demonstrate that multiple hops could uncover more abstractive evidences than single hop, and could yield improved results on question answering and language modeling.

3 Deep Memory Network for Aspect Level Sentiment Classification

In this section, we describe the deep memory network approach for aspect level sentiment classification. We first give the task definition. Afterwards, we describe an overview of the approach before presenting the content- and location- based attention models in each computational layer. Lastly, we describe the use of this approach for aspect level sentiment classification.

3.1 Task Definition and Notation

Given a sentence $s = \{w_1, w_2, ..., w_i, ...w_n\}$ consisting of n words and an aspect word w_i [1] occurring in sentence s, aspect level sentiment classification aims at determining the sentiment polarity of

[1] In practice, an aspect might be a multi word expression such as "*battery life*". For simplicity we still consider aspect as a single word in this definition.

sentence s towards the aspect w_i. For example, the sentiment polarity of sentence "*great food but the service was dreadful!*" towards aspect "*food*" is positive, while the polarity towards aspect "*service*" is negative. When dealing with a text corpus, we map each word into a low dimensional, continuous and real-valued vector, also known as word embedding (Mikolov et al., 2013; Pennington et al., 2014). All the word vectors are stacked in a word embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where d is the dimension of word vector and $|V|$ is vocabulary size. The word embedding of w_i is notated as $e_i \in \mathbb{R}^{d \times 1}$, which is a column in the embedding matrix L.

3.2 An Overview of the Approach

We present an overview of the deep memory network for aspect level sentiment classification.

Given a sentence $s = \{w_1, w_2, ..., w_i, ...w_n\}$ and the aspect word w_i, we map each word into its embedding vector. These word vectors are separated into two parts, aspect representation and context representation. If aspect is a single word like "*food*" or "*service*", aspect representation is the embedding of aspect word. For the case where aspect is multi word expression like "*battery life*", aspect representation is an average of its constituting word vectors (Sun et al., 2015). To simplify the interpretation, we consider aspect as a single word w_i. Context word vectors $\{e_1, e_2 ... e_{i-1}, e_{i+1} ... e_n\}$ are stacked and regarded as the external memory $m \in \mathbb{R}^{d \times (n-1)}$, where n is the sentence length.

An illustration of our approach is given in Figure 1, which is inspired by the use of memory network in question answering (Sukhbaatar et al., 2015). Our approach consists of multiple computational layers (hops), each of which contains an attention layer and a linear layer. In the first computational layer (hop 1), we regard aspect vector as the input to adaptively select important evidences from memory m through attention layer. The output of attention layer and the linear transformation of aspect vector[2] are summed and the result is considered as the input of next layer (hop 2). In a similar way, we stack multiple hops and run these steps multiple times, so that more abstractive evidences could be selected from the external memory m. The output vector in last hop is considered as the representation of sentence with regard to the aspect, and is further used as the feature for aspect level sentiment classification.

[2]In preliminary experiments, we tried directly using aspect vector without a linear transformation, and found that adding a linear layer works slightly better.

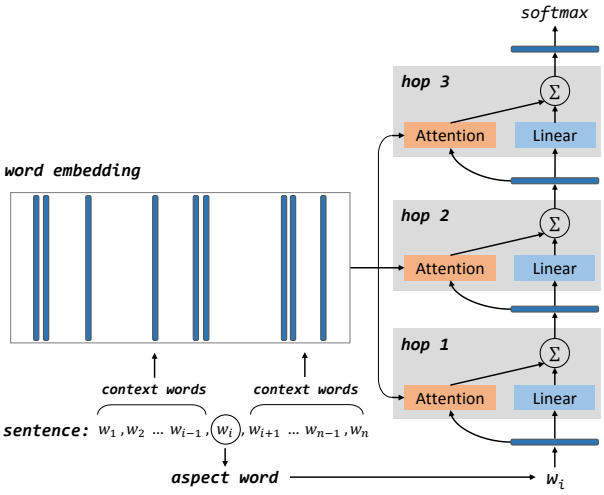


Figure 1: An illustration of our deep memory network with three computational layers (hops) for aspect level sentiment classification.

It is helpful to note that the parameters of attention and linear layers are shared in different hops. Therefore, the model with one layer and the model with nine layers have the same number of parameters.

3.3 Content Attention

We describe our attention model in this part. The basic idea of attention mechanism is that it assigns a weight/importance to each lower position when computing an upper level representation (Bahdanau et al., 2015). In this work, we use attention model to compute the representation of a sentence with regard to an aspect. The intuition is that context words do not contribute equally to the semantic meaning of a sentence. Furthermore, the importance of a word should be different if we focus on different aspect. Let us again take the example of "*great food but the service was dreadful!*". The context word "*great*" is more important than "*dreadful*" for aspect "*food*". On the contrary, "*dreadful*" is more important than "*great*" for aspect "*service*".

Taking an external memory $m \in \mathbb{R}^{d \times k}$ and an aspect vector $v_{aspect} \in \mathbb{R}^{d \times 1}$ as input, the attention model outputs a continuous vector $vec \in \mathbb{R}^{d \times 1}$. The output vector is computed as a weighted sum of each

piece of memory in m, namely

$$vec = \sum_{i=1}^{k} \alpha_i m_i \quad (1)$$

where k is the memory size, $\alpha_i \in [0, 1]$ is the weight of m_i and $\sum_i \alpha_i = 1$. We implement a neural network based attention model. For each piece of memory m_i, we use a feed forward neural network to compute its semantic relatedness with the aspect. The scoring function is calculated as follows, where $W_{att} \in \mathbb{R}^{1\times 2d}$ and $b_{att} \in \mathbb{R}^{1\times 1}$.

$$g_i = tanh(W_{att}[m_i; v_{aspect}] + b_{att}) \quad (2)$$

After obtaining $\{g_1, g_2, ... g_k\}$, we feed them to a $softmax$ function to calculate the final importance scores $\{\alpha_1, \alpha_2, ... \alpha_k\}$.

$$\alpha_i = \frac{exp(g_i)}{\sum_{j=1}^{k} exp(g_j)} \quad (3)$$

We believe that such an attention model has two advantages. One advantage is that this model could adaptively assign an importance score to each piece of memory m_i according to its semantic relatedness with the aspect. Another advantage is that this attention model is differentiable, so that it could be easily trained together with other components in an end-to-end fashion.

3.4 Location Attention

We have described our neural attention framework and a content-based model in previous subsection. However, the model mentioned above ignores the location information between context word and aspect. Such location information is helpful for an attention model because intuitively a context word closer to the aspect should be more important than a farther one. In this work, we define the location of a context word as its absolute distance with the aspect in the original sentence sequence[3]. On this basis, we study four strategies to encode the location information in the attention model. The details are described below.

[3]The location of a context word could also be measured by its distance to the aspect along a syntactic path. We leave this as a future work as we prefer to developing a purely data-driven approach without using external parsing results.

• Model 1. Following Sukhbaatar et al. (2015), we calculate the memory vector m_i with

$$m_i = e_i \odot v_i \quad (4)$$

where $\odot$ means element-wise multiplication and $v_i \in \mathbb{R}^{d\times 1}$ is a location vector for word w_i. Every element in v_i is calculated as follows,

$$v_i^k = (1 - l_i/n) - (k/d)(1 - 2 \times l_i/n) \quad (5)$$

where n is sentence length, k is the hop number and l_i is the location of w_i.

• Model 2. This is a simplified version of Model 1, using the same location vector v_i for w_i in different hops. Location vector v_i is calculated as follows.

$$v_i = 1 - l_i/n \quad (6)$$

• Model 3. We regard location vector v_i as a parameter and compute a piece of memory with vector addition, namely

$$m_i = e_i + v_i \quad (7)$$

All the position vectors are stacked in a position embedding matrix, which is jointly learned with gradient descent.

• Model 4. Location vectors are also regarded as parameters. Different from Model 3, location representations are regarded as neural gates to control how many percent of word semantics is written into the memory. We feed location vector v_i to a sigmoid function σ, and calculate m_i with element-wise multiplication:

$$m_i = e_i \odot \sigma(v_i) \quad (8)$$

3.5 The Need for Multiple Hops

It is widely accepted that computational models that are composed of multiple processing layers have the ability to learn representations of data with multiple levels of abstraction (LeCun et al., 2015). In this work, the attention layer in one layer is essentially a weighted average compositional function, which is not powerful enough to handle the sophisticated computationality like negation, intensification and contrary in language. Multiple computational layers allow the deep memory network to learn representations of text with multiple levels of abstraction. Each layer/hop retrieves important context words,

and transforms the representation at previous level into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions of sentence representation towards an aspect can be learned.

3.6 Aspect Level Sentiment Classification

We regard the output vector in last hop as the feature, and feed it to a $softmax$ layer for aspect level sentiment classification. The model is trained in a supervised manner by minimizing the cross entropy error of sentiment classification, whose loss function is given below, where T means all training instances, C is the collection of sentiment categories, (s, a) means a sentence-aspect pair.

$$loss = -\sum_{(s,a)\in T}\sum_{c\in C} P_c^g(s,a)\cdot log(P_c(s,a)) \quad (9)$$

$P_c(s,a)$ is the probability of predicting (s,a) as category c produced by our system. $P_c^g(s,a)$ is 1 or 0, indicating whether the correct answer is c. We use back propagation to calculate the gradients of all the parameters, and update them with stochastic gradient descent. We clamp the word embeddings with 300-dimensional Glove vectors (Pennington et al., 2014), which is trained from web data and the vocabulary size is 1.9M[4]. We randomize other parameters with uniform distribution $U(-0.01, 0.01)$, and set the learning rate as 0.01.

4 Experiment

We describe experimental settings and report empirical results in this section.

4.1 Experimental Setting

We conduct experiments on two datasets from SemEval 2014 (Pontiki et al., 2014), one from laptop domain and another from restaurant domain. Statistics of the datasets are given in Table 1. It is worth noting that the original dataset contains the fourth category - conflict, which means that a sentence expresses both positive and negative opinion towards an aspect. We remove conflict category as the number of instances is very tiny, incorporating which will make the dataset extremely unbalanced. Evaluation metric is classification accuracy.

Dataset	Pos.	Neg.	Neu.
Laptop-Train	994	870	464
Laptop-Test	341	128	169
Restaurant-Train	2164	807	637
Restaurant-Test	728	196	196

Table 1: Statistics of the datasets.

4.2 Comparison to Other Methods

We compare with the following baseline methods on both datasets.

(1) **Majority** is a basic baseline method, which assigns the majority sentiment label in training set to each instance in the test set.

(2) **Feature-based SVM** performs state-of-the-art on aspect level sentiment classification. We compare with a top system using ngram features, parse features and lexicon features (Kiritchenko et al., 2014).

(3) We compare with three LSTM models (Tang et al., 2015a)). In **LSTM**, a LSTM based recurrent model is applied from the start to the end of a sentence, and the last hidden vector is used as the sentence representation. **TDLSTM** extends LSTM by taking into account of the aspect, and uses two LSTM networks, a forward one and a backward one, towards the aspect. **TDLSTM+ATT** extends TDLSTM by incorporating an attention mechanism (Bahdanau et al., 2015) over the hidden vectors. We use the same Glove word vectors for fair comparison.

(4) We also implement **ContextAVG**, a simplistic version of our approach. Context word vectors are averaged and the result is added to the aspect vector. The output is fed to a $softmax$ function.

Experimental results are given in Table 2. Our approach using only content attention is abbreviated to MemNet (k), where k is the number of hops. We can find that feature-based SVM is an extremely strong performer and substantially outperforms other baseline methods, which demonstrates the importance of a powerful feature representation for aspect level sentiment classification. Among three recurrent models, TDLSTM performs better than LSTM, which indicates that taking into account of the aspect information is helpful. This is reasonable as the sentiment polarity of a sentence towards different as-

[4]Available at: http://nlp.stanford.edu/projects/glove/.

	Laptop	Restaurant
Majority	53.45	65.00
Feature+SVM	**72.10**	**80.89**
LSTM	66.45	74.28
TDLSTM	68.13	75.63
TDLSTM+ATT	66.24	74.31
ContextAVG	61.22	71.33
MemNet (1)	67.66	76.10
MemNet (2)	71.14	78.61
MemNet (3)	71.74	79.06
MemNet (4)	72.21	79.87
MemNet (5)	71.89	80.14
MemNet (6)	72.21	80.05
MemNet (7)	**72.37**	80.32
MemNet (8)	72.05	80.14
MemNet (9)	72.21	**80.95**

Table 2: Classification accuracy of different methods on laptop and restaurant datasets. Best scores in each group are in bold.

pects (e.g. "*food*" and "*service*") might be different. It is somewhat disappointing that incorporating attention model over TDLSTM does not bring any improvement. We consider that each hidden vector of TDLSTM encodes the semantics of word sequence until the current position. Therefore, the model of TDLSTM+ATT actually selects such mixed semantics of word sequence, which is weird and not an intuitive way to selectively focus on parts of contexts. Different from TDLSTM+ATT, the proposed memory network approach removes the recurrent calculator over word sequence and directly apply attention mechanism on context word representations.

We can also find that the performance of ContextAVG is very poor, which means that assigning the same weight/importance to all the context words is not an effective way. Among all our models from single hop to nine hops, we can observe that using more computational layers could generally lead to better performance, especially when the number of hops is less than six. The best performances are achieved when the model contains seven and nine hops, respectively. On both datasets, the proposed approach could obtain comparable accuracy compared to the state-of-art feature-based SVM system.

4.3 Runtime Analysis

We study the runtime of recurrent neural models and the proposed deep memory network approach with different hops. We implement all these approaches based on the same neural network infrastructure, use the same 300-dimensional Glove word vectors, and run them on the same CPU server.

Method	Time cost
LSTM	417
TDLSTM	490
TDLSTM + ATT	520
MemNet (1)	3
MemNet (2)	7
MemNet (3)	9
MemNet (4)	15
MemNet (5)	20
MemNet (6)	24
MemNet (7)	26
MemNet (8)	27
MemNet (9)	29

Table 3: Runtime (seconds) of each training epoch on the restaurant dataset.

The training time of each iteration on the restaurant dataset is given in Table 3. We can find that LSTM based recurrent models are indeed computationally expensive, which is caused by the complex operations in each LSTM unit along the word sequence. Instead, the memory network approach is simpler and evidently faster because it does not need recurrent calculators of sequence length. Our approach with nine hops is almost 15 times faster than the basic LSTM model.

4.4 Effects of Location Attention

As described in Section 3.4, we explore four strategies to integrate location information into the attention model. We incorporate each of them separately into the basic content-based attention model. It is helpful to restate that the difference between four location-based attention models lies in the usage of location vectors for context words. In Model 1 and Model 2, the values of location vectors are fixed and calculated in a heuristic way. In Model 3 and Model 4, location vectors are also regarded as the parameters and jointly learned along with other parameters in the deep memory network.

(a) Aspect: *service*, Answer: -1, Prediction: -1

	hop 1	hop 2	hop 3	hop 4	hop 5
great	0.20	0.15	0.14	0.13	0.23
food	0.11	0.07	0.08	0.12	0.06
but	0.20	0.10	0.10	0.12	0.13
the	0.03	0.07	0.08	0.12	0.06
was	0.08	0.07	0.08	0.12	0.06
dreadful	0.20	0.45	0.45	0.28	0.40
!	0.19	0.08	0.08	0.12	0.07

(b) Aspect: *food*, Answer: +1, Prediction: -1

	hop 1	hop 2	hop 3	hop 4	hop 5
great	0.22	0.12	0.14	0.12	0.20
but	0.21	0.11	0.10	0.11	0.12
the	0.03	0.11	0.08	0.11	0.06
service	0.11	0.11	0.08	0.11	0.06
was	0.04	0.11	0.08	0.11	0.06
dreadful	0.22	0.32	0.45	0.32	0.43
!	0.16	0.11	0.08	0.11	0.07

Table 4: Examples of attention weights in different hops for aspect level sentiment classification. The model only uses content attention. The hop columns show the weights of context words in each hop, indicated by values and gray color. This example shows the results of sentence "*great food but the service was dreadful!*" with "*food*" and "*service*" as the aspects.

(a) Aspect: *service*, Answer: -1, Prediction: -1

	hop 1	hop 2	hop 3	hop 4	hop 5
great	0.08	0.10	0.10	0.09	0.09
food	0.08	0.07	0.07	0.07	0.07
but	0.10	0.15	0.16	0.13	0.11
the	0.07	0.07	0.07	0.07	0.07
was	0.07	0.07	0.07	0.07	0.07
dreadful	0.52	0.48	0.48	0.50	0.52
!	0.07	0.07	0.07	0.07	0.07

(b) Aspect: *food*, Answer: +1, Prediction: +1

	hop 1	hop 2	hop 3	hop 4	hop 5
great	0.31	0.26	0.32	0.28	0.32
but	0.14	0.18	0.15	0.18	0.15
the	0.08	0.05	0.08	0.05	0.07
service	0.09	0.09	0.09	0.08	0.09
was	0.09	0.08	0.09	0.08	0.08
dreadful	0.18	0.21	0.18	0.22	0.19
!	0.11	0.12	0.10	0.11	0.10

Table 5: Examples of attention weights in different hops for aspect level sentiment classification. The model also takes into account of the location information (Model 2). This example is as same as the one we use in Table 4.

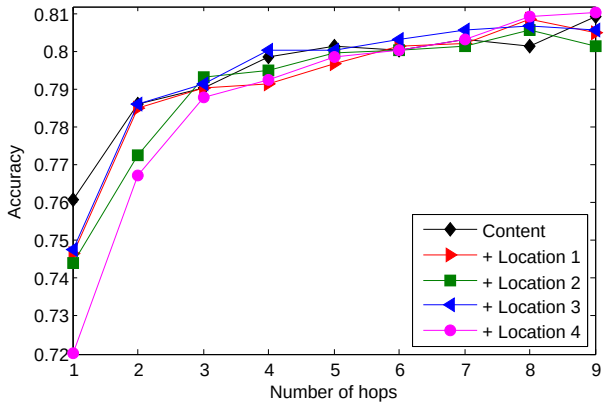


Figure 2: Classification accuracy of different attention models on the restaurant dataset.

Figure 2 shows the classification accuracy of each attention model on the restaurant dataset. We can find that using multiple computational layers could consistently improve the classification accuracy in all these models. All these models perform comparably when the number of hops is larger than five. Among these four location-based models, we prefer Model 2 as it is intuitive and has less computation cost without loss of accuracy. We also find that Model 4 is very sensitive to the choice of neural gate. Its classification accuracy decreases by almost 5 percentage when the $sigmoid$ operation over location vector is removed.

4.5 Visualize Attention Models

We visualize the attention weight of each context word to get a better understanding of the deep memory network approach. The results of context-based model and location-based model (Model 2) are given in Table 4 and Table 5, respectively.

From Table 4(a), we can find that in the first hop the context words "*great*", "*but*" and "*dreadful*" contribute equally to the aspect "*service*". While after the second hop, the weight of "*dreadful*" increases and finally the model correctly predict the polarity towards "*service*" as negative. This case shows the effects of multiple hops. However, in Table 4(b), the content-based model also gives a larger weight to "*dreadful*" when the target we focus on is "*food*". As a result, the model incorrectly predicts the polarity towards "*food*" as negative. This phenomenon might be caused by the neglect of location information. From Table 5(b), we can find that the weight of "*great*" is increased when the location of context word is considered. Accordingly, Model 2 predict-

s the correct sentiment label towards "*food*". We believe that location-enhanced model captures both content and location information. For instance, in Table 5(a) the closest context words of the aspect "*service*" are "*the*" and "*was*", while "*dreadful*" has the largest weight.

4.6 Error Analysis

We carry out an error analysis of our location enhanced model (Model 2) on the restaurant dataset, and find that most of the errors could be summarized as follows. The first factor is non-compositional sentiment expression. This model regards single context word as the basic computational unit and cannot handle this situation. An example is "*dessert was also to die for!*", where the aspect is underlined. The sentiment expression is "*die for*", whose meaning could not be composed from its constituents "*die*" and "*for*". The second factor is complex aspect expression consisting of many words, such as "*ask for the round corner table next to the large window.*" This model represents an aspect expression by averaging its constituting word vectors, which could not well handle this situation. The third factor is sentimental relation between context words such as negation, comparison and condition. An example is "*but dinner here is never disappointing, even if the prices are a bit over the top*". We believe that this is caused by the weakness of weighted average compositional function in each hop. There are also cases when comparative opinions are expressed such as "*i 've had better japanese food at a mall food court*".

5 Related Work

This work is connected to three research areas in natural language processing. We briefly describe related studies in each area.

5.1 Aspect Level Sentiment Classification

Aspect level sentiment classification is a fine-grained classification task in sentiment analysis, which aims at identifying the sentiment polarity of a sentence expressed towards an aspect (Pontiki et al., 2014). Most existing works use machine learning algorithms, and build sentiment classifier from sentences with manually annotated polarity labels. One of the most successful approaches in literature is feature based SVM. Experts could design effective feature templates and make use of external resources like parser and sentiment lexicons (Kiritchenko et al., 2014; Wagner et al., 2014). In recent years, neural network approaches (Dong et al., 2014; Lakkaraju et al., 2014; Nguyen and Shirai, 2015; Tang et al., 2015a) are of growing attention for their capacity to learn powerful text representation from data. However, these neural models (e.g. LSTM) are computationally expensive, and could not explicitly reveal the importance of context evidences with regard to an aspect. Instead, we develop simple and fast approach that explicitly encodes the context importance towards a given aspect. It is worth noting that the task we focus on differs from fine-grained opinion extraction, which assigns each word a tag (e.g. B,I,O) to indicate whether it is an aspect/sentiment word (Choi and Cardie, 2010; Irsoy and Cardie, 2014; Liu et al., 2015). The aspect word in this work is given as a part of the input.

5.2 Compositionality in Vector Space

In NLP community, compositionality means that the meaning of a composed expression (e.g. a phrase/sentence/document) comes from the meanings of its constituents (Frege, 1892). Mitchell and Lapata (2010) exploits a variety of addition and multiplication functions to calculate phrase vector. Yessenalina and Cardie (2011) use matrix multiplication as compositional function to compute vectors for longer phrases. To compute sentence representation, researchers develop denoising autoencoder (Glorot et al., 2011), convolutional neural network (Kalchbrenner et al., 2014; Kim, 2014; Yin and Schütze, 2015), sequence based recurrent neural models (Sutskever et al., 2014; Kiros et al., 2015; Li et al., 2015b) and tree-structured neural networks (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015). Several recent studies calculate continuous representation for documents with neural networks (Le and Mikolov, 2014; Bhatia et al., 2015; Li et al., 2015a; Tang et al., 2015b; Yang et al., 2016).

5.3 Attention and Memory Networks

Recently, there is a resurgence in computational models with attention mechanism and explicit memory to learn representations of texts (Graves et al., 2014; Weston et al., 2014; Sukhbaatar et al., 2015;

Bahdanau et al., 2015). In this line of research, memory is encoded as a continuous representation and operations on memory (e.g. reading and writing) are typically implemented with neural networks. Attention mechanism could be viewed as a compositional function, where lower level representations are regarded as the memory, and the function is to choose "where to look" by assigning a weight/importance to each lower position when computing an upper level representation. Such attention based approaches have achieved promising performances on a variety of NLP tasks (Luong et al., 2015; Kumar et al., 2015; Rush et al., 2015).

6 Conclusion

We develop deep memory networks that capture importances of context words for aspect level sentiment classification. Compared with recurrent neural models like LSTM, this approach is simpler and faster. Empirical results on two datasets verify that the proposed approach performs comparable to state-of-the-art feature based SVM system, and substantively better than LSTM architectures. We implement different attention strategies and show that leveraging both content and location information could learn better context weight and text representation. We also demonstrate that using multiple computational layers in memory network could obtain improved performance. Our potential future plans are incorporating sentence structure like parsing results into the deep memory network.

Acknowledgments

We would especially want to thank Xiaodan Zhu for running their system on our setup. We greatly thank Yaming Sun for tremendously helpful discussions. We also thank the anonymous reviewers for their valuable comments. This work was supported by the National High Technology Development 863 Program of China (No. 2015AA015407), National Natural Science Foundation of China (No. 61632011 and No.61273321).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274. Association for Computational Linguistics.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Gottlob Frege. 1892. On sense and reference. *Ludlow (1997)*, pages 563–584.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan

Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31nd International Conference on Machine Learning*, pages 1188–1196.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015a. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1106–1115.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015b. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

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

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

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

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

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

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

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1333–1339.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015a. Target-Dependent Sentiment Classification with Long Short Term Memory. *ArXiv preprint arXiv:1512.01100*.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Document modeling with gated recurrent neural network for sentiment classification. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic

features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1347–1353.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 172–182.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 204–214.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1604–1612.

Lifelong-RL: Lifelong Relaxation Labeling for Separating Entities and Aspects in Opinion Targets

Lei Shu[1], **Bing Liu**[1], **Hu Xu**[1], **Annice Kim**[2]
[1]Department of Computer Science, University of Illinois at Chicago, USA
[2]Center for Health Policy Science and Tobacco Research, RTI International, USA
[1]{lshu3, liub, hxu48}@uic.edu, [2]akim@rti.org

Abstract

It is well-known that opinions have targets. Extracting such targets is an important problem of opinion mining because without knowing the target of an opinion, the opinion is of limited use. So far many algorithms have been proposed to extract opinion targets. However, an opinion target can be an entity or an aspect (part or attribute) of an entity. An opinion about an entity is an opinion about the entity as a whole, while an opinion about an aspect is just an opinion about that specific attribute or aspect of an entity. Thus, opinion targets should be separated into entities and aspects before use because they represent very different things about opinions. This paper proposes a novel algorithm, called *Lifelong-RL*, to solve the problem based on *lifelong machine learning* and *relaxation labeling*. Extensive experiments show that the proposed algorithm Lifelong-RL outperforms baseline methods markedly.

1 Introduction

A core problem of opinion mining or sentiment analysis is to identify each opinion/sentiment target and to classify the opinion/sentiment polarity on the target (Liu, 2012). For example, in a review sentence for a car, one wrote "Although the engine is slightly weak, this car is great." The person is positive (opinion polarity) about the car (opinion target) as a whole, but slightly negative (opinion polarity) about the car's engine (opinion target).

Past research has proposed many techniques to extract *opinion targets* (we will just call them *targets* hereafter for simplicity) and also to classify sentiment polarities on the targets. However, a target can be an entity or an aspect (part or attribute) of an entity. "Engine" in the above sentence is just one aspect of the car, while "this car" refers to the whole car. Note that in (Liu, 2012), an entity is called a *general aspect*. For effective opinion mining, we need to classify whether a target is an entity or an aspect because they refer to very different things. One can be positive about the whole entity (car) but negative about some aspects of it (e.g., engine) and vice versa. This paper aims to perform the target classification task, which, to our knowledge, has not been attempted before. Although in supervised extraction one can annotate entities and aspects with separate labels in the training data to build a model to extract them separately, in this paper our goal is to help unsupervised target extraction methods to classify targets. Unsupervised target extraction methods are often preferred because they save the time-consuming data labeling or annotation step for each domain.

Problem Statement: Given a set of opinion targets $T = \{t_1, \ldots, t_n\}$ extracted from an opinion corpus d, we want to classify each target $t_i \in T$ into one of the three classes, *entity*, *aspect*, or *NIL*, which are called class labels. NIL means that the target is neither an entity nor an aspect and is used because target extraction algorithms can make mistakes.

This paper does not propose a new target extraction algorithm. We use an existing unsupervised method, called *Double Propagation* (DP) (Qiu et al., 2011), for extraction. We only focus on target classification after the targets have been extracted. Note that an entity here can be a named entity, a prod-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 225–235,
Austin, Texas, November 1-5, 2016.

uct category, or an abstract product (e.g., "this machine" and "this product"). An named entity can be the name of a brand, a model, or a manufacturer. An aspect is a part or attribute of an entity, e.g., "battery" and "price" of the entity "camera".

Since our entities not just include the traditional named entities (e.g., "Microsoft" and "Google") but also other expressions that refer to such entities, traditional named entity recognition algorithms are not sufficient. Pronouns such as "it," "they," etc., are not considered in this paper as co-reference resolution is out of the scope of this work.

We solve this problem in an unsupervised manner so that there is no need for labor-intensive manual labeling of the training data. One key observation of the problem is that although entities and aspects are different, they are closely related because aspects are parts or attributes of entities and they often have syntactic relationships in a sentence, e.g., "This phone's screen is super." Thus it is natural to solve the problem using a relational learning method. We employ the graph labeling algorithm, *Relaxation Labeling* (RL) (Hummel and Zucker, 1983), which performs unsupervised belief propagation on a graph. In our case, each target extracted from the given corpus d forms a graph node and each relation identified in d between two targets forms an edge. With some initial probability assignments, RL can assign each target node the most probable class label. Although some other graph labeling methods can be applied as well, the key issue here is that just using a propagation method in isolation is far from sufficient due to lack of information from the given corpus, which we detail in Section 5. We then employ *Lifelong Machine Learning* (LML) (Thrun, 1998; Chen and Liu, 2014b) to make a major improvement.

LML works as follows: The learner has performed a number learning tasks in the past and has retained the knowledge gained so far. In the new/current task, it makes use of the past knowledge to help current learning and problem solving. Since RL is unsupervised, we can assume that the system has performed the same task on reviews of a large number of products/domains (or corpora). It has also saved all the graphs and classification results from those past domains in a *Knowledge Base* (KB). It then exploits this past knowledge to help classification in the current task/domain. We call this combined approach of relaxation labeling and LML *Lifelong-RL*. The approach is effective because there is a significant amount of sharing of targets and target relations across domains.

LML is different from the classic learning paradigm (supervised or unsupervised) because classic learning has no memory. It basically runs a learning algorithm on a given data in isolation without considering any past learned knowledge (Silver et al., 2013). LML aims to mimic human learning, which always retains the learned knowledge from the past and uses it to help future learning.

Our experimental results show that the proposed Lifelong-RL system is highly promising. The paradigm of LML helps improve the classification results greatly.

2 Related Work

Although many target extraction methods exist (Hu and Liu, 2004; Zhuang et al., 2006; Ku et al., 2006; Wang and Wang, 2008; Wu et al., 2009; Lin and He, 2009; Zhang et al., 2010; Mei et al., 2007; Li et al., 2010; Brody and Elhadad, 2010; Wang et al., 2010; Mukherjee and Liu, 2012; Fang and Huang, 2012; Zhou et al., 2013; Liu et al., 2013; Poria et al., 2014), we are not aware of any attempt to solve the proposed problem. As mentioned in the introduction, although in supervised target extraction, one can annotate entities and aspects with different labels, supervised methods need manually labeled training data, which is time-consuming and labor-intensive to produce (Jakob and Gurevych, 2010; Choi and Cardie, 2010; Mitchell et al., 2013). Note that relaxation labeling was used for sentiment classification in (Popescu and Etzioni, 2007), but not for target classification. More details of opinion mining can be found in (Liu, 2012; Pang and Lee, 2008).

Our work is related to transfer learning (Pan and Yang, 2010), which uses the source domain labeled data to help target domain learning, which has little or no labeled data. Our work is not just using a source domain to help a target domain. It is a continuous and cumulative learning process. Each new task can make use of the knowledge learned from all past tasks. Knowledge learned from the new task can also help improve learning of any past task. Transfer learning is not continuous, does not

accumulate knowledge over time and cannot improve learning in the source domain. Our work is also related to multi-task learning (Caruana, 1997), which jointly optimizes a set of related learning tasks. Clearly, multi-task learning is different as we learn and save information which is more realistic when a large number of tasks are involved.

Our work is most related to Lifelong Machine Learning (LML). Traditional LML focuses on supervised learning (Thrun, 1998; Ruvolo and Eaton, 2013; Chen et al., 2015). Recent work used LML in topic modeling (Chen and Liu, 2014a), which is unsupervised. Basically, they used topics generated from past domains to help current domain model inference. However, they are just for aspect extraction. So is the method in (Liu et al., 2016). They do not solve our problem. Their LML methods are also different from ours as we use a graph and results obtained in the past domains to augment the current task/domain graph to solve the problem.

3 Lifelong-RL: The General Framework

In this section, we present the proposed general framework of lifelong relaxation labeling (Lifelong-RL). We first give an overview of the relaxation labeling algorithm, which forms the base. We then incorporate it with the LML capability. The next two sections detail how this general framework is applied to our proposed task of separating entities and aspects in opinion targets.

3.1 Relaxation Labeling

Relaxation Labeling (RL) is an unsupervised graph-based label propagation algorithm that works iteratively. The graph consists of nodes and edges. Each edge represents a binary relationship between two nodes. Each node t_i in the graph is associated with a multinomial distribution $P(L(t_i))$ ($L(t_i)$ being the label of t_i) on a label set Y. Each edge is associated with two conditional probability distributions $P(L(t_i)|L(t_j))$ and $P(L(t_j)|L(t_i))$, where $P(L(t_i)|L(t_j))$ represents how the label $L(t_j)$ influences the label $L(t_i)$ and vice versa. The neighbors $Ne(t_i)$ of a node t_i are associated with a weight distribution $w(t_j|t_i)$ with $\sum_{t_j \in Ne(t_i)} w(t_j|t_i) = 1$.

Given the initial values of these quantities as inputs, RL iteratively updates the label distribution of each node until convergence. Initially, we have $P^0(L(t_i))$. Let $\Delta P^{r+1}(L(t_i))$ be the change of $P(L(t_i))$ at iteration $r+1$. Given $P^r(L(t_i))$ at iteration r, $\Delta P^{r+1}(L(t_i))$ is computed by:

$$\Delta P^{r+1}(L(t_i)) = \sum_{t_j \in Ne(t_i)} (w(t_j|t_i) \cdot \sum_{y \in Y} (P(L(t_i)|L(t_j) = y) P^r(L(t_j) = y))) \quad (1)$$

Then, the updated label distribution for iteration $r+1$, $P^{r+1}(L(t_i))$, is computed as follows:

$$P^{r+1}(L(t_i)) = \frac{P^r(L(t_i))(1+\Delta P^{r+1}(L(t_i)))}{\sum_{y \in Y} P^r(L(t_i)=y)(1+\Delta P^{r+1}(L(t_i)=y))} \quad (2)$$

Once RL ends, the final label of node t_i is its highest probable label: $L(t_i) = \underset{y \in Y}{\operatorname{argmax}}(P(L(t_i) = y))$.

Note that $P(L(t_i)|L(t_j))$ and $w(t_j|t_i)$ are not updated in each RL iteration but only $P(L(t_i))$ is. $P(L(t_i)|L(t_j))$, $w(t_j|t_i)$ and $P^0(L(t_i))$ are provided by the user or computed based on the application context. RL uses these values as input and iteratively updates $P(L(t_i))$ based on Equations (1) and (2) until convergence. Next we discuss how to incorporate LML in RL.

3.2 Lifelong Relaxation Labeling

For LML, it is assumed that at any time step, the system has worked on u past domain corpora $D = \{d_1, \ldots, d_u\}$. For each past domain corpus $d \in D$, the same Lifelong-RL algorithm was applied and its results were saved in the Knowledge Base (KB). Then the algorithm can borrow some useful prior/past knowledge in the KB to help RL in the new/current domain d_{u+1}. Once the results of the current domain are produced, they are also added to the KB for future use.

We now detail the specific types of information or knowledge that can be obtained from the past domains to help RL in the future, which should thus be stored in the KB.

1. *Prior edges*: In many applications, the graph is not given. Instead, it has to be constructed based on the data from the new task/domain data d_{u+1}. However, due to the limited data in d_{u+1}, some edges between nodes that should be present are not extracted from the data. But such edges between the nodes may exist in

some past domains. Then, those edges and their associated probabilities can be borrowed.

2. *Prior labels*: Some nodes in the current new domain may also exist in some past domains. Their labels in the past domains are very likely to be the same as those in the current domain. Then, those prior labels can give us a better idea about the initial label probability distributions of the nodes in the current domain d_{u+1}.

To leverage those edges and labels from the past domains, the system needs to ensure that they are likely to be correct and applicable to the current task domain. This is a challenge problem. In the next two sections, we detail how to ensure these to a large extent in our application context along with how to compute those initial probabilities.

4 Initialization of Relaxation Labeling

We now discuss how the proposed Lifelong-RL general framework is applied to solve our problem. In our case, each node in the graph is an extracted target $t_i \in T$, and each edge represents a binary relationship between two targets. T is the given set of all opinion targets extracted by an extraction algorithm from a review dataset/corpus d. The label set for each target is $Y = \{\text{entity}, \text{aspect}, \text{NIL}\}$. In this section, we describe how to use text clues in the corpus d to compute $P(L(t_i)|L(t_j))$, $w(t_j|t_i)$ and $P^0(L(t_i))$. In the next section, we present how these quantities are improved using prior knowledge from the past domains in the LML fashion.

4.1 Text Clues for Initialization

We use two kinds of text clues, called *type modifiers* $M(t)$ and *relation modifiers* M_R to compute the initial label distribution $P(L(t_i))$ and conditional label distribution $P(L(t_i)|L(t_j))$ respectively.

Type Modifier: This has two kinds $M_T = \{m_E, m_A\}$, where m_E and m_A represent entity modifier and aspect modifier respectively. For example, the word "this" as in "this camera is great" indicates that "camera" is probably an entity. Thus, "this" is a type modifier indicating $M(camera) = m_E$. "These" is also a type modifier. Aspect modifier is implicitly assumed when the number of appearances of entity modifiers is less than or equal to a threshold (see Section 4.2).

Relation Modifier: Given two targets, t_i and t_j, we use $M_{t_j}(t_i)$ to denote the relation modifier that the label of target t_i is influenced by the label of target t_j. Relation modifiers are further divided into 3 kinds: $M_R = \{m_c, m_{A|E}, m_{E|A}\}$.

Conjunction modifier m_c: Conjoined items are usually of the same type. For example, in "price and service", "and service" indicates a conjunction modifier for "price" and vice versa.

Entity-aspect modifier $m_{A|E}$: A possessive expression indicates an entity and an aspect relation. For example, in "the camera's battery", "camera" indicates an *entity-aspect modifier* for "battery".

Aspect-entity modifier $m_{E|A}$: Same as above except that "battery" indicates an *aspect-entity modifier* for "camera".

Modifier Extraction: These modifiers are identified from the corpus d using three syntactic rules. "This" and "these" are used to extract type modifier $M(t) = m_E$. $C_{m_E}(t)$ is the occurrence count of that modifier on target t, which is used in determining the initial label distribution in Section 4.2.

Relation modifiers are identified by dependency relations $conj(t_i, t_j)$ and $poss(t_i, t_j)$ using the Stanford Parser (Klein and Manning, 2003). Each occurrence of a relation rule contributes one count of $M_{t_j}(t_i)$ for t_i and one count of $M_{t_i}(t_j)$ for t_j. We use $C_{m_c,t_j}(t_i)$, $C_{m_{A|E},t_j}(t_i)$ and $C_{m_{E|A},t_j}(t_i)$ to denote the count of t_j modifying t_i with conjunction, entity-aspect and aspect-entity modifiers respectively. For example, "price and service" will contribute one count to $C_{m_c,\text{price}}(\text{service})$ and one count to $C_{m_c,\text{service}}(\text{price})$. Similarly, "camera's battery" will contribute one count to $C_{m_{A|E},\text{camera}}(\text{battery})$ and one count to $C_{m_{E|A},\text{battery}}(\text{camera})$.

4.2 Computing Initial Probabilities

The initial label probability distribution of target t is computed based on $C_{m_E}(t)$, i.e.,

$$P^0(L(t)) = \begin{cases} P_{m_E}(L(t)) & \text{if} \quad C_{m_E}(t) > \alpha \\ P_{m_A}(L(t)) & \text{if} \quad C_{m_E}(t) \leq \alpha \end{cases} \tag{3}$$

Here, we have two pre-defined distributions: P_{m_E} and P_{m_A}, which have a higher probability on entity and aspect respectively. The parameter α is a threshold indicating that if the entity modifier rarely occurs, the target is more likely to be an aspect. These

values are set empirically (see Section 6).

Let term $q(M_{t_j}(t_i) = m)$ be the normalized weight on the count for each kind of relation modifier $m \in M_R$:

$$q(M_{t_j}(t_i) = m) = \frac{C_{m,t_j}(t_i)}{C_{t_j}(t_i)} \quad (4)$$

where $C_{t_j}(t_i) = \sum_{m \in M_R} C_{m,t_j}(t_i)$.

The conditional label distribution $P(L(t_i)|L(t_j))$ of t_i given the label of t_j is the weighted sum over the three kinds of relation modifiers:

$$\begin{aligned} &P(L(t_i)|L(t_j)) = \\ &q(M_{t_j}(t_i) = m_c) \cdot P_{m_c}(L(t_i)|L(t_j)) \\ &+q(M_{t_j}(t_i) = m_{A|E}) \cdot P_{m_{A|E}}(L(t_i)|L(t_j)) \\ &+q(M_{t_j}(t_i) = m_{E|A}) \cdot P_{m_{E|A}}(L(t_i)|L(t_j)) \end{aligned} \quad (5)$$

where P_{m_c}, $P_{m_{A|E}}$, and $P_{m_{E|A}}$ are pre-defined conditional distributions. They are filled with values to model the label influence from neighbors and can be found in Section 6.

Finally, target t_i's neighbor weight for target t_j, i.e., $w(t_j|t_i)$, is the ratio of the count of relation modifiers $C_{t_j}(t_i)$ over the total of all t_i's neighbors:

$$w(t_j|t_i) = \frac{C_{t_j}(t_i)}{\sum_{t_{j'} \in Ne(t_i)} C_{t_{j'}}(t_i)} \quad (6)$$

If $C_{t_j}(t_i) = 0$, t_i and t_j has no edge between them.

5 Using Past Knowledge in Lifelong-RL

Due to the fact that the review corpus d_{u+1} in the current task domain may not be very large and that we use high quality syntactic rules to extract relations to build the graph to ensure precision, the number of relations extracted can be small and insufficient to produce a graph that is information rich with accurate initial probabilities. We thus apply LML to help using knowledge learned in the past. The proposed LML process in Lifelong-RL for our task is shown in Figure 1.

Our prior knowledge includes type modifiers, relation modifiers and labels of targets obtained from past domains in D. Each record in the KB is stored as a 9-tuple: $(d, t_i, t_j, M^d(t_i), M^d(t_j), C^d_{m,t_j}(t_i), C^d_{m,t_i}(t_j), L^d(t_i), L^d(t_j))$ where $d \in D$ is a past domain; t_i and t_j are two targets; $M^d(t_i)$, $M^d(t_j)$ are their type

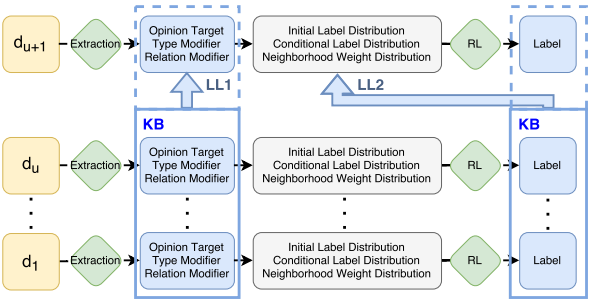


Figure 1: The proposed LML process.

modifiers, $C^d_{m,t_j}(t_i)$ and $C^d_{m,t_i}(t_j)$ are counts for relation modifiers; $L^d(t_i)$ and $L^d(t_j)$ are labels decided by RL. For example, the sentence "This camera's battery is good" forms: $(d, \text{camera}, \text{battery}, m_E, m_A, C_{m_{E|A},battery}(camera) = 1, C_{m_{A|E},camera}(battery) = 1, \text{entity}, \text{aspect})$. It means that in the past domain d, "camera" and "battery" are extracted targets. Since "camera" is followed by "this", its type modifier is m_E. Since "battery" is not identified by an entity modifier, it is m_A. The pattern "camera's battery" contributes one count for both relation modifiers $C_{m_{E|A},battery}(camera)$ and $C_{m_{A|E},camera}(battery)$. RL has labeled "camera" as entity and "battery" as aspect in d.

The next two subsections present how to use the knowledge in the KB to improve the initial assignments for the label distributions, conditional label distributions and neighborhood weight distributions in order to achieve better final labeling/classification results for the current/new domain d_{u+1}.

5.1 Exploiting Relation Modifiers in the KB

If two targets in the current domain corpus have no edge, we can check whether relation modifiers of the same two targets exist in some past domains. If so, we may be able to borrow them. But to ensure suitability, two consistency checks are performed.

Label Consistency Check: Since RL makes mistakes, we need to ensure that relation modifiers in a record in the KB are consistent with target labels in that past domain. For example, "camera's battery" is confirmed by "camera" being labeled as entity and "battery" being labeled as aspect in a past domain $d \in D$. Without this consistency, the record may not be reliable and should be discarded from the KB.

We define an indicator variable $\mathbb{I}^d_{m,t_j}(t_i)$ to ensure that the record r's relation modifier is *consistent*

with the labels of its two targets:

$$\mathbb{I}^d_{m_{A|E},t_j}(t_i) = \begin{cases} 1 & \text{if } C^d_{m_{A|E},t_j}(t_i) > 0 \\ & \text{and } L^d(t_i) = \text{aspect} \\ & \text{and } L^d(t_j) = \text{entity} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

For example, if "camera" is labeled as entity and "battery" is labeled as aspect in the past domain d, we have $\mathbb{I}^d_{m_{A|E},\text{camera}}(\text{battery}) = 1$ and $\mathbb{I}^d_{m_{E|A},\text{battery}}(\text{camera}) = 1$.

Type Consistency Check: Here we ensure the type modifiers for two targets in the current domain d_{u+1} are consistent with these type modifiers in the past domain $d \in D$. This is because an item can be an aspect in one domain but an entity in another. For example, if the current domain is "Cellphone", borrowing the relation "camera's battery" from domain "Camera" can introduce an error because "camera" is an aspect in domain "Cellphone".

Syntactic pattern "this" is a good indicator for this checking. In the "Cellphone" domain, "its camera" or "the camera" are often mentioned but not "this camera". In the "Camera" domain, "this camera" is often mentioned. The type modifier of "camera" in "Cellphone" is m_A, but in "Camera" it is m_E.

Updating Probabilities in Current Domain d_{u+1}: Edges for RL are in the forms of conditional label distribution $P(L(t_i)|L(t_j))$ and neighborhood weight distribution $w(t_j|t_i)$. We now discuss how to use the KB to estimate them more accurately.

Updating Conditional Label Distribution: Equation (5) tells that conditional label distribution $P(L(t_i)|L(t_j))$ is the weighted sum of relation modifiers' label distributions P_{m_c}, $P_{m_{A|E}}$, and $P_{m_{E|A}}$. These 3 label distributions are pre-defined and given in Table 2. They are not changed. Thus, we update conditional label distribution through updating the three relation modifiers' weights $q(M_{t_j}(t_i))$ with the knowledge in the KB. Recall the three relation modifiers are $M_R = \{m_c, m_{A|E}, m_{E|A}\}$.

After consistency check, there can be multiple relation modifiers between two targets in similar past domains $D^s \subset D$. The number of domains supporting a relation modifier $m \in M_R$ can tell which kind of relation modifiers is common and likely to be correct. For example, given many past domains like "Laptop", "Tablet", "Cellphone", etc., "camera and battery" appears more than "camera's battery", "camera" should be modified by "battery" more with $m_{E|A}$ rather than m_c (likely to be an aspect).

Let $C^{d_{u+1}}_{m,t_j}(t_i)$ be the count that target t_i modified by target t_j on relation m in the current domain d_{u+1} (not in KB). The count $C^{(\text{CL})}$ is for updating the Conditional Label (CL) distributions considering the information in both the current domain d_{u+1} and the KB. It is calculated as:

$$C^{(\text{CL})}_{m,t_j}(t_i) = \begin{cases} C^{d_{u+1}}_{m,t_j}(t_i) & \text{if } C^{d_{u+1}}_{m,t_j}(t_i) > 0 \\ \sum_{d \in D^s} \mathbb{I}^d_{m,t_j}(t_i) & \text{if } \sum_{m \in M_R} C^{d_{u+1}}_{m,t_j}(t_i)) = 0 \end{cases}$$

This equation says that if there is any relation modifier existing between the two targets in the new domain d_{u+1}, we do not borrow edges from the KB; Otherwise, the number of similar past domains supporting the relation modifier m is used. Recall that $\mathbb{I}^d_{m,t_j}(t_i)$ is the result calculated by Equation (7) after label consistency check.

We use count $C^{(\text{CL})}_{m,t_j}(t_i)$ to update $q^{d_{u+1}}(M_{t_j}(t_i))$ using Equation (4) in Section 4.2. Then the conditional label distribution accommodating relation modifiers in the KB, $P^{(\text{LL1})}(L(t_i)|L(t_j))$, is calculated by Equation, (5) using $q^{d_{u+1}}(M_{t_j}(t_i))$. LL1 denotes *Lifelong Learning* 1.

Updating Neighbor Weight Distribution: Equation (6) says that $w(t_j|t_i)$ is the importance of target t_i's neighbor t_j to t_i among all t_i's neighbors. When updating conditional label distribution using the KB, the number of domains can decide which kind of relation modifiers m is more common between the two targets t_i and t_j. But we cannot tell that neighbor t_j is more important than another neighbor $t_{j'}$ to t_i.

For example, given the past domains such as "Laptop", "Tablet", "Cellphone", etc., no matter how many domains believe "camera" is an aspect given "battery" is also an aspect, if the current domain is "All-in-one desktop computer", we should not consider the strong influences from "battery" in the past domains. We should rely more on the weights of "camera"'s neighbors provided by "All-in-one desktop computer". That means "mouse", "keyboard", "screen" etc., should have strong influences on "camera" than "battery" because most All-in-one desktops (e.g. iMac) do not have battery.

We introduce another indicator variable $\mathbb{I}^D_{m,t_j}(t_i) = \bigcup_{d \in D^s} \mathbb{I}^d_{m,t_j}(t_i)$, to indicate whether target t_j modified t_i on relation m in past similar domains D^s. It only considers the existence of a

relation modifier m among domains D^s.

The count $C_{t_j}^{(w)}(t_i)$ for updating the neighbor weight (w) distribution considers both the KB and the current domain d_{u+1}. It is as follows: $C_{t_j}^{(w)}(t_i) = \begin{cases} \sum_{m \in M_R} C_{m,t_j}^{d_{u+1}}(t_i) & \text{if } \sum_{m \in M_R} C_{m,t_j}^{d_{u+1}}(t_i) > 0 \\ \sum_{m \in M_R} \mathbb{I}_{m,t_j}^{D_u}(t_i) & \text{if } \sum_{m \in M_R} C_{m,t_j}^{d_{u+1}}(t_i) = 0 \end{cases}$ This equation tells that if there are relation modifiers existing between the two targets in the new domain d_{u+1}, we count the total times that t_j modifies t_i in the new domain; Otherwise, we count the total kinds of relation modifiers in M_R if a relation modifier $m \in M_R$ existed in past domains. Let $w^{(\text{LL1})}(t_j|t_i)$ be the neighbor weight distribution considering knowledge from the KB and d_{u+1}. It is calculated by Equation (6) using $C_{t_j}^{(w)}(t_i)$.

The initial label distribution $P^{d_{u+1},0}$ is calculated by Equation (3) only using type modifiers found in the new domain d_{u+1}. We use Lifelong-RL-1 to denote the method that employs $P^{(\text{LL1})}(L(t_i)|L(t_j))$, $w^{(\text{LL1})}(t_j|t_i)$ and $P^{d_{u+1},0}$ as inputs for RL.

5.2 Exploiting Target Labels in the KB

Since we have target labels from past domains, we may have a better idea about the initial label probabilities of targets in the current domain d_{u+1}. For example, after labeling domains like "Cellphone", "Laptop", "Tablet," and "E-reader", we may have a good sense that "camera" is likely to be an aspect. To use such knowledge, we need to check if the type modifier of target t in the current domain matches those in past domains and only keep those domains that have such a matching type modifier.

Let $D^s \subset D$ be the past domains consistent with target t's type modifier in the current domain d_{u+1}. Let $C^{D^s}(L(t))$ be the number of domains in D^s that target t is labeled as $L(t)$. Let λ be the ratio that controls how much we trust knowledge from the KB. Then the initial label probability distribution $P^{d_{u+1},0}$ calculated by Equation (3) only using type modifier found in d_{u+1} is replaced by :

$$P^{(\text{LL2}),0}(L(t)) = \frac{|D| \times P^{d_{u+1},0}(L(t)) + \lambda C^{D^s}(L(t))}{|D| + \lambda |D|} \tag{8}$$

Similarly, let $D^s \subset D$ be the past domains consistent with both targets t_i's and t_j's type modifiers in d_{u+1}. Let $C^{D^s}(L(t_i), L(t_j))$ be the number of domains in D^s that t_i and t_j are labeled as $L(t_i)$ and $L(t_j)$ respectively. The conditional label probability distribution accommodating relation modifiers in the KB, $P^{(\text{LL1})}(L(t_i)|L(t_j))$, is further updated to $P^{(\text{LL2})}(L(t_i)|L(t_j))$ by exploiting the target labels in KB (LL2 denotes *Lifelong Learning* 2):

$$P^{(\text{LL2})}(L(t_i)|L(t_j)) = \frac{|D| \times P^{(\text{LL1})}(L(t_i)|L(t_j)) + \lambda C^{D^s}(L(t_i), L(t_j))}{|D| + \lambda |D|} \tag{9}$$

For example, given "this camera", "battery" in the current domain, we are more likely to consider domains (e.g. "Film Camera", "DSLR", but not "Cellphone") that have entity modifiers on "camera" and aspect modifiers on "battery". Then we count the number of those domains that label "camera" as entity and "battery" as aspect: $C^{D^s}(L(\text{camera}) = \text{entity}, L(\text{battery}) = \text{aspect})$. Similarly, we count domains having other types of target labels on "camera" and "battery". These counts form an updated conditional label distribution that estimates "camera" as an entity and "battery" as an aspect.

Note that $|D - D^s|$, the number of past domains not consistent with targets' type modifiers, is added to $C^{D^s}(L(t_i) = \text{NIL})$ and $C^{D^s}(L(t_i) = \text{NIL}, L(t_j))$ for Equations (8) and (9) respectively to make the sum over $L(t_i)$ equal to 1. We use Lifelong-RL to denote this method which uses $P^{(\text{LL2}),0}(L(t))$, $P^{(\text{LL2})}(L(t_i)|L(t_j))$ and $w^{(\text{LL1})}(t_j|t_i)$ as input for RL.

6 Experiments

We now evaluate the proposed method and compare with baselines. We use the DP method for target extraction (Qiu et al., 2011). This method uses dependency relations between opinion words and targets to extract targets using seed opinion words. Since our paper does not focus on extraction, interested readers can refer to (Qiu et al., 2011) for details.

6.1 Experiment Settings

Evaluation Datasets: We use two sets of datasets. The first set consists of eight (8) annotated review datasets. We use each of them as the new domain data in LML to compute precision, recall, F1 scores. Five of them are from (Hu and Liu, 2004), and the remaining three are from (Liu et al., 2016). They have been used for target extraction, and thus have annotated targets, but no annotation on whether a

Dataset	Product Type	# of Sentence	# of entity	# of aspect
D1	Computer	531	50	151
D2	Wireless Router	879	97	186
D3	Speaker	689	64	218
D4	DVD Player	740	50	159
D5	Digital Camera	597	70	239
D6	MP3 Player	1716	60	370
D7	Digital Camera	346	28	151
D8	Cell Phone	546	36	188

Table 1: Annotation details of the benchmark datasets.

Distribution	$L(t) =$ entity	$L(t) =$ aspect	$L(t) =$ NIL
$\mathbf{P_{m_E}}$	0.45	0.25	0.3
$\mathbf{P_{m_A}}$	0.3	0.4	0.3

$\mathbf{P_{m_c}}$	$L(t_j) =$ entity	$L(t_j) =$ aspect	$L(t_j) =$ NIL
$L(t_i) =$ entity	0.8	0.0	0.33
$L(t_i) =$ aspect	0.0	0.8	0.33
$L(t_i) =$ NIL	0.2	0.2	0.33
$\mathbf{P_{m_{E\|A}}}$	$L(t_j) =$ entity	$L(t_j) =$ aspect	$L(t_j) =$ NIL
$L(t_i) =$ entity	0.33	0.8	0.33
$L(t_i) =$ aspect	0.33	0.0	0.33
$L(t_i) =$ NIL	0.33	0.2	0.33
$\mathbf{P_{m_{A\|E}}}$	$L(t_j) =$ entity	$L(t_j) =$ aspect	$L(t_j) =$ NIL
$L(t_i) =$ entity	0.0	0.33	0.33
$L(t_i) =$ aspect	0.8	0.33	0.33
$L(t_i) =$ NIL	0.2	0.33	0.33

Table 2: Label Distribution for P_E and P_A and Conditional Label Distribution for P_{m_c}, $P_{m_{A|E}}$ and $P_{m_{E|A}}$

target is an entity or aspect. We made this annotation, which is straightforward. We used two annotators to annotate the datasets. The Cohen's kappa is 0.84. Through discussion, the annotators got complete agreement. Details of the datasets are listed in Table 1. Each cell is the number of distinct terms. These datasets are not very large but they are realistic because many products do not have a large number of reviews.

The second set consists of unlabeled review datasets from 100 diverse products or domains (Chen and Liu 2014). Each domain has 1000 reviews. They are treated as past domain data in LML since they are not annotated and thus cannot be used for computing evaluation measures.

Evaluating Measures: We mainly use precision $\mathcal{P}$, recall $\mathcal{R}$, and F_1-score $\mathcal{F}_1$ as evaluation measures. We take multiple occurrences of the same target as one count, and only evaluate entities and aspects. We will also give the accuracy results.

Compared Methods: We compare the following methods, including our proposed method, *Lifelong-RL*.

NER+TM: NER is Named Entity Recognition. We can regard the extracted terms from a NER system as entities and the rest of the targets as aspects. However, a NER system cannot identify entities such as "this car" from "this car is great." Its result is rather poor. But our type modifier (TM) does that, i.e., if an opinion target appears after "this" or "these" in at least two sentences, TM labels the target as an entity; otherwise an aspect. However, TM cannot extract named entities. Its result is also rather poor. We thus combine the two methods to give NER+TM as they complement each other very well. To make NER more powerful, we use two NER systems: Stanford-NER [1](Manning et al., 2014) and UIUC-NER[2] (Ratinov and Roth, 2009). NER+TM treats the extracted entities by the three systems as entities and the rest of the targets as aspects.

NER+TM+DICT: We run NER+TM on the 100 datasets for LML to get a list of entities, which we call the $dictionary$ (DICT). For a new task, if any target word is in the list, it is treated as an entity; otherwise an aspect.

RL: This is the base method described in Section 3. It performs relaxation labeling (RL) without the help of LML.

Lifelong-RL-1: This performs LML with RL but the current task only uses the relations in the KB from previous tasks (Section 5.1).

Lifelong-RL: This is our proposed final method. It improves Lifelong-RL-1 by further incorporating target labels in the KB from previous tasks (Section 5.2).

Parameter Settings: RL has 2 initial label distributions P_{m_E} and P_{m_A} and 3 conditional label distributions P_{m_c}, $P_{m_{E|A}}$ and $P_{m_{A|E}}$. Like other belief propagation algorithms, these probabilities need to be set empirically, as shown in Table 2. The parameter α is set to 1. Our LML method has one parameter λ for Lifelong-RL. We set it to 0.1.

6.2 Results Analysis

Table 3 shows the test results of all systems in precision, recall and F_1-score except NER+TM+DICT. NER+TM+DICT is not included due to space limitations and because it performed very poorly. The reason is that a target can be an entity in one domain

[1]http://nlp.stanford.edu/software/CRF-NER.shtml

[2]https://cogcomp.cs.illinois.edu/page/software_view/NETagger

Dataset	Entity												Aspect											
	NER+TM			RL			Lifelong-RL-1			Lifelong-RL			NER+TM			RL			Lifelong-RL-1			Lifelong-RL		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
D1	56.3	88	68.7	80	56	65.9	76.1	70	72.9	83.1	66	73.6	71.0	74.8	72.9	72.6	74.2	73.4	75.3	71.5	73.4	74.2	72.8	73.5
D2	64.8	75.3	69.7	71.6	42.3	53.2	81.1	62.9	70.9	85.5	78.4	81.8	61.9	90.3	73.5	61.4	85	71.3	67.2	92.5	77.9	70.4	90.9	79.3
D3	56.8	68.6	62.2	63.4	37.5	47.1	79.8	62.5	70.1	76.3	64.1	69.6	76.3	81.7	78.9	76.6	77.5	77.1	73.7	84.4	78.7	73.5	82.6	77.8
D4	76.7	42	54.3	69.3	42	52.3	77.9	70	73.7	78.6	70	74	68.8	71.7	70.2	68.3	70.4	69.3	70.4	65.4	67.8	70.6	66	68.2
D5	62.7	54.3	58.2	62.1	61.4	61.8	78.5	94.3	85.7	86.4	91.4	88.9	85.6	81.6	83.5	85.5	77.8	81.5	87	81.2	84	87.7	82	84.8
D6	69.9	38.3	49.5	67	56.7	61.4	74.7	75	74.8	77.4	73.3	75.3	75.4	83	79	76.2	81.1	78.6	78.8	85.9	82.2	78.9	86.2	82.4
D7	95	64.28	76.7	95.2	67.9	79.2	93.8	92.8	93.3	94.7	92.9	93.8	87.5	86.1	86.8	87.9	86.8	87.3	89.1	88.1	88.6	90.7	88.7	89.7
D8	65.9	41.7	51.1	65.5	72.2	68.7	72.3	83.3	77.4	79.4	86.1	82.6	76.1	81.9	78.9	77.8	80.9	79.3	81.4	89.4	85.2	81.9	89.9	85.7
Average	68.5	59.1	61.3	71.8	54.5	61.2	79.3	76.4	77.4	82.7	77.8	79.9	75.3	81.4	78	75.8	79.2	77.2	77.9	82.3	79.7	78.5	82.4	80.2

Table 3: Comparative results on Entity and Aspect in precision, recall and F_1 score: NER+TM+DICT's results are very poor and not included (see Section 6.2) for the average results.

but an aspect in another. Its average F_1-score for entity is only 49.2, and for aspect is only 50.2.

Entity Results Comparison: We observe from the table that although NER+TM combines NER and TM, its result for entities is still rather poor. We notice that phrases like "this price" causes low precision. Since it does not use many other relations and NER does not recognize many named entities that are written in lower case letters (e.g., "apple is good"), its recall is also low.

RL has a higher precision as it considers relation modifiers. However, its recall is low because it lacks information in its graph, which causes RL to make many wrong decisions. Lifelong-RL-1 introduces relation modifiers in KB from past domains into the current task. Both precision and recall increase markedly.

Lifelong-RL improves Lifelong-RL-1 further by considering target labels of past domains. Their counts improve the initial label probability distributions and conditional label probability distributions. For example, "this price" may appear in some domains but "price"'s target label is mostly aspect. We consider their counts in initial label distributions and thus rectify the initial distribution of "price". This makes "price" easier to be classified as aspect and thus improves the precision for entity.

Aspect Results Comparison: For aspects, the trend is the same but the improvements are not as dramatic as for entity. This is because the distribution of entity and aspect in the data is highly skewed. There are many more aspects than entities as we can see from the Table 1. When an entity term is wrongly classified as an aspect, it has much less impact on the aspect result than on the entity result.

Accuracy Results Comparison: Table 4 gives the classification accuracy results considering all three classes. We can see the similar trend. NER+TM+DICT's average accuracy is only 45.89 and is not included in the table.

Dataset	NER+TM	RL	Lifelong-RL-1	Lifelong-RL
D1	64.93	74.29	75.51	76.34
D2	62.94	63.53	69.8	73.82
D3	70.04	73.74	74.83	74.1
D4	70.81	68.57	73.33	73.63
D5	82.07	81.46	85.22	87.5
D6	74.83	75.06	78	78.63
D7	88.18	88.63	89.68	91.3
D8	74.54	75.43	79.57	81.4
Average	73.55	75.07	78.24	79.59

Table 4: Results in accuracy: NER+TM+DICT's results are again very poor and thus not included.

7 Conclusion

This paper studied the problem of classifying opinion targets into entities and aspects. To the best of our knowledge, this problem has not been attempted in the unsupervised opinion target extraction setting. But this is an important problem because without separating or classifying them one will not know whether an opinion is about an entity as a whole or about a specific aspect of an entity. This paper proposed a novel method based on relaxation labeling and the paradigm of lifelong machine learning to solve the problem. Experimental results showed the effectiveness of the proposed method.

Acknowledgments

This work was partially supported by National Science Foundation (NSF) grants IIS-1407927 and IIS-1650900, and NCI grant R01CA192240. The content of the paper is solely the responsibility of the authors and does not necessarily represent the official views of the NSF or NCI.

References

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *NAACL '10*, pages 804–812.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Zhiyuan Chen and Bing Liu. 2014a. Mining topics in documents: Standing on the shoulders of big data. In *KDD '14*, pages 1116–1125.

Zhiyuan Chen and Bing Liu. 2014b. Topic modeling using topics from many domains, lifelong learning and big data. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 703–711.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. *Volume 2: Short Papers*, pages 750–756.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *ACL '10*, pages 269–274.

Lei Fang and Minlie Huang. 2012. Fine granular aspect analysis using latent structural models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 333–337.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Robert A Hummel and Steven W Zucker. 1983. On the foundations of relaxation labeling processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (3):267–287.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *EMNLP '10*, pages 1035–1045.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 100107.

Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Sentiment analysis with global topics and local dependency. In *AAAI '10*, pages 1371–1376.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384.

Kang Liu, Liheng Xu, and Jun Zhao. 2013. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews. In *ACL (1)*, pages 1754–1763.

Qian Liu, Bing Liu, Yuanlin Zhang, DooSoon Kim, and Zhiqiang Gao. 2016. Improving opinion aspect extraction using semantic similarity and aspect associations. In *AAAI*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *WWW '07*, pages 171–180.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *ACL '13*, pages 1643–1654.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *ACL '12*, volume 1, pages 339–348.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. *SocialNLP 2014*, page 28.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37(1):9–27.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 6.

Paul Ruvolo and Eric Eaton. 2013. Active task selection for lifelong machine learning. In *AAAI*.

Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 49–55.

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Bo Wang and Houfeng Wang. 2008. Bootstrapping both product features and opinion words from chinese customer reviews with cross-inducing. In *IJCNLP '08*, pages 289–295.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. In *KDD '10*, pages 783–792.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *EMNLP '09*, pages 1533–1541.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *COLING '10: Posters*, pages 1462–1470.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2013. Collective opinion target extraction in chinese microblogs. In *EMNLP*, volume 13, pages 1840–1850.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *CIKM '06*, pages 43–50.

Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification

Jianfei Yu
School of Information Systems
Singapore Management University
jfyu.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

In this paper, we study cross-domain sentiment classification with neural network architectures. We borrow the idea from Structural Correspondence Learning and use two auxiliary tasks to help induce a sentence embedding that supposedly works well across domains for sentiment classification. We also propose to jointly learn this sentence embedding together with the sentiment classifier itself. Experiment results demonstrate that our proposed joint model outperforms several state-of-the-art methods on five benchmark datasets.

1 Introduction

With the growing need of correctly identifying the sentiments expressed in subjective texts such as product reviews, sentiment classification has received continuous attention in the NLP community for over a decade (Pang et al., 2002; Pang and Lee, 2004; Hu and Liu, 2004; Choi and Cardie, 2008; Nakagawa et al., 2010). One of the big challenges of sentiment classification is how to adapt a sentiment classifier trained on one domain to a different new domain. This is because sentiments are often expressed with domain-specific words and expressions. For example, in the **Movie** domain, words such as *moving* and *engaging* are usually positive, but they may not be relevant in the **Restaurant** domain. Since labeled data is expensive to obtain, it would be very useful if we could adapt a model trained on a *source domain* to a *target domain*.

Much work has been done in sentiment analysis to address this *domain adaptation* problem (Blitzer et al., 2007; Pan et al., 2010; Bollegala et al., 2011; Ponomareva and Thelwall, 2012; Bollegala et al., 2016). Among them, an appealing method is the Structural Correspondence Learning (SCL) method (Blitzer et al., 2007), which uses pivot feature prediction tasks to induce a projected feature space that works well for both the source and the target domains. The intuition behind is that these pivot prediction tasks are highly correlated with the original task. For sentiment classification, Blitzer et al. (2007) first chose pivot words which have high mutual information with the sentiment labels, and then set up the pivot prediction tasks to be the predictions of each of these pivot words using the other words.

However, the original SCL method is based on traditional discrete feature representations and linear classifiers. In recent years, with the advances of deep learning in NLP, multi-layer neural network models such as RNNs and CNNs have been widely used in sentiment classification and achieved good performance (Socher et al., 2013; Dong et al., 2014a; Dong et al., 2014b; Kim, 2014; Tang et al., 2015). In these models, dense, real-valued feature vectors and non-linear classification functions are used. By using real-valued word embeddings pre-trained from a large corpus, these models can take advantage of the embedding space that presumably better captures the syntactic and semantic similarities between words. And by using non-linear functions through multi-layer neural networks, these models represent a more expressive hypothesis space. Therefore, it would be interesting to explore how these neural network models could be extended for cross-domain sentiment classification.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 236–246,
Austin, Texas, November 1-5, 2016.

There has been some recent studies on neural network-based domain adaptation (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2014). They use Stacked Denoising Auto-encoders (SDA) to induce a hidden representation that presumably works well across domains. However, SDA is fully unsupervised and does not consider the end task we need to solve, i.e., the sentiment classification task. In contrast, the idea behind SCL is to use carefully-chosen auxiliary tasks that correlate with the end task to induce a hidden representation. Another line of work aims to learn a low dimensional representation for each feature in both domains based on predicting its neighboring features (Yang and Eisenstein, 2015; Bollegala et al., 2015). Different from these methods, we aim to directly learn sentence embeddings that work well across domains.

In this paper, we aim to extend the main idea behind SCL to neural network-based solutions to sentiment classification to address the domain adaptation problem. Specifically, we borrow the idea of using pivot prediction tasks from SCL. But instead of learning thousands of pivot predictors and performing singular value decomposition on the learned weights, which all relies on *linear* transformations, we introduce only two auxiliary binary prediction tasks and directly learn a *non-linear* transformation that maps an input to a dense embedding vector. Moreover, different from SCL and the auto-encoder-based methods, in which the hidden feature representation and the final classifier are learned sequentially, we propose to jointly learn the hidden feature representation together with the sentiment classification model itself, and we show that joint learning works better than sequential learning.

We conduct experiments on a number of different source and target domains for sentence-level sentiment classification. We show that our proposed method is able to achieve the best performance compared with a number of baselines for most of these domain pairs.

2 Related Work

Domain Adaptation: Domain adaptation is a general problem in NLP and has been well studied in recent years (Blitzer et al., 2006; Daumé III, 2007; Jiang and Zhai, 2007; Dredze and Crammer, 2008; Titov, 2011; Yu and Jiang, 2015). For sentiment classification, most existing domain adaptation methods are based on traditional discrete feature representations and linear classifiers. One line of work focuses on inducing a general low-dimensional cross-domain representation based on the co-occurrences of domain-specific and domain-independent features (Blitzer et al., 2007; Pan et al., 2010; Pan et al., 2011). Another line of work tries to derive domain-specific sentiment words (Bollegala et al., 2011; Li et al., 2012). Our proposed method is similar to the first line of work in that we also aim to learn a general, cross-domain representation (sentence embeddings in our case).

Neural Networks for Sentiment Classification: A recent trend of deep learning enhances various kinds of neural network models for sentiment classification, including Convolutional Neural Networks (CNNs), Recursive Neural Network (ReNNs) and Recurrent Neural Network (RNNs), which have been shown to achieve competitive results across different benchmarks (Socher et al., 2013; Dong et al., 2014a; Dong et al., 2014b; Kim, 2014; Tang et al., 2015). Inspired by their success in standard in-domain settings, it is intuitive for us to apply these neural network models to domain adaptation settings.

Denoising Auto-encoders for Domain Adaptation: Denoising Auto-encoders have been extensively studied in cross-domain sentiment classification, since the representations learned through multi-layer neural networks are robust against noise during domain adaptation. The initial application of this idea is to directly employ stacked denoising auto-encoders (SDA) by reconstructing the original features from data that are corrupted with noise (Glorot et al., 2011), and Chen et al. (2012) proposed to analytically marginalize out the corruption during SDA training. Later Yang and Eisenstein (2014) further showed that their proposed structured dropout noise strategy can dramatically improve the efficiency without sacrificing the accuracy. However, these methods are still based on traditional discrete representation and do not exploit the idea of using auxiliary tasks that are related to the end task. In contrast, the sentence embeddings learned from our method are derived from real-valued feature vectors and rely on related auxiliary tasks.

3 Method

In this section we present our sentence embedding-based domain adaptation method for sentiment classification. We first introduce the necessary notation and an overview of our method. we then delve into the details of the method.

3.1 Notation and Method Overview

We assume that each input is a piece of text consisting of a sequence of words. For the rest of this paper, we assume each input is a sentence, although our method is general enough for longer pieces of text. Let $\boldsymbol{x} = (x_1, x_2, \ldots)$ denote a sentence where each $x_i \in \{1, 2, \ldots, V\}$ is a word in the vocabulary and V is the vocabulary size. Let the sentiment label of $\boldsymbol{x}$ be $y \in \{+, -\}$ where $+$ denotes a positive sentiment and $-$ a negative sentiment. We further assume that we are given a set of labeled training sentences from a source domain, denoted by $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N^s}$. Also, we have a set of unlabeled sentences from a target domain, denoted by $\mathcal{D}^t = \{\boldsymbol{x}_i^t\}_{i=1}^{N^t}$. Our goal is to learn a good sentiment classifier from both $\mathcal{D}^s$ and $\mathcal{D}^t$ such that the classifier works well on the target domain.

A baseline solution without considering any domain difference is to simply train a classifier using $\mathcal{D}^s$, and with the recent advances in neural network-based methods to sentence classification, we consider a baseline that uses a multi-layer neural network such as a CNN or an RNN to perform the classification task. To simplify the discussion and focus on the domain adaptation ideas we propose, we will leave the details of the neural network model we use in Section 3.5. For now, we assume that a multi-layer neural network is used to transform each input $\boldsymbol{x}$ into a sentence embedding vector $\boldsymbol{z}$. Let us use f_Θ to denote the transformation function parameterized by Θ, that is, $\boldsymbol{z} = f_\Theta(\boldsymbol{x})$. Next, we assume that a linear classifier such as a softmax classifier is learned to map $\boldsymbol{z}$ to a sentiment label y.

We introduce two auxiliary tasks which presumably are highly correlated with the sentiment classification task itself. Labels for these auxiliary tasks can be automatically derived from unlabeled data in both the source and the target domains. With the help of the two auxiliary tasks, we learn a non-linear transformation function $f_{\Theta'}$ from unlabeled data and use it to derive a sentence embedding vector $\boldsymbol{z}'$ from sentence $\boldsymbol{x}$, which supposedly works better across domains. Finally we use the source domain's training data to learn a linear classifier on the representation $\boldsymbol{z} \oplus \boldsymbol{z}'$, where $\oplus$ is the operator that concatenates two vectors. Figure 1 gives the outline of our method.

3.2 Auxiliary Tasks

Our two auxiliary tasks are about whether an input sentence contains a positive or negative domain-independent sentiment word. The intuition is the following. If we have a list of domain-independent positive sentiment words, then an input sentence that contains one of these words, regardless of the domain the sentence is from, is more likely to contain an overall positive sentiment. For example, a sentence containing the word *good* is likely to be overall positive. Moreover, the rest of the sentence excluding the word *good* may contain domain-specific words or expressions that also convey a positive sentiment. For example, in the sentence "The laptop is good and goes really fast," we can see that the word *fast* is a domain-specific sentiment word, and its sentiment polarity correlates with that of the word *good*, which is domain-independent. Therefore, we can hide the domain-independent positive words in a sentence and try to use the other words in the sentence to predict whether the original sentence contains a domain-independent positive word. There are two things to note about this auxiliary task: (1) The label of the task can be automatically derived provided that we have the domain-independent positive word list. (2) The task is closely related to the original task of sentence-level sentiment classification. Similarly, we can introduce a task to predict the existence of a domain-independent negative sentiment word in a sentence.

Formally, let us assume that we have two domain-independent sentiment word lists, one for the positive sentiment and the other for the negative sentiment. Details of how these lists are obtained will be given in Section 3.5. Borrowing the term from SCL, we refer to these sentiment words as pivot words. For each sentence $\boldsymbol{x}$, we replace all the occurrences of these pivot words with a special token *UNK*. Let $g(\cdot)$ be a function that denotes this procedure, that is, $g(\boldsymbol{x})$ is the resulting sentence with

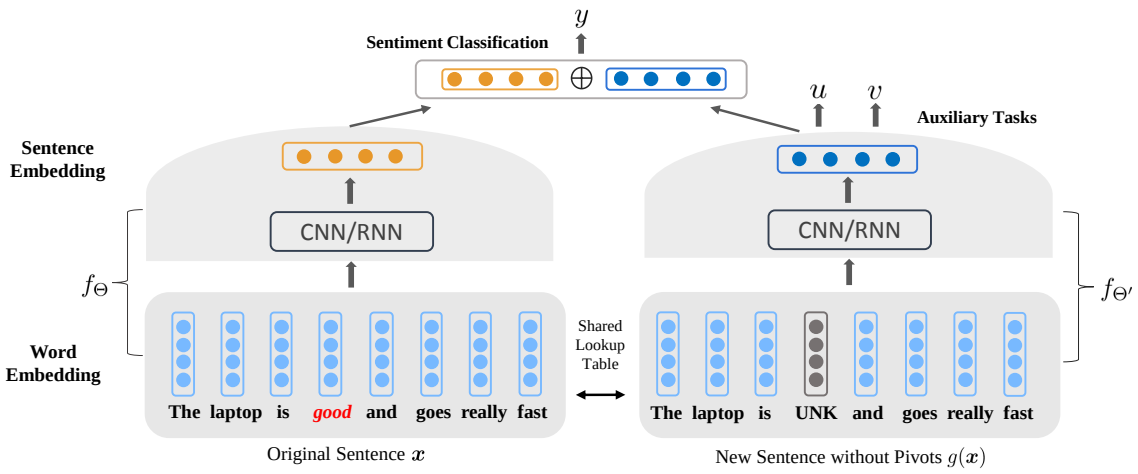


Figure 1: The Outline of our Proposed Method.

UNK tokens. We then introduce two binary labels for $g(\boldsymbol{x})$. The first label u indicates whether the original sentence $\boldsymbol{x}$ contains at least one domain-independent positive sentiment word, and the second label v indicates whether $\boldsymbol{x}$ contains at least one domain-independent negative sentiment word. Figure 1 shows an example sentence $\boldsymbol{x}$, its modified version $g(\boldsymbol{x})$ and the labels u and v for $\boldsymbol{x}$. We further use $\mathcal{D}^a = \{(\boldsymbol{x}_i, u_i, v_i)\}_{i=1}^{N^a}$ to denote a set of training sentences for the auxiliary tasks. Note that the sentences in $\mathcal{D}^a$ can be from the sentences in $\mathcal{D}^s$ and $\mathcal{D}^t$, but they can also be from other unlabeled sentences.

3.3 Sentence Embeddings for Domain Adaptation

With the two auxiliary tasks, we can learn a neural network model in a standard way to produce sentence embeddings that work well for the auxiliary tasks. Specifically, we still use Θ' to denote the parameters of the neural network that produces the sentence embeddings (and $f_{\Theta'}$ the corresponding transformation function), and we use $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ to denote the parameters of two softmax classifiers for the two auxiliary tasks, respectively. Using cross-entropy loss, we can learn Θ' by minimizing the following loss function:

$$
\begin{aligned}
& J(\Theta', \boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \\
= & -\sum_{(\boldsymbol{x},u,v)\in\mathcal{D}^a} \Big(\log p(u|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^+) \\
& + \log p(v|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^-) \Big),
\end{aligned}
$$

where $p(y|\boldsymbol{z}; \boldsymbol{\beta})$ is the probability of label y given vector $\boldsymbol{z}$ and parameter $\boldsymbol{\beta}$ under softmax regression.

With the learned Θ', we can derive a sentence embedding $\boldsymbol{z}'$ from any sentence. Although we could simply use this embedding $\boldsymbol{z}'$ for sentiment classification through another softmax classifier, this may not be ideal because $\boldsymbol{z}'$ is transformed from $g(\boldsymbol{x})$, which has the domain-independent sentiment words removed. Similar to SCL and some other previous work, we concatenate the embedding vector $\boldsymbol{z}'$ with the standard embedding vector $\boldsymbol{z}$ for the final classification.

3.4 Joint Learning

Although we can learn Θ' using $\mathcal{D}^a$ as a first step, here we also explore a joint learning setting. In this setting, Θ' is learned together with the neural network model used for the end task, i.e., sentiment classification. This way, the learning of Θ' depends not only on $\mathcal{D}^a$ but also on $\mathcal{D}^s$, i.e., the sentiment-labeled training data from the source domain.

Specifically, we use Θ to denote the parameters for a neural network that takes the original sentence $\boldsymbol{x}$ and transforms it to a sentence embedding (and f_Θ the corresponding transformation function). We use $\boldsymbol{\gamma}$ to denote the parameters of a softmax classifier that operates on the concatenated sentence embedding $\boldsymbol{z} \oplus \boldsymbol{z}'$ for sentiment classification. With joint learning, we try to minimize the following loss func-

tion:

$$
\begin{aligned}
& J(\Theta, \Theta', \boldsymbol{\gamma}, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \\
= & -\sum_{(\boldsymbol{x},y)\in\mathcal{D}^s} \Big(\log p(y|f_\Theta(\boldsymbol{x}) \oplus f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\gamma}) \Big) \\
& -\sum_{(\boldsymbol{x},u,v)\in\mathcal{D}^a} \Big(\log p(u|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^+) \\
& + \log p(v|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^-) \Big).
\end{aligned}
$$

We can see that this loss function contains two parts. The first part is the cross-entropy loss based on the true sentiment labels of the sentences in $\mathcal{D}^s$. The second part is the loss based on the auxiliary tasks and the data $\mathcal{D}^a$, which are derived from unlabeled sentences.

Finally, to make a prediction on a sentence, we use the learned Θ and Θ' to derive a sentence embedding $f_\Theta(\boldsymbol{x}) \oplus f_{\Theta'}(g(\boldsymbol{x}))$, and then use the softmax classifier parameterized by the learned $\boldsymbol{\gamma}$ to make the final prediction.

3.5 Implementation Details

In this section we explain some of the model details.

Pivot Word Selection

Recall that the two auxiliary tasks depend on two domain-independent sentiment word lists, i.e., pivot word lists. Different from Blitzer et al. (2007), we employ weighted log-likelihood ratio (WLLR) to select the most positive and negative words in both domains as pivots. The reason is that in our preliminary experiments we observe that mutual information (used by Blitzer et al. (2007)) is biased towards low frequency words. Some high frequency words including *good* and *great* are scored low. In comparison, WLLR does not have this issue. The same observation was also reported previously by Li et al. (2009).

More specifically, we first tokenize the sentences in $\mathcal{D}^s$ and $\mathcal{D}^t$ and perform part-of-speech tagging using the NLTK toolkit. Next, we extract only adjectives, adverbs and verbs with a frequency of at least 3 in the source domain and at least 3 in the target domain. We also remove negation words such as *not* and stop words using a stop word list. We then measure each remaining candidate word's relevance to the positive and the negative classes based on $\mathcal{D}^s$ by computing the following scores:

$$
r(w, y) \;=\; \tilde{p}(w|y) \log \frac{\tilde{p}(w|y)}{\tilde{p}(w|\bar{y})},
$$

where w is a word, $y \in \{+, -\}$ is a sentiment label, $\bar{y}$ is the opposite label of y, and $\tilde{p}(w|y)$ is the empirical probability of observing w in sentences labeled with y. We can then rank the candidate words in decreasing order of $r(w, +)$ and $r(w, -)$. Finally, we select the top 25% from each ranked list as the final lists of pivot words for the positive and the negative sentiments. Some manual inspection shows that most of these words are indeed domain-independent sentiment words.

Neural Network Model

Our framework is general and potentially we can use any neural network model to transform an input sentence to a sentence embedding vector. In this paper, we adopt a CNN-based approach because it has been shown to work well for sentiment classification. Specifically, each word (including the token *UNK*) is represented by a word embedding vector. Let $\mathbf{W} \in \mathbb{R}^{d\times V}$ denote the lookup table for words, where each column is a d-dimensional embedding vector for a word type. Two separate CNNs are used to process $\boldsymbol{x}$ and $g(\boldsymbol{x})$, and their mechanisms are the same. For a word x_i in each CNN, the embedding vectors inside a window of size n centered at i are concatenated into a new vector, which we refer to as $\mathbf{e}_i \in \mathbb{R}^{nd}$. A convolution operation is then performed by applying a filter $\mathbf{F} \in \mathbb{R}^{h\times nd}$ on $\mathbf{e}_i$ to produce a hidden vector $\mathbf{h}_i = m(\mathbf{F}\mathbf{e}_i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^h$ is a bias vector and m is an element-wise non-linear transformation function. Note that we pad the original sequence in front and at the back to ensure that at each position i we have n vectors to be combined into $\mathbf{h}_i$. After the convolution operation is applied to the whole sequence, we obtain $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots]$, and we apply a max-over-time pooling operator to take the maximum value of each row of $\mathbf{H}$ to obtain an overall hidden vector, i.e., $\boldsymbol{z}$ for $\boldsymbol{x}$ and $\boldsymbol{z}'$ for $g(\boldsymbol{x})$.

It is worth noting that the two neural networks corresponding to f_Θ and $f_{\Theta'}$ share the same word embedding lookup table. This lookup table is initialized with word embeddings from *word2vec*[1] and

[1] https://code.google.com/p/word2vec/

is updated during our learning process. Note that the token *UNK* is initialized as a zero vector and never updated.

3.6 Differences from SCL

Although our method is inspired by SCL, there are a number of major differences: (1) Our method is based on neural network models with continuous, dense feature representations and non-linear transformation functions. SCL is based on discrete, sparse feature vectors and linear transformations. (2) Although our pivot word selection is similar to that of SCL, in the end we only use two auxiliary tasks while SCL uses much more pivot prediction tasks. (3) We can directly learn the transformation function f'_Θ that produces the hidden representation, while SCL relies on SVD to learn the projection function. (4) We perform joint learning of the auxiliary tasks and the end task, i.e., sentiment classification, while SCL performs the learning in a sequential manner.

4 Experiments

4.1 Data Sets and Experiment Settings

Data Set	# Sentences	# Words
Movie1(MV1)	10662	18765
Movie2(MV2)	9613	16186
Camera(CR)	3770	5340
Laptop(LT)	1907	2837
Restaurant(RT)	1572	2930

Table 1: Statistics of our data sets.

To evaluate our proposed method, we conduct experiments using five benchmark data sets. The data sets are summarized in Table 1. *Movie1*[2] and *Movie2*[3] are movie reviews labeled by Pang and Lee (2005) and Socher et al. (2013), respectively. *Camera*[4] are reviews of digital products such as MP3 players and cameras (Hu and Liu, 2004). *Laptop* and *Restaurant*[5] are laptop and restaurant reviews taken from SemEval 2015 Task 12.

We consider 18 pairs of data sets where the two data sets come from different domains.[6] For neural network-based methods, we randomly pick 200 sentences from the target domain as the development set for parameter tuning, and the rest of the data from the target domain as the test data.

4.2 Baselines and Hyperparameters

We consider the following baselines:

Naive is a non-domain-adaptive baseline based on bag-of-word representations.

SCL is our implementation of the Structural Correspondence Learning method. We set the number of induced features K to 100 and rescale factor $\alpha = 5$, and we use 1000 pivot words based on our preliminary experiments.

mDA is our implementation of marginalized Denoising Auto-encoders (Chen et al., 2012), one of the state-of-the-art domain adaptation methods, which learns a shared hidden representation by reconstructing pivot features from corrupted inputs. Following Yang and Eisenstein (2014), we employ the efficient and effective structured dropout noise strategy without any parameter. The top 500 features are chosen as pivots based on our preliminary experiments.

NaiveNN is a non-domain-adaptive baseline based on CNN, as described in Section 3.5.

Aux-NN is a simple combination of our auxiliary tasks with NaiveNN, which treats the derived label of two auxiliary tasks as two features and then appends them to the hidden representation learned from CNN, followed by a softmax classifier.

SCL-NN is a naive combination of SCL with NaiveNN, which appends the induced representation from **SCL** to the hidden representation learned from CNN, followed by a softmax classifier.

mDA-NN is similar to **SCL-NN** but uses the hidden representation derived from **mDA**.

Sequential is our proposed method without joint learning, which first learns Θ' based on $\mathcal{D}^a$ and then learns Θ and γ based on $\mathcal{D}^s$ with fixed Θ'.

Joint is our proposed joint learning method, that is, we jointly learn Θ and Θ'.

[2] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] http://nlp.stanford.edu/sentiment/

[4] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[5] Note that the original data set is for aspect-level sentiment analysis. We remove sentences with opposite polarities towards different aspects, and use the consistent polarity as the sentence-level sentiment of each remaining sentence.

[6] Because *Movie1* and *Movie2* come from the same domain, we do not take this pair.

Task		Method									
Source	Target	Naive	Naive++	SCL++	mDA++	NaiveNN	Aux-NN	SCL-NN	mDA-NN	Sequential	Joint
MV1	*LT*	0.656	0.739	0.742	0.742	0.773	0.779	0.776	0.780	0.774	**0.804***
MV1	*RT*	0.625	0.742	0.750	0.761	0.802	0.794	0.817	0.819	0.814	**0.825***
MV1	*CR*	0.609	0.684	0.688	0.688	0.721	0.717	0.734	0.730	0.717	**0.747***
MV2	*LT*	0.699	0.760	0.765	0.772	0.805	0.811	0.800	0.811	0.808	**0.827***
MV2	*RT*	0.696	0.761	0.768	0.778	0.813	0.819	0.824	0.825	0.833	**0.840***
MV2	*CR*	0.644	0.697	0.705	0.706	0.738	0.732	0.736	0.756	0.745	**0.768***
CR	*LT*	0.780	0.791	0.802	0.806	0.848	0.848	0.846	0.850	0.856	**0.858***
CR	*RT*	0.746	0.784	0.782	0.789	0.827	0.835	0.841	0.839	0.835	**0.844***
CR	*MV1*	0.593	0.597	0.612	0.612	0.685	0.689	0.689	0.692	0.687	**0.696***
CR	*MV2*	0.609	0.629	0.644	0.640	0.735	0.726	0.734	0.731	0.735	**0.736**
LT	*RT*	0.736	0.781	0.800	0.810	0.819	0.820	0.823	**0.852**	0.841	0.840
LT	*MV1*	0.574	0.601	0.612	0.630	**0.711**	0.703	0.702	0.709	0.705	0.707
LT	*MV2*	0.588	0.632	0.645	0.663	0.742	0.745	0.739	**0.747**	0.746	**0.747**
LT	*CR*	0.736	0.762	0.768	0.780	0.791	0.796	0.803	**0.819**	0.803	0.817
RT	*LT*	0.732	0.777	0.777	0.799	0.817	0.822	0.831	0.826	0.828	**0.834***
RT	*MV1*	0.580	0.604	0.618	0.643	0.721	0.726	0.724	**0.734**	0.722	0.724
RT	*MV2*	0.605	0.630	0.633	0.664	0.761	0.762	0.756	**0.772**	0.757	0.765
RT	*CR*	0.689	0.708	0.704	0.732	0.764	0.772	0.759	0.774	0.772	**0.779***
Average		0.661	0.704	0.712	0.723	0.770	0.772	0.774	0.781	0.777	**0.787**

Table 2: Comparison of classification accuracies of different methods. * indicates that our joint method is significantly better than NaiveNN, Aux-NN, SCL-NN and mDA-NN with $p < 0.05$ based on McNemar's paired significance test.

For **Naive**, **SCL** and **mDA**, we use LibLinear[7] to train linear classifiers and use its default hyperparameters. In all the tasks, we use unigrams and bigrams with a frequency of at least 4 as features for classification. For the word embeddings, we set the dimension d to 300. For CNN, we set the window size to 3. Also, the size of the hidden representations $\boldsymbol{z}$ and $\boldsymbol{z}'$ is set to 100. Following Kim (2014), the non-linear activation function in CNN is Relu, the mini-batch size is 50, the dropout rate α equals 0.5, and the hyperparameter for the l_2 norms is set to be 3. For **Naive**, **SCL** and **mDA**, we do not use the 200 sentences in the development set for tuning parameters. Hence, for fair comparison, we also include settings where the 200 sentences are added to the training set. We denote these settings by **++**.

4.3 Results

In Table 2, we report the results of all the methods. It is easy to see that the performance of **Naive** is very limited, and the incorporation of 200 reviews in the development set (**Naive++**) brings in 4.3% of improvement on average. **SCL++** and **mDA++** can further improve the average accuracy respectively by 0.8% and 1.9%, which verifies the usefulness of these two domain adaptation methods. However, we can easily see that the performance of these domain adaptation methods based on discrete, bag-of-word representations is even much lower than the non-domain-adaptive method on continuous representations (**NaiveNN**). This confirms that it is useful to develop domain adaptation methods based on embedding vectors and neural network models.

Moreover, we can find that the performance of simply appending two features from auxiliary tasks to **NaiveNN** (i.e., **Aux-NN**) is quite close to that of **NaiveNN** on most data set pairs, which shows that it is not ideal for domain adaptation. In addition, although the shared hidden representations derived from SCL and mDA are based on traditional bag-of-word representations, **SCL-NN** and **mDA-NN** can still improve the performance of **NaiveNN** on most data set pairs, which indicates that the derived shared hidden representations by SCL and by mDA can generalize better across domains and are generally useful for domain adaptation.

Finally, it is easy to see that our method with joint learning outperforms **SCL-NN** on almost all the data set pairs. And in comparison with **mDA-NN**, our method with joint learning can also outper-

[7] http://www.csie.ntu.edu.tw/cjlin/liblinear/

Task		Method		
Source	Target	NaiveNN	mDA-NN	Joint
MV1	*LT*	0.802	0.799	**0.816***
MV1	*RT*	0.816	0.820	**0.838***
MV1	*CR*	0.744	0.757	**0.767***
MV2	*LT*	0.823	0.830	**0.839***
MV2	*RT*	0.837	0.829	**0.850***
MV2	*CR*	0.753	0.769	**0.773***
CR	*LT*	0.853	0.863	**0.870***
CR	*RT*	0.840	**0.856**	0.851
CR	*MV1*	0.699	0.701	**0.704***
CR	*MV2*	**0.745**	0.741	**0.745**
LT	*RT*	0.839	**0.849**	**0.849**
LT	*MV1*	0.714	0.710	**0.720***
LT	*MV2*	0.759	**0.767**	0.766
LT	*CR*	0.803	**0.815**	0.814
RT	*LT*	0.825	0.839	**0.841**
RT	*MV1*	0.724	**0.737**	0.732
RT	*MV2*	0.762	**0.771**	0.768
RT	*CR*	0.773	0.777	**0.783***
Average		0.784	0.790	**0.796**

Table 3: Comparison of our method **Joint** with **NaiveNN** and **mDA-NN** in a setting where some labeled target data is used.

form it on most data set pairs, especially when the size of the labeled data in the source domain is relatively large. Furthermore, we can easily observe that for our method, joint learning generally works better than sequential learning. All these observations show the advantage of our joint learning method.

In Table 3, we also show the comparison between **mDA-NN** and our model under a setting some labeled target data is used. Specifically, we randomly select 100 sentences from the development set and mix them with the training set. We can observe that our method **Joint** outperforms **NaiveNN** and **mDA-NN** by 1.2% and 0.6%, respectively, which further confirms the effectiveness of our model. But, in comparison with the setting where no target data is available, the average improvement of our method over **NaiveNN** is relatively small.

Hence, to give a deeper analysis, we further show the comparison of **Joint** and **NaiveNN** with respect to the number of labeled target data in Figure 2. Note that for space limitation, we only present the results on *MV2* $\rightarrow$ *RT* and *MV2* $\rightarrow$ *CR*. Similar trends have been observed on other data set pairs. As we can see from Figure 2, the difference between the performance of **NaiveNN** and that of **Joint** gradually decreases with the increase of the number of labeled

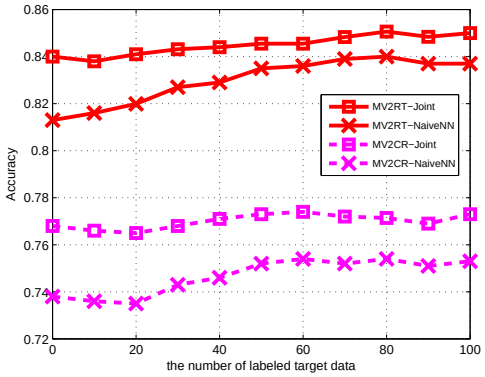


Figure 2: The influence of the number of labeled target data.

target data. This indicates that our joint model is much more effective when no or small number of labeled target data is available.

4.4 Case Study

To obtain a better understanding of our method, we conduct a case study where the source is *CR* and the target is *RT*.

For each sentiment polarity, we try to extract the most useful trigrams for the final predictions. Recall that our CNN models use a window size of 3, which corresponds to trigrams. By tracing the final prediction scores back through the neural network, we are able to locate the trigrams which have contributed the most through max-pooling. In Table 4, we present the most useful trigrams of each polarity extracted by **NaiveNN** and by the two components of our sequential and joint method. **Sequential-original** and **Joint-original** refer to the CNN corresponding to f_Θ while **Sequential-auxiliary** and **Joint-auxiliary** refer to the CNN corresponding to $f_{\Theta'}$, which is related to the auxiliary tasks.

In Table 4, we can easily observe that for **NaiveNN**, the most important trigrams are domain-independent, which contain some general sentiment words like *good*, *great* and *disappointing*. For our sequential model, the most important trigrams captured by **Sequential-original** are similar to **NaiveNN**, but due to the removal of the pivot words in each sentence, the most important trigrams extracted by **Sequential-auxiliary** are domain-specific, including target-specific sentiment words like *oily*, *friendly* and target-specific aspect words like *flavor*, *atmosphere*. But since aspect words are irrelevant to our sentiment classification

Method	Negative Sentiment	Positive Sentiment
NaiveNN	disappointing_*_*, **disgusting_*_***, it_is_not, slow_*_*, *_too_bad, *_*_terrible, **place_is_not**, unpleasant_experience_*, **would_not_go**, *_the_only	*_*_great, good_*_*,*_*_best, *_i_love, was_very_good,*_*_excellent, wonderful_*_*, *_*_amazing, *_*_nice
Sequential-original	**disgusting_*_***, disappointing_*_*, *_*_terrible expensive_*_*, it_is_not, unpleasant_experience_*, slow_*_*, *_too_bad, probably_would_not, awful_*_*	*_*_great, good_*_*, *_*_best, *_i_love, *_highly_recommended, *_*_excellent, wonderful_*_*, is_amazing_*, is_the_perfect
Sequential-auxiliary	**disgusting_*_***, **never_go_back**, money_*_*, **rude_*_***, flavor_*_*, *_this_place, **oily_*_***, prices_*_*, **inedible_!_***, **this_place_survives**	**delicious_*_***, **friendly_*_***, food_*_*, food_is_UNK, *_highly_UNK, **fresh_*_***, atmosphere_*_*, *_i_highly, nyc_*_*
Joint-original	**disgusting_*_***, **soggy_*_***, disappointing_*_*, *_too_bad, *_would_never, it_is_not, **rude_*_***, *_*_terrible, **place_is_not**, disappointment_*_*	*_*_great, good_*_*, *_*_best, *_i_love, *_*_amazing, **delicious_*_***, **back_*_***, *_i_highly, of_my_favorite
Joint-auxiliary	**soggy_*_***, **disgusting_*_***, **rude_*_***, disappointment_*_*, **not_go_back**, **was_not_fresh**, prices_*_*, **inedible_!_***, **oily_*_***, overpriced_*_*	**delicious_*_***, **go_back_***, **is_always_fresh**, **friendly_*_***, **to_die_for**, also_very_UNK, of_my_favorite, food_*_*, *_i_highly, **delicious_!_***

Table 4: Comparison of the most useful trigrams chosen by our method and by NaiveNN on $CR \rightarrow RT$. Here * denotes a "padding", which we added at the beginning and the end of each sentence. The domain-specific sentiment words are in **bold**.

task, it might bring in some noise and affect the performance of our sequential model. In contrast to **Sequential-auxiliary**, **Joint-auxiliary** is jointly learnt with the sentiment classification task, and it is easy to see that most of its extracted trigrams are target-specific sentiment words. Also, for **Joint-original**, since we share the word embeddings of two components and do not remove any pivot, it is intuitive to see that the extracted trigrams contain both domain-independent and domain-specific sentiment words. These observations agree with our motivations behind the model.

Finally, we also sample several sentences from the test dataset, i.e., *RT*, to get a deeper insight of our joint model. Although **NaiveNN** and **Sequential** correctly predict sentiments of the following two sentences:

1. "*I've also been amazed at all the new additions in the past few years: A new Jazz Bar, the most fantastic Dining Garden, the Best Thin Crust Pizzas, and now a Lasagna Menu which is **to die for**!*"

2. "*The have a great cocktail with Citrus Vodka and lemon and lime juice and mint leaves that is **to die for**!*"

Both of them give wrong predictions on another three sentences containing ***to die for***:

3. "*Try their chef's specials– they are **to die for**.*"

4. "*Their tuna tartar appetizer is **to die for**.*"

5. "*It's **to die for**!*".

However, since ***to die for*** co-occurs with some general sentiment words like *fantastic*, *best* and *great* in previous two sentences, our joint model can implicitly learn that ***to die for*** is highly correlated with the positive sentiment via our auxiliary tasks, and ultimately make correct predictions for the latter three sentences. This further indicates that our joint model can identify more domain-specific sentiment words in comparison with **NaiveNN** and **Sequential**, and therefore improve the performance.

5 Conclusions

We presented a domain adaptation method for sentiment classification based on sentence embeddings. Our method induces a sentence embedding that works well across domains, based on two auxiliary tasks. We also jointly learn the cross-domain sentence embedding and the sentiment classifier. Experiment results show that our proposed joint method can outperform several highly competitive domain adaptation methods on 18 source-target pairs using five benchmark data sets. Moreover, further analysis confirmed that our method is able to pick up domain-specific sentiment words.

Acknowledgment

This research is supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 132–141. Association for Computational Linguistics.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 730–740, Beijing, China, July. Association for Computational Linguistics.

Danushka Bollegala, Tingting Mu, and John Goulermas. 2016. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge & Data Engineering*, 6(2):398–410.

Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014a. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014b. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 689–697.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In Proceedings of the Twenty-eight International Conference on Machine Learning*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, October.

Shoushan Li, Rui Xia, Chengqing Zong, and Chu-Ren Huang. 2009. A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 692–700. Association for Computational Linguistics.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 410–419. Association for Computational Linguistics.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In

Proceedings of the 19th international conference on World wide web, pages 751–760. ACM.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 655–665. Association for Computational Linguistics.

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

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September. Association for Computational Linguistics.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 62–71.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 538–544.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 672–682.

Jianfei Yu and Jing Jiang. 2015. A hassle-free unsupervised domain adaptation method using instance similarity features. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 168–173, Beijing, China, July. Association for Computational Linguistics.

Attention-based LSTM Network for Cross-Lingual Sentiment Classification

Xinjie Zhou, **Xiaojun Wan** and **Jianguo Xiao**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{zhouxinjie,wanxiaojun,xiaojianguo}@pku.edu.cn

Abstract

Most of the state-of-the-art sentiment classification methods are based on supervised learning algorithms which require large amounts of manually labeled data. However, the labeled resources are usually imbalanced in different languages. Cross-lingual sentiment classification tackles the problem by adapting the sentiment resources in a resource-rich language to resource-poor languages. In this study, we propose an attention-based bilingual representation learning model which learns the distributed semantics of the documents in both the source and the target languages. In each language, we use Long Short Term Memory (LSTM) network to model the documents, which has been proved to be very effective for word sequences. Meanwhile, we propose a hierarchical attention mechanism for the bilingual LSTM network. The sentence-level attention model learns which sentences of a document are more important for determining the overall sentiment while the word-level attention model learns which words in each sentence are decisive. The proposed model achieves good results on a benchmark dataset using English as the source language and Chinese as the target language.

1 Introduction

Most of the sentiment analysis research focuses on sentiment classification which aims to determine whether the users attitude is positive, neutral or negative. There are two classes of mainstreaming sentiment classification algorithms: unsupervised methods which usually require a sentiment lexicon (Taboada et al., 2011) and supervised methods (Pang et al., 2002) which require manually labeled data. However, both of these sentiment resources are unbalanced in different languages. The sentiment lexicon or labeled data are rich in several languages such as English and are poor in others. Manually building these resources for all the languages will be expensive and time-consuming. Cross-lingual sentiment classification tackles the problem by trying to adapt the resources in one language to other languages. It can also be regarded as a special kind of cross-lingual text classification task.

Recently, there have been several bilingual representation learning methods such as (Hermann and Blunsom, 2014; Gouws et al., 2014) for cross-lingual sentiment or text classification which achieve promising results. They try to learn a joint embedding space for different languages such that the training data in the source language can be directly applied to the test data in the target language. However, most of the studies only use simple functions, e.g. arithmetic average, to synthesize representations for larger text sequences. Some of them use more complicated compositional models such as the bi-gram non-linearity model in (Hermann and Blunsom, 2014) which also fail to capture the long distance dependencies in texts.

In this study, we propose an attention-based bilingual LSTM network for cross-lingual sentiment classification. LSTMs have been proved to be very effective to model word sequences and are powerful to learn on data with long range temporal dependencies. After translating the training data into the target language using machine translation

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 247–256,
Austin, Texas, November 1-5, 2016.

tools, we use the bidirectional LSTM network to model the documents in both of the source and the target languages. The LSTMs show strong ability to capture the compositional semantics for the bilingual texts in our experiments.

For the traditional LSTM network, each word in the input document is treated with equal importance, which is reasonable for traditional text classification tasks. In this paper, we propose a hierarchical attention mechanism which enables our model to focus on certain part of the input document. The motivation mainly comes from the following three observations: 1) the machine translation tool that we use to translate the documents will always introduce much noise for sentiment classification. We hope that the attention mechanism can help to filter out these noises. 2) In each individual language, the sentiment of a document is usually decided by a relative small part of it. In a long review document, the user might discuss both the advantages and disadvantages of a product. The sentiment will be confusing if we consider each sentence of the same contribution. For example, in the first review of Table 1, the first sentence reveals a negative sentiment towards the movie but the second one reveals a positive sentiment. As human readers, we can understand that the review is expressing a positive overall sentiment but it is hard for the sequence modeling algorithms including LSTM to capture. 3) At the sentence level, it is important to focus on the sentiment signals such as the sentiment words. They are usually very decisive to determine the polarity even for a very long sentence, e.g. "easy" and "nice" in the second example of Table 1.

"I felt it could have been a lot better with a little less comedy and a little more drama to get the point across. *However, its still a must see for any Jim Carrey fan.* "
"It is *easy* to read, it is *easy* to look things up in and provides a *nice* section on the treatments."

Table 1: Examples of the sentiment attention

In sum, the main contributions of this study are summarized as follows:

1) We propose a bilingual LSTM network for cross-lingual sentiment classification. Compared to the previous methods which only use weighted or arithmetic average of word embeddings to represent the document, LSTMs have obvious advantage to model the compositional semantics and to capture the long distance dependencies between words for bilingual texts.

2) We propose a hierarchical bilingual attention mechanism for our model. To the best of our knowledge, this is the first attention-based model designed for cross-lingual sentiment analysis.

3) The proposed framework achieves good results on a benchmark dataset from a cross-language sentiment classification evaluation. It outperforms the best team in the evaluation as well as several strong baseline methods.

2 Related Work

Sentiment analysis is the field of studying and analyzing peoples opinions, sentiments, evaluations, appraisals, attitudes, and emotions (Liu, 2012). The most common task of sentiment analysis is polarity classification which arises with the emergence of customer reviews on the Internet. Pang et al. (2002) used supervised learning methods and achieved promising results with simple unigram and bi-gram features. In subsequent research, more features and learning algorithms were tried for sentiment classification by a large number of researchers. Recently, the emerging of deep learning has also shed light on this area. Lots of representation learning methods has been proposed to address the sentiment classification task and many of them achieve the state-of-the-art performance on several benchmark datasets, such as the recursive neural tensor network (Socher et al., 2013), paragraph vector (Le and Mikolov, 2014), multi-channel convolutional neural networks (Kim, 2012), dynamic convolutional neural network (Blunsom et al., 2014) and tree structure LSTM (Tai et al., 2015). Very recently, Yang et al. (2016) proposed a similar hierarchical attention network based on GRU in the monolingual setting. Note that our work is independent with theirs and their study was released online after we submitted this study.

Cross-lingual sentiment classification is also a popular research topic in the sentiment analysis

community which aims to solve the sentiment classification task from a cross-language view. It is of great importance since it can exploit the existing labeled information in a source language to build a sentiment classification system in any other target language. Cross-lingual sentiment classification has been extensively studied in the very recent years. Mihalcea et al. (2007) translated English subjectivity words and phrases into the target language to build a lexicon-based classifier. Wan (2009) translated both the training data (English to Chinese) and the test data (Chinese to English) to train different models in both the source and target languages. Chen et al. (2015) proposed a knowledge validation method and incorporated it into a boosting model to transfer credible information between the two languages during training.

There have also been several studies addressing the task via multi-lingual text representation learning. Xiao and Guo (2013) learned different representations for words in different languages. Part of the word vector is shared among different languages and the rest is language-dependent. Klementiev et al. (2012) treated the task as a multi-task learning problem where each task corresponds to a single word, and the task relatedness is derived from co-occurrence statistics in bilingual parallel corpora. Chandar A P et al. (2014) and Zhou et al. (2015) used the autoencoders to model the connections between bilingual sentences. It aims to minimize the reconstruction error between the bag-of-words representations of two parallel sentences. Pham et al. (2015) extended the paragraph model into bilingual setting. Each pair of parallel sentences shares the same paragraph vector.

Compared to the existing studies, we propose to use the bilingual LSTM network to learn the document representations of reviews in each individual language. It has obvious advantage to model the compositional semantics and to capture the long distance dependencies between words. Besides, we propose a hierarchical neural attention mechanism to capture the sentiment attention in each document. The attention model helps to filter out the noise which is irrelevant to the overall sentiment.

3 Preliminaries

3.1 Problem Definition

Cross-language sentiment classification aims to use the training data in the source language to build a model which is adaptable for the test data in the target language. In our setting, we have labeled training data in English $L_{EN} = \{x_i, y_i\}_{i=1}^{N}$, where x_i is the review text and y_i is the sentiment label vector. $y_i = (1, 0)$ represents the positive sentiment and $y_i = (0, 1)$ represents the negative sentiment. In the target language Chinese, we have the test data $T_{CN} = \{x_i\}_{i=1}^{T}$ and unlabeled data $U_{CN} = \{x_i\}_{i=1}^{M}$. The task is to use L_{EN} and U_{CN} to learn a model and classify the sentiment polarity for the review texts in T_{CN}.

In our method, the labeled, unlabeled and test data are all translated into the other language using an online machine translation tool. In the subsequent part of the paper, we refer to a document and its corresponding translation in the other language as a pair of parallel documents.

3.2 RNN and LSTM

Recurrent neural network (RNN) (Rumelhart et al., 1988) is a special kind of feed-forward neural network which is useful for modeling time-sensitive sequences. At each time *t*, the model receives input from the current example and also from the hidden layer of the network's previous state. The output is calculated given the hidden state at that time stamp. The recurrent connection makes the output at each time associated with all the previous inputs. The vanilla RNN model has been considered to be difficult to train due to the well-known problem of vanishing and exploding gradients. The LSTM (Hochreiter and Schmidhuber, 1997) addresses the problem by re-parameterizing the RNN model. The core idea of LSTM is introducing the "gates" to control the data flow in the recurrent neural unit. The LSTM structure ensures that the gradient of the long-term dependencies cannot vanish. The detailed architecture that we use in shown in Figure 1.

4 Framework

In this study, we try to model the bilingual texts through the attention based LSTM network. We first

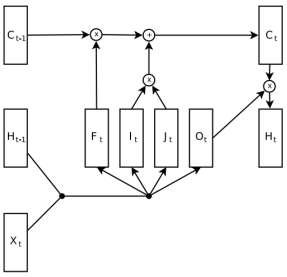


Figure 1: The LSTM architecture. The image is adopted from (Jozefowicz et al., 2015).

describe the general architecture of the model and then describe the attention mechanism used in it.

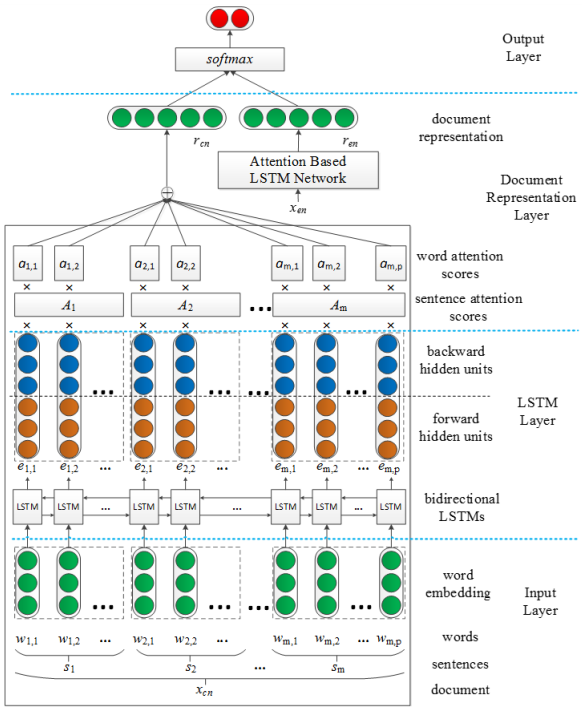


Figure 2: The architecture of the proposed framework. The inputs x_{cn} and x_{en} are parallel documents. Due to space limit, we only illustrate the attention based LSTM network in Chinese language. For the English document x_{en}, the network architecture is the same as the Chinese side but has different model parameters.

4.1 Architecture

The general architecture of our approach is shown in Figure 2. For a pair of parallel documents x_{cn} and x_{en}, each of them is sent into the attention based LSTM network. The English-side and Chinese-side architectures are the same but have different parameters. We only show the Chinese-side network in the figure due to space limit. The whole model is divided into four layers. In the input layer, the documents are represented as a word sequence where each position corresponds to a word vector from pre-trained word embeddings. In the LSTM layer, we get the high-level representation from a bidirectional LSTM network. We use the hidden units from both the forward and backward LSTMs. In the document representation layer, we incorporate the attention model into the network and derive the final document representation. At the output layer, we concatenate the representations of the English and Chinese documents and use the softmax function to predict the sentiment label.

Input Layer: The input layer of the network is the word sequences in a document x which can be either Chinese or English. The document x contains several sentences $\{s_i\}_{i=1}^{|x|}$ and each sentence is composed of several words $s_i = \{w_{i,j}\}_{j=1}^{|s_i|}$. We represent each word in the document as a fixed-size vector from pre-trained word embeddings.

LSTM Layer: In each individual language, we use bi-directional LSTMs to model the input sequences. In the bidirectional architecture, there are two layers of hidden nodes from two separate LSTMs. The two LSTMs capture the dependencies in different directions. The first hidden layers have recurrent connections from the past words while second one's direction of recurrent of connections is flipped, passing activation backwards in the texts. Therefore, in the LSTM layer, we can get the forward hidden state $\overrightarrow{h}_{i,j}$ from the forward LSTM network and the backward hidden state $\overleftarrow{h}_{i,j}$ from the backward LSTM network. We represent the final state at position (i, j), i.e. the j-th word in the i-th sentence of the document, with the concatenation of $\overrightarrow{h}_{i,j}$ and $\overleftarrow{h}_{i,j}$.

$$h_{i,j} = \overrightarrow{h}_{i,j} \parallel \overleftarrow{h}_{i,j}$$

It captures the compositional semantics in both directions of the word sequences.

Document Representation Layer: As described above, different parts of the document usually have different importance for the overall sentiment. Some

sentences or words can be decisive while the others are irrelevant. In this study, we use a hierarchical attention mechanism which assigns a real value score for each word and a real value score for each sentence. The detailed strategy of our attention model will be described in the next subsection.

Suppose we have the sentence attention score A_i for each sentence $s_i \in x$, and the word attention score $a_{i,j}$ for each word $w_{i,j} \in s_i$, both of the scores are normalized which satisfy the following equations,

$$\sum_i A_i = 1 \quad and \quad \sum_j a_{i,j} = 1$$

The sentence attention measures which sentence is more important for the overall sentiment while the word attention captures sentiment signals such as sentiment words in each sentence. Therefore, the document representation r for document x is calculated as follows,

$$r = \sum_i [A_i \cdot \sum_j (a_{i,j} \cdot h_{i,j})]$$

Note that many LSTM based models represent the word sequences only using the hidden layer at the final node. In this study, the hidden states at all the positions are considered with different attention weights. We believe that, for document sentiment classification, focusing on some certain parts of the document will be effective to filter out the sentiment-irrelevant noise.

Output Layer: At the output layer, we need to predict the overall sentiment of the document. For each English document x_{en} and its corresponding translation x_{cn}, suppose the document representations of them are obtained in previous steps as r_{en} and r_{cn}, we simply concatenate them as the feature vector and use the softmax function to predict the final sentiment.

$$\hat{y} = softmax(r_{cn} \parallel r_{en})$$

4.2 Hierarchical Attention Mechanism

For document-level sentiment classification task, we have shown that capturing both the sentence and word level attention is important. The general idea is inspired by previous works such as Bahdanau et al. (2014) and Hermann et al. (2015) which have successfully applied the attention model to machine translation and question answering. Bahdanau et al. (2014) incorporated the attention model into the sequence to sequence learning framework. During the decoding phase of the machine translation task, the attention model helps to find which input word should be "aligned" to the current output. In our case, the output of the model is not a sequence but only one sentiment vector. We hope to find the important units in the input sequence which are influential for the output.

We propose to learn a hierarchical attention model jointly with the bilingual LSTM network. The first level is the sentence attention model which measures which sentences are more important for the overall sentiment of a document. For each sentence $s_i = \{w_{i,j}\}_{j=1}^{|s_i|}$ in the document, we represent the sentence via the final hidden state of the forward LSTM and the backward LSTM, i.e.

$$s_i = \vec{h}_{i,|s_i|} \parallel \overleftarrow{h}_{i,1}$$

We use a two-layer feed-forward neural network to predict the attention score of s_i

$$\hat{A}_i = f(s_i; \theta_s)$$

$$A_i = \frac{\exp(\hat{A}_i)}{\sum_j \exp(\hat{A}_j)}$$

where f denotes the two-layer feed-forward neural network and θ_s denotes the parameters in it.

At the word level, we represent each word $w_{i,j}$ using its word embedding and the hidden state of the bidirectional LSTM layer, i.e. $h_{i,j}$. Similarly, we use a two-layer feed forward neural network to predict the attention score of $w_{i,j}$,

$$e_{i,j} = w_{i,j} \parallel \vec{h}_{i,j} \parallel \overleftarrow{h}_{i,j}$$

$$\hat{a}_{i,j} = f(e_{i,j}; \theta_w)$$

$$a_{i,j} = \frac{\exp(\hat{a}_{i,j})}{\sum_j \exp(\hat{a}_{i,j})}$$

where θ_w denotes the parameters for predicting word attention.

4.3 Training of the Proposed Model

The proposed model is trained in a semi-supervised manner. In the supervised part, we use the cross entropy loss to minimize the sentiment prediction error between the output results and the gold standard labels,

$$L_1 = \sum_{(x_{en}, x_{cn})} \sum_i -y_i \log(\hat{y}_i)$$

where x_{en} and x_{cn} are a pair of parallel documents in the training data, y is the gold-standard sentiment vector and $\hat{y}$ is the predicted vector from our model.

The unsupervised part tries to minimize the document representations between the parallel data. Following previous research, we simply measure the distance of two parallel documents via the Euclidean Distance,

$$L_2 = \sum_{(x_{en}, x_{cn})} \|r_{en} - r_{cn}\|^2$$

where x_{en} and x_{cn} are a pair of parallel documents from both the labeled and unlabeled data.

The final objective function is a weighted sum of L_1 and L_2,

$$L = L_1 + \alpha \cdot L_2$$

where α is the hyper-parameter controlling the weight. We use Adadelta (Zeiler, 2012) to update the parameters during training. It can dynamically adapt over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent.

In the test phase, the test document in T_{CN} is sent into our model along with the corresponding machine translated text in T_{EN}. The final sentiment is predicted via a softmax function over the concatenated representation of the bilingual texts as described above.

5 Experiment

5.1 Dataset

We use the dataset from the cross-language sentiment classification evaluation of NLP&CC 2013.[1] The dataset contains reviews in three domains including book, DVD and music. In each domain, it has 2000 positive reviews and 2000 negative reviews in English for training and 4000 Chinese reviews for test. It also contains 44113, 17815 and 29678 unlabeled reviews for book, DVD and music respectively.

5.2 Implementation Detail

We use Google Translate[2] to translate the labeled data to Chinese and translate the unlabeled data and test data to English. All the texts are tokenized and converted into lower case.

In the proposed framework, the dimensions of the word vectors and the hidden layers of LSTMs are set as 50. The initial word embeddings are trained on both the unlabeled and labeled reviews using `word2vec` in each individual language. The word vectors are fine-tuned during the training procedure. The hyper-parameter a is set to 0.2. The dropout rate is set to 0.5 to prevent overfitting. Ten percent of the training data are randomly selected as validation set. The training procedure is stopped when the prediction accuracy does not improve for 10 iterations. We implement the framework based on theano (Bastien et al., 2012) and use a GTX 980TI graphic card for training.

5.3 Baselines and Results

To evaluate the performance of our model, we compared it with the following baseline methods:

LR and **SVM:** We use logistic regression and SVM to learn different classifiers based on the translated Chinese training data. We simply use unigram features.

MT-PV: Paragraph vector (Le and Mikolov, 2014) is considered as one of the state-of-the-art monolingual document modeling methods. We translate all the training data into Chinese and use paragraph vector to learn a vector representation for the training and test data. A logistic regression classifier is used to predict the sentiment polarity.

Bi-PV: Pham et al. (2015) is one the state-of-the-art bilingual document modeling methods. It extends the paragraph vector into bilingual setting.

[1] The dataset can be found at http://tcci.ccf.org.cn/conference/2013/index.html. NLP&CC is an annual conference specialized in the fields of Natural Language Processing (NLP) and Chinese Computing (CC) organized by Chinese Computer Federation (CCF).

[2] http://translate.google.com/

Each pair of parallel sentences in the training data shares the same vector representation.

BSWE: Zhou et al. (2015) proposed the bilingual sentiment word embedding algorithm based on denoising autoencoders. It learns the vector representations for 2000 sentiment words. Each document is then represented by the sentiment words and the corresponding negation words in it.

H-Eval: Gui et al. (2013) got the highest performance in the NLP&CC 2013 cross-lingual sentiment classification evaluation. It uses a mixed CLSC model by combining co-training and transfer learning strategies.

A-Eval: This is the average performance of all the teams in the NLP&CC 2013 cross-lingual sentiment classification evaluation.

The attention-based models **EN-Attention, CN-Attention** and **BI-Attention**: Bi-Attention is the model described in the above sections which concatenate the document representations of the English side and the Chinese side texts. EN-Attention only translates the Chinese test data into English and uses English-side attention model while CN-Attention only uses the Chinese side attention model.

Method	Domains			Average
	book	DVD	music	
LR	0.765	0.796	0.741	0.767
SVM	0.779	0.814	0.707	0.767
MT-PV	0.753	0.799	0.748	0.766
Bi-PV	0.785	0.820	0.753	0.796
BSWE	0.811	0.816	0.794	0.807
A-Eval	0.662	0.660	0.675	0.666
H-Eval	0.785	0.777	0.751	0.771
EN-Attention	0.798	0.827	0.808	0.811
CN-Attention	0.820	**0.840**	0.809	0.823
BI-Attention	**0.821**	0.837	**0.813**	**0.824**

Table 2: Cross-lingual sentiment prediction accuracy of our methods and the comparison approaches.

Table 2 shows the cross-lingual sentiment classification accuracy of all the approaches. The first kind baseline algorithms are based on traditional bag-of-word features. SVM performs better than LR on book and DVD but gets much worse result on music. The second kind baseline algorithms are based on deep learning methods which learn the vector representations for words or documents. MT-PV achieves similar results with LR. Bi-PV improves the accuracy by about 0.03 using both the bilingual documents. While MT-PV and Bi-PV directly learn document representations, BSWE learns the embedding for the words in a bilingual sentiment lexicon. It gets higher accuracy than both Bi-PV and MT-PV which shows that the sentiment words are very important for this task.

Our attention based models achieve the highest prediction accuracy among all the approaches. The results show that CN-Attention always outperforms EN-Attention. The combination of the English-side and Chinese-side model brings improvement to both the book and music domains and yields the highest average prediction accuracy. The attention-based models outperform the algorithms using traditional features as well as the existing deep learning based methods. Compared to the highest performance in the NLP&CC evaluation, we improve the average accuracy by about 0.05.

5.4 Influence of the Attention Mechanism

In this study, we propose a hierarchical attention mechanism to capture the sentiment-related information of each document. In table 3, we show the results of models with different attention mechanisms. All the models are based on the bilingual bi-directional LSTM network as shown in Figure 2. LSTM is the basic bilingual bi-directional LSTM network. LSTM+SA considers only sentence-level attention while LSTM+WA considers only word-level attention. LSTM+HA combines both word-level and sentence-level attentions. From the results, we can observe that LSTM+HA outperforms the other three methods, which proves the effectiveness of the hierarchical attention mechanism. Besides, the word-level attention shows better performance than the sentence-level attention.

Method	Average Accuracy
LSTM	0.811
LSTM+SA	0.814
LSTM+WA	0.821
LSTM+HA	0.824

Table 3: Comparison of different attention mechanisms

We also conduct a case study using the examples in Table 1. We show the visualized word attention

using a heat map in Figure 3 by drawing the attention of each word in it. The darker color reveals higher attention scores while the lighter part has little importance. We can observe that our model successfully identifies the important units of the sentence. The sentiment word "easy" gets much higher attention score than the other words. The word "nice" gets the third highest score in the sentence right after the two "easy". Note that our attention mechanism considers both the word embedding vector and the hidden state vectors. Therefore, the same word "easy" gets different scores in different positions.

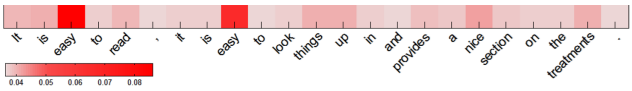


Figure 3: Attention visualization for a review sentence

5.5 Influence of the Word Embeddings

For the deep learning based methods, the initial word embeddings used as the inputs for the network usually play an important role. We study four different settings called *rand*, *static*, *fine-tuned* and *multi-channel*, respectively. In *rand* setting, the word embeddings are randomly initialized. The *static* setting keeps initial embedding fixed while the *fine-tuned* setting learns a refined embedding during the training procedure. *Multi-channel* is the combination of *static* and *fine-tuned*. Two same word vectors are concatenated to represent each word. During the training procedure, half of it is fine-tuned while the rest is fixed. Note that *fine-tuned* is the embedding setting that we use in our model.

Embedding Settings	Domains			Average
	book	DVD	music	
rand	0.789	0.786	0.746	0.774
static	0.804	0.810	0.784	0.799
fine-tuned	0.821	0.837	0.813	0.824
multi-channel	0.822	0.835	0.806	0.821

Table 4: Performance of our model with four different word embedding settings

Table 4 shows the performance of our model in these settings. *Rand* gets the lowest accuracy among them. The *fine-tuned* word embeddings perform better than *static* which fits the results in previous study (Kim, 2012). *Multi-channel* gets similar results with *fine-tuned* on DVD and music but is a bit lower on book. We also find that using pre-trained word embeddings helps the model to converge much faster than random initialization.

5.6 Influence of Vector Sizes

In our experiment, we set the size of the hidden layers in both the forward and backward LSTMs the same as the size of the input word vectors. Therefore, the dimension of the document representation is twice of the word vector size. In Figure 4, we show the performance of our model with different input vector sizes. We use the vector size in the following set $\{10, 25, 50, 100, 150, 200\}$. Note that the dimensions of all the units in the model also change with that.

We can observe from Figure 4 that the prediction accuracy for the book domain keeps steady when the vector size changes. For DVD and music, the performance increases at the beginning and becomes stable after the vector size grows larger than 50. It shows that our model is robust to a wide range of vector sizes.

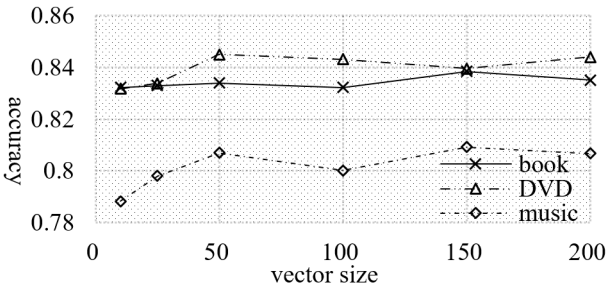


Figure 4: Performance with different vector sizes

6 Conclusion

In this paper, we propose an attention based LSTM network for cross-language sentiment classification. We use the bilingual bi-directional LSTMs to model the word sequences in the source and target languages. Based on the special characteristics of the sentiment classification task, we propose a hierarchical attention model which is jointly trained with the LSTM network. The sentence level attention

enables us to find the key sentences in a document and the word level attention helps to capture the sentiment signals. The proposed model achieves promising results on a benchmark dataset using Chinese as the source language and English as the target language. It outperforms the best results in the NLPC&CC cross-language sentiment classification evaluation as well as several strong baselines. In future work, we will evaluate the performance of our model on more datasets and more language pairs. The sentiment lexicon is also another kind of useful resource for classification. We will explore how to make full usages of these resources in the proposed framework.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61331011), National Hi-Tech Research and Development Program (863 Program) of China (2015AA015403, 2014AA015102) and IBM Global Faculty Award Program. We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

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

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Qiang Chen, Wenjie Li, Yu Lei, Xule Liu, and Yanxiang He. 2015. Learning to adapt credible knowledge in cross-lingual sentiment analysis. In *Proceedings of 52rd Annual Meeting of the Association for Computational Linguistic*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*.

Lin Gui, Ruifeng Xu, Jun Xu, Li Yuan, Yuanlin Yao, Jiyun Zhou, Qiaoyun Qiu, Shuwei Wang, Kam-Fai Wong, and Ricky Cheung. 2013. A mixed model for cross lingual opinion analysis. In *Natural Language Processing and Chinese Computing*, pages 93–104.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of 52rd Annual Meeting of the Association for Computational Linguistic*, pages 58–68.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Yoon Kim. 2012. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*, pages 1746–1751.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1759–1774.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

B Liu. 2012. Sentiment analysis and opinion mining: Synthesis lectures on human language technologies, vol. 16. *Morgan & Claypool Publishers, San Rafael*.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Hieu Pham, Minh-Thang Luong, and Christopher D Manning. 2015. Learning distributed representations for multilingual text sequences. In *Proceedings of NAACL-HLT*, pages 88–94.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2013. Semi-supervised representation learning for cross-lingual text classification. In *Proceedings of EMNLP 2013*, pages 1465–1475.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of 52rd Annual Meeting of the Association for Computational Linguistic*, pages 430–440.

Neural *versus* Phrase-Based Machine Translation Quality: a Case Study

Luisa Bentivogli
FBK, Trento
Italy

Arianna Bisazza
University of Amsterdam
The Netherlands

Mauro Cettolo
FBK, Trento
Italy

Marcello Federico
FBK, Trento
Italy

Abstract

Within the field of Statistical Machine Translation (SMT), the neural approach (NMT) has recently emerged as the first technology able to challenge the long-standing dominance of phrase-based approaches (PBMT). In particular, at the IWSLT 2015 evaluation campaign, NMT outperformed well established state-of-the-art PBMT systems on English-German, a language pair known to be particularly hard because of morphology and syntactic differences. To understand in what respects NMT provides better translation quality than PBMT, we perform a detailed analysis of neural *vs.* phrase-based SMT outputs, leveraging high quality post-edits performed by professional translators on the IWSLT data. For the first time, our analysis provides useful insights on what linguistic phenomena are best modeled by neural models – such as the reordering of verbs – while pointing out other aspects that remain to be improved.

1 Introduction

The wave of neural models has eventually reached the field of Statistical Machine Translation (SMT). After a period in which Neural MT (NMT) was too computationally costly and resource demanding to compete with state-of-the-art Phrase-Based MT (PBMT)[1], the situation changed in 2015. For the first time, in the latest edition of IWSLT[2] (Cettolo et al., 2015), the system described in (Luong and Manning, 2015) overtook a variety of PBMT approaches with a large margin ($+5.3$ BLEU points) on a difficult language pair like English-German – anticipating what, most likely, will be the new NMT era.

This impressive improvement follows the distance reduction previously observed in the WMT 2015 shared translation task (Bojar et al., 2015). Just few months earlier, the NMT systems described in (Jean et al., 2015b) ranked on par with the best phrase-based models on a couple of language pairs. Such rapid progress stems from the improvement of the recurrent neural network encoder-decoder model, originally proposed in (Sutskever et al., 2014; Cho et al., 2014b), with the use of the attention mechanism (Bahdanau et al., 2015). This evolution has several implications. On one side, NMT represents a simplification with respect to previous paradigms. From a management point of view, similar to PBMT, it allows for a more efficient use of human and data resources with respect to rule-based MT. From the architectural point of view, a large recurrent network trained for end-to-end translation is considerably simpler than traditional MT systems that integrate multiple components and processing steps. On the other side, the NMT process is less transparent than previous paradigms. Indeed, it represents a further step in the evolution from rule-based approaches that explicitly manipulate knowledge, to the statistical/data-driven framework, still comprehensible in its inner workings, to a sub-symbolic framework in which the translation process is totally opaque to the analysis.

What do we know about the strengths of NMT

[1] We use the generic term phrase-based MT to cover standard phrase-based, hierarchical and syntax-based SMT approaches.

[2] International Workshop on Spoken Language Translation (http://workshop2015.iwslt.org/)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 257–267,
Austin, Texas, November 1-5, 2016.

and the weaknesses of PBMT? What are the linguistic phenomena that deep learning translation models can handle with such greater effectiveness? To answer these questions and go beyond poorly informative BLEU scores, we perform the very first comparative analysis of the two paradigms in order to shed light on the factors that differentiate them and determine their large quality differences.

We build on evaluation data available for the IWSLT 2015 MT English-German task, and compare the results of the first four top-ranked participants. We choose to focus on one language pair and one task because of the following advantages: *(i)* three state-of-the art PBMT systems compared against the NMT system on the same data and in the very same period (that of the evaluation campaign); *(ii)* a challenging language pair in terms of morphology and word order differences; *(iii)* availability of MT outputs' post-editing done by professional translators, which is very costly and thus rarely available. In general, post-edits have the advantage of allowing for informative and detailed analyses since they directly point to translation errors. In this specific framework, the high quality data created by professional translators guarantees reliable evaluations. For all these reasons we present our study as a solid contribution to the better understanding of this new paradigm shift in MT.

After reviewing previous work (Section 2), we introduce the analyzed data and the systems that produced them (Section 3). We then present three increasingly fine levels of MT quality analysis. We first investigate how MT systems' quality varies with specific characteristics of the input, *i.e.* sentence length and type of content of each talk (Section 4). Then, we focus on differences among MT systems with respect to morphology, lexical, and word order errors (Section 5). Finally, based on the finding that word reordering is the strongest aspect of NMT compared to the other systems, we carry out a fine-grained analysis of word order errors (Section 6).

2 Previous Work

To date, NMT systems have only been evaluated by BLEU in single-reference setups (Bahdanau et al., 2015; Sutskever et al., 2014; Luong et al., 2015; Jean et al., 2015a; Gülçehre et al., 2015). Additionally, the Montreal NMT system submitted to WMT 2015 (Jean et al., 2015b) was part of a manual evaluation experiment where a large number of non-professional annotators were asked to rank the outputs of multiple MT systems (Bojar et al., 2015). Results for the Montreal system were very positive – ranked first in English-German, third in German-English, English-Czech and Czech-English – which confirmed and strengthened the BLEU results published so far. Unfortunately neither BLEU nor manual ranking judgements tell us which translation aspects are better modeled by different MT frameworks. To this end, a detailed and systematic error analysis of NMT *vs.* PBMT output is required.

Translation error analysis, as a way to identify systems' weaknesses and define priorities for their improvement, has received a fair amount of attention in the MT community. In this work we opt for the *automatic* detection and classification of translation errors based on *manual* post-edits of the MT output. We believe this choice provides an optimal trade-off between fully manual error analysis (Farrús Cabeceran et al., 2010; Popović et al., 2013; Daems et al., 2014; Federico et al., 2014; Neubig et al., 2015), which is very costly and complex, and fully automatic error analysis (Popović and Ney, 2011; Irvine et al., 2013), which is noisy and biased towards one or few arbitrary reference translations.

Existing tools for translation error detection are either based on Word Error Rate (WER) and Position-independent word Error Rate (PER) (Popović, 2011) or on output-reference alignment (Zeman et al., 2011). Regarding error classification, Hjerson (Popović, 2011) detects five main types of word-level errors as defined in (Vilar et al., 2006): morphological, reordering, missing words, extra words, and lexical choice errors. We follow a similar but simpler error classification (morphological, lexical, and word order errors), but detect the errors differently using TER as this is the most natural choice in our evaluation framework based on post-edits (see also Section 3.4). Irvine et al. (2013) propose another word-level error analysis technique specifically focused on lexical choice and aimed at understanding the effects of domain differences on MT. Their error classification is strictly related to model coverage and insensitive to word order differences. The technique requires access to the sys-

tem's phrase table and is thus not applicable to NMT, which does not rely on a fixed inventory of translation units extracted from the parallel data.

Previous error analyses based on manually post-edited translations were presented in (Bojar, 2011; Koponen, 2012; Popović et al., 2013). We are the first to conduct this kind of study on the output of a neural MT system.

3 Experimental Setting

We perform a number of analyses on data and results of the IWSLT 2015 MT *En-De* task, which consists in translating manual transcripts of English TED talks into German. Evaluation data are publicly available through the WIT^3 repository (Cettolo et al., 2012).[3]

3.1 Task Data

TED Talks[4] are a collection of rather short speeches (max 18 minutes each, roughly equivalent to 2,500 words) covering a wide variety of topics. All talks have captions, which are translated into many languages by volunteers worldwide. Besides representing a popular benchmark for spoken language technology, TED Talks embed interesting research challenges. Translating TED Talks implies dealing with spoken rather than written language, which is hence expected to be structurally less complex, formal and fluent (Ruiz and Federico, 2014). Moreover, as human translations of the talks are required to follow the structure and rhythm of the English captions, a lower amount of rephrasing and reordering is expected than in the translation of written documents.

As regards the English-German language pair, the two languages are interesting since, while belonging to the same language family, they have marked differences in levels of inflection, morphological variation, and word order, especially long-range reordering of verbs.

3.2 Evaluation Data

Five systems participated in the MT *En-De* task and were manually evaluated on a representative subset of the official 2015 test set. The Human Evaluation (HE) set includes the first half of each of the 12 test talks, for a total of 600 sentences and around 10K words. Five professional translators were asked to post-edit the MT output by applying the minimal edits required to transform it into a fluent sentence with the same meaning as the source sentence. Data were prepared so that all translators equally post-edited the five MT outputs, *i.e.* 120 sentences for each evaluated system.

The resulting evaluation data consist of five new reference translations for each of the sentences in the HE set. Each one of these references represents the *targeted translation* of the system output from which it was derived, but the other four *additional translations* can also be used to evaluate each MT system. We will see in the next sections how we exploited the available post-edits in the more suitable way depending on the kind of analysis carried out.

System	Approach	Data
PBSY (Huck and Birch, 2015)	Combination: Phrase+Syntax-based GHKM string-to-tree; hierarchical + sparse lexicalized reordering models	175M/ 3.1B
HPB (Jehl et al., 2015)	Hierarchical Phrase-based source pre-ordering (dependency tree -based); re-scoring with neural LM	166M/ 854M
SPB (Ha et al., 2015)	Standard Phrase-based source pre-ordering (POS- and tree-based); re-scoring with neural LMs	117M/ 2.4B
NMT (Luong & Manning, 2015)	Recurrent neural network (LSTM) attention-based; source reversing; rare words handling	120M/ –

Table 1: MT systems' overview. Data column: size of parallel/monolingual training data for each system in terms of English and German tokens.

3.3 MT Systems

Our analysis focuses on the first four top-ranking systems, which include NMT (Luong and Manning, 2015) and three different phrase-based approaches: standard phrase-based (Ha et al., 2015), hierarchical (Jehl et al., 2015) and a combination of phrase-based and syntax-based (Huck and Birch, 2015). Table 1 presents an overview of each system, as well as figures about the training data used.[5]

The phrase+syntax-based (PBSY) system combines the outputs of a string-to-tree decoder, trained with the GHKM algorithm, with those of two stan-

[3] wit3.fbk.eu

[4] http://www.ted.com/

[5] Detailed information about training data was kindly made available by participating teams.

dard phrase-based systems featuring, among others, adapted phrase tables and language models enriched with morphological information, hierarchical lexicalized reordering models and different variations of the operational sequence model.

The hierarchical phrase-based MT (HPB) system leverages thousands of lexicalised features, data-driven source pre-ordering (dependency tree-based), word-based and class-based language models, and n-best re-scoring models based on syntactic and neural language models.

The standard phrase-based MT (SPB) system features an adapted phrase-table combining in-domain and out-domain data, discriminative word lexicon models, multiple language models (word-, POS- and class-based), data-driven source pre-ordering (POS- and constituency syntax-based), n-best re-scoring models based on neural lexicons and neural language models.

Finally, the neural MT (NMT) system is an ensemble of 8 long short-term memory (LSTM) networks of 4 layers featuring 1,000-dimension word embeddings, attention mechanism, source reversing, 50K source and target vocabularies, and out-of-vocabulary word handling. Training with TED data was performed on top of models trained with large out-domain parallel data.

With respect to the use of training data, it is worth noticing that NMT is the only system not employing monolingual data in addition to parallel data. Moreover, NMT and SPB were trained with smaller amounts of parallel data with respect to PBSY and HPB (see Table 1).

3.4 Translation Edit Rate Measures

The *Translation Edit Rate* (TER) (Snover et al., 2006) naturally fits our evaluation framework, where it traces the edits done by post-editors. Also, TER *shift* operations are reliable indicators of reordering errors, in which we are particularly interested. We exploit the available post-edits in two different ways: *(i)* for *Human-targeted TER* (HTER) we compute TER between the machine translation and its manually post-edited version (targeted reference), *(ii)* for *Multi-reference TER* (mTER), we compute TER against the closest translation among all available post-edits (*i.e.* targeted and additional references) for each sentence.

system	BLEU	HTER	mTER
PBSY	25.3	28.0	21.8
HPB	24.6	29.9	23.4
SPB	25.8	29.0	22.7
NMT	31.1*	21.1*	16.2*

Table 2: Overall results on the HE Set: BLEU, computed against the original reference translation, and TER, computed with respect to the targeted post-edit (HTER) and multiple post-edits (mTER).

Throughout sections 4 and 5, we mark a score achieved by NMT with the symbol * if this is better than the score of its best competitor at statistical significance level 0.01. Significance tests for HTER and mTER are computed by bootstrap re-sampling, while differences among proportions are assessed via one-tailed z-score tests.

4 Overall Translation Quality

Table 2 presents overall system results according to HTER and mTER, as well as BLEU computed against the original TED Talks reference translation. We can see that NMT clearly outperforms all other approaches both in terms of BLEU and TER scores. Focusing on mTER results, the gain obtained by NMT over the second best system (PBSY) amounts to 26%. It is also worth noticing that mTER is considerably lower than HTER for each system. This reduction shows that exploiting all the available post-edits as references for TER is a viable way to control and overcome post-editors variability, thus ensuring a more reliable and informative evaluation about the real overall performance of MT systems. For this reason, the two following analyses rely on mTER. In particular, we investigate how specific characteristics of input documents affect the system's overall translation quality, focusing on *(i)* sentence length and *(ii)* the different talks composing the dataset.

4.1 Translation quality by sentence length

Long sentences are known to be difficult to translate by the NMT approach. Following previous work (Cho et al., 2014a; Pouget-Abadie et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), we investigate how sentence length affects overall translation quality. Figure 1 plots mTER scores against source sentence length. NMT clearly outperforms every PBMT system in any length bin, with statistically

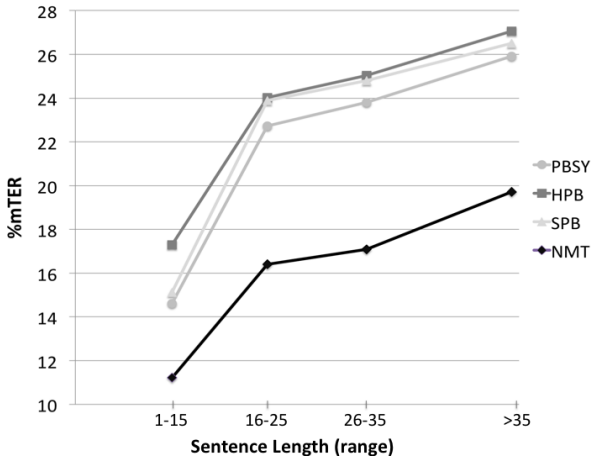


Figure 1: mTER scores on bins of sentences of different length. Points represent the average mTER of the MT outputs for the sentences in each given bin.

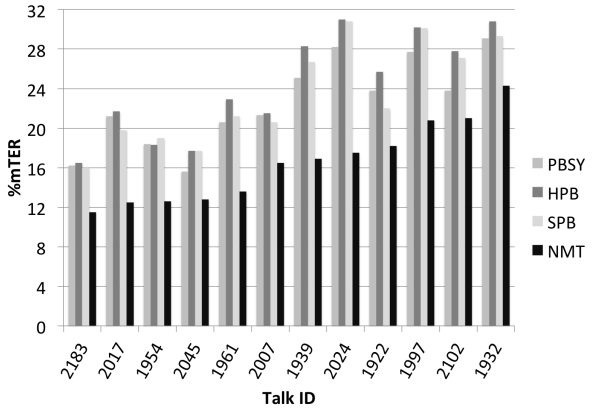


Figure 2: mTER scores per talk, sorted in ascending order of NMT scores.

significant differences. As a general tendency, the performance of all approaches worsens as sentence length increases. However, for sentences longer than 35 words we see that NMT quality degrades more markedly than in PBMT systems. Considering the percentage decrease with respect to the preceding length bin (26-35), we see that the $\%\Delta$ for NMT (-15.4) is much larger than the average $\%\Delta$ for the three PBMT systems (-7.9). Hence, this still seems an issue to be addressed for further improving NMT.

4.2 Translation quality by talk

As we saw in Section 3.1, the TED dataset is very heterogeneous since it consists of talks covering different topics and given by speakers with different styles. It is therefore interesting to evaluate translation quality also at the talk level.

Figure 2 plots the mTER scores for each of the twelve talks included in the HE set, sorted in ascending order of NMT scores. In all talks, the NMT system outperforms the PBMT systems in a statistically significant way.

We analysed different factors which could impact translation quality in order to understand if they correlate with such performance differences. We studied three features which are typically considered as indicators of complexity (see (François and Fairon, 2012) for an overview), namely *(i)* the length of the talk, *(ii)* its average sentence length, and *(iii)* the type-token ratio[6] (TTR) which – measuring lexical diversity – reflects the size of a speaker's vocabulary and the variety of subject matter in a text.

For the first two features we did not find any correlation; on the contrary, we found a moderate Pearson correlation (R=0.7332) between TTR and the mTER gains of NMT over its closest competitor in each talk. This result suggests that NMT is able to cope with lexical diversity better than any other considered approach.

5 Analysis of Translation Errors

We now turn to analyze which types of linguistic errors characterize NMT *vs.* PBMT. In the literature, various error taxonomies covering different levels of granularity have been developed (Flanagan, 1994; Vilar et al., 2006; Farrús Cabeceran et al., 2010; Stymne and Ahrenberg, 2012; Lommel et al., 2014). We focus on three error categories, namely *(i)* morphology errors, *(ii)* lexical errors, and *(iii)* word order errors. As for lexical errors, a number of existing taxonomies further distinguish among translation errors due to missing words, extra words, or incorrect lexical choice. However, given the proven difficulty of disambiguating between these three subclasses (Popović and Ney, 2011; Fishel et al., 2012), we prefer to rely on a more coarse-grained linguistic error classification where lexical errors include all of them (Farrús Cabeceran et al., 2010).

[6]The type-token-ratio of a text is calculated dividing the number of word types (vocabulary) by the total number of word tokens (occurrences).

For error analysis we rely on HTER results under the assumption that, since the targeted translation is generated by post-editing the given MT output, this method is particularly informative to spot MT errors. We are aware that translator subjectivity is still an issue (see Section 4), however in this more fine-grained analysis we prefer to focus on what a human implicitly annotated as a translation error. This particularly holds in our specific evaluation framework, where the goal is not to measure the absolute number of errors made by each system, but to compare systems with each other. Moreover, the post-edits collected for each MT output within IWSLT allow for a fair and reliable comparison since systems were equally post-edited by all translators (see Section 3.2), making all analyses uniformly affected by such variability.

5.1 Morphology errors

A morphology error occurs when a generated word form is wrong but its corresponding base form (lemma) is correct. Thus, we assess the ability of systems to deal with morphology by comparing the HTER score computed on the surface forms (*i.e.* morphologically inflected words) with the HTER score obtained on the corresponding lemmas. The additional matches counted on lemmas with respect to word forms indicate morphology errors. Thus, the closer the two HTER scores, the more accurate the system in handling morphology.

To carry out this analysis, the lemmatized (and POS tagged) version of both MT outputs and corresponding post-edits was produced with the German parser ParZu (Sennrich et al., 2013). Then, the HTER-based evaluation was slightly adapted in order to be better suited to an accurate detection of morphology errors. First, punctuation was removed since – not being subject to morphological inflection – it could smooth the results. Second, *shift* errors were not considered. A word form or a lemma that matches a corresponding word or lemma in the post-edit, but is in the wrong position with respect to it, is counted as a *shift* error in TER. Instead – when focusing on morphology – exact matches are not errors, regardless their position in the text.[7]

[7]Note that the TER score calculated by setting to 0 the cost of shifts approximates the Position-independent Error Rate (Tillmann et al., 1997).

system	HTERnoShft		
	word	lemma	%Δ
PBSY	27.1	22.5	-16.9
HPB	28.7	23.5	-18.4
SPB	28.3	23.2	-18.0
NMT	21.7*	18.7*	-13.7

Table 3: HTER ignoring shift operations computed on words and corresponding lemmas, and their % difference.

Table 3 presents HTER scores on word forms and lemmas, as well as their percentage difference which gives an indication of morphology errors. We can see that NMT generates translations which are morphologically more correct than the other systems. In particular, the %Δ for NMT (-13.7) is lower than that of the second best system (PBSY, -16.9) by 3.2% absolute points, leading to a percentage gain of around 19%. We can thus say that NMT makes at least 19% less morphology errors than any other PBMT system.

5.2 Lexical errors

Another important feature of MT systems is their ability to choose lexically appropriate words. In order to compare systems under this aspect, we consider HTER results at the lemma level as a way to abstract from morphology errors and focus only on actual lexical choice problems. The evaluation on the lemmatised version of the data performed to identify morphology errors fits this purpose, since its driving assumptions (*i.e.* punctuation can be excluded and lemmas in the wrong order are not errors) hold for lexical errors too.

The lemma column of Table 3 shows that NMT outperforms the other systems. More precisely, the NMT score (18.7) is better than the second best (PBSY, 22.5) by 3.8% absolute points. This corresponds to a relative gain of about 17%, meaning that NMT makes at least 17% less lexical errors than any PBMT system. Similarly to what observed for morphology errors, this can be considered a remarkable improvement over the state of the art.

5.3 Word order errors

To analyse reordering errors, we start by focusing on *shift* operations identified by the HTER metrics. The first three columns of Table 4 show, respectively: *(i)* the number of words generated by each system

system	#words	#shifts	%shifts	KRS
PBSY	11,517	354	3.1	84.6
HPB	11,417	415	3.6	84.3
SPB	11,420	398	3.5	84.5
NMT	11,284	173	1.5*	88.3*

Table 4: Word reordering evaluation in terms of shift operations in HTER calculation and of KRS. For each system, the number of generated words, the number of shift errors and their corresponding percentages are reported.

(ii) the number of shifts required to align each system output to the corresponding post-edit; and *(iii)* the corresponding percentage of shift errors. Notice that the *shift* error percentages are incorporated in the HTER scores reported in Table 2. We can see in Table 4 that *shift* errors in NMT translations are definitely less than in the other systems. The error reduction of NMT with respect to the second best system (PBSY) is about 50% (173 *vs.* 354).

It should be recalled that these numbers only refer to *shifts* detected by HTER, that is (groups of) words of the MT output and corresponding post-edit that are identical but occurring in different positions. Words that had to be moved and modified at the same time (for instance replaced by a synonym or a morphological variant) are not counted in HTER *shift* figures, but are detected as *substitution*, *insertion* or *deletion* operations. To ensure that our reordering evaluation is not biased towards the alignment between the MT output and the post-edit performed by HTER, we run an additional assessment using KRS – Kendall Reordering Score (Birch et al., 2010) – which measures the similarity between the source-reference reorderings and the source-MT output reorderings.[8] Being based on bilingual word alignment via the source sentence, KRS detects reordering errors also when post-edit and MT words are not identical. Also unlike TER, KRS is sensitive to the *distance* between the position of a word in the MT output and that in the reference.

Looking at the last column of Table 4, we can say that our observations on HTER are confirmed by the KRS results: the reorderings performed by NMT are much more accurate than those performed by any PBMT system.[9] Moreover, according to the approximate randomization test, KRS differences are statistically significant between NMT and all other systems, but not among the three PBMT systems.

Given the concordant results of our two quantitative analyses, we conclude that one of the major strengths of the NMT approach is its ability to place German words in the right position even when this requires considerable reordering. This outcome calls for a deeper investigation, which is carried out in the following section.

6 Fine-grained Word Order Error Analysis

We have observed that word reordering is a very strong aspect of NMT compared to PBMT, according to both HTER and KRS. To better understand this finding, we investigate whether reordering errors concentrate on specific linguistic constructions across our systems. Using the POS tagging and dependency parsing of the post-edits produced by ParZu, we classify the *shift* operations detected by HTER and count how often a word with a given POS label was misplaced by each of the systems (alone or as part of a shifted block). For each word class, we also compute the percentage order error reduction of NMT with respect to the PBMT system that has highest reordering accuracy overall, that is PBSY. Results are presented in Table 5, ranked by NMT-vs-PBSY gain. Punctuation is omitted as well as word classes that were shifted less than 10 times by all systems. Examples of salient word order error types are presented in Table 6.

The upper part of Table 5 shows that verbs are by far the most often misplaced word category in all PBMT systems – an issue already known to affect standard phrase-based SMT between German and English (Bisazza and Federico, 2013). Reordering is particularly difficult when translating *into* German, since the position of verbs in this language varies according to the clause type (*e.g.* main *vs.* subordinate). Our results show that even syntax-informed PBMT does not solve this issue. Using syntax at decoding time, as done by one of the systems combined within PBSY, appears to be a better strategy

[8] To compute the word alignments required by KRS, we used the FastAlign tool (Dyer et al., 2013).

[9] To put our results into perspective, note that Birch (2011) reports a difference of 5 KRS points between the translations of a PBMT system and those produced by four *human* translators tested against each other, in a Chinese-English experiment.

Class	NMT-vs-PBSY	NMT	PBSY	HPB	SPB
V	-70%	35	116	133	155
PRO	-57%	22	51	53	62
PTKZU	-54%	6	13	4	11
ADV	-50%	14	28	44	36
N	-47%	37	70	99	56
KON	-33%	6	9	8	12
PREP	-18%	18	22	27	28
PTKNEG	-17%	10	12	10	7
ART	-4%	26	27	38	35
aux:V	-87%	3	23	17	18
neb:V	-83%	2	12	7	19
objc:V	-79%	3	14	21	24
subj:PRO	-70%	12	40	34	46
root:V	-68%	6	19	28	27
adv:ADV	-67%	8	24	33	28
obja:N	-65%	6	17	28	12
cj:V	-59%	7	17	21	22
part:PTKZU	-54%	6	13	4	11
obja:PRO	-38%	5	8	14	7
mroot:V	-36%	7	11	26	20
pn:N	-36%	16	25	33	19
subj:N	-33%	6	9	10	7
pp:PREP	-30%	14	20	19	23
adv:PTKNEG	-17%	10	12	10	7
det:ART	-4%	26	27	38	34
all	-48%	222	429	493	488

Table 5: Main POS tags and dependency labels of words occurring in shifted blocks detected by HTER. NMT-vs-PBSY denotes the reduction of reordering errors in NMT *vs.* PBSY system. Only word classes that were shifted 10 or more times in at least one system output are shown.

than using it for source pre-ordering, as done by the HPB and SPB systems. However this only results in a moderate reduction of verb reordering errors (-12% and -25% *vs.* HPB and SPB respectively). On the contrary, NMT reduces verb order errors by an impressive -70% with respect to PBSY (-74% and -77% *vs.* HPB and SPB respectively) despite being trained on raw parallel data without any syntactic annotation, nor explicit modeling of word reordering. This result shows that the recurrent neural language model at the core of the NMT architecture is very successful at generating well-formed sentences even in languages with less predictable word order, like German (see examples in Table 6(a,b)). NMT, though, gains notably less on nouns (-47%), which is the second most often misplaced word category in PBSY. More insight on this is provided by the lower part of the table, where reordering errors are divided by their dependency label as well as POS tag. Here we see that order errors on nouns are notably reduced by NMT when they act as syntactic objects (-65% obja:N) but less when they act as preposition complements (-36% pn:N) or subjects (-33% subj:N).

The smallest NMT-*vs*-PBSY gains are observed on prepositions (-18% PREP), negation particles (-17% PTKNEG) and articles (-4% ART). Manual inspection of a data sample reveals that misplaced prepositions are often part of misplaced prepositional phrases acting, for instance, as temporal or instrumental adjuncts (e.g. *'in my life'*, *'with this video'*). In these cases, the original MT output is overall understandable and grammatical, but does not conform to the order of German semantic arguments that is consistently preferred by post-editors (see example in Table 6(c)). Articles, due to their commonness, are often misaligned by HTER and marked as *shift* errors instead of being marked as two unrelated substitutions. Finally, negation particles account for less than 1% of the target tokens but play a key role in determining the sentence meaning. Looking closely at some error examples, we found that the correct placement of the German particle *nicht* was determined by the focus of negation in the source sentence, which is difficult to detect in English. For instance in Table 6(d) two interpretations are possible (*'that did not work'* or *'that worked, but not for systematic reasons'*), each resulting in a different, but equally grammatical, location of *nicht*. In fact, negation-focus detection calls for a deep understanding of the sentence semantics, often requiring extra-sentential context (Blanco and Moldovan, 2011). When faced with this kind of translation decisions, NMT performs as poorly as its competitors.

In summary, our fine-grained analysis confirms that NMT concentrates its word order improvements on important linguistic constituents and, specifically in English-German, is very close to solving the infamous problem of long-range verb reordering which so many PBMT approaches have only poorly managed to handle. On the other hand, NMT still struggles with more subtle translation decisions depending, for instance, on the semantic ordering of adjunct prepositional phrases or on the focus of negation.

*Auxiliary-main verb construction [aux:*V*]:*			
	SRC	in this experiment , individuals **were shown** hundreds of hours of YouTube videos	
(a)	HPB	in diesem Experiment , Individuen **gezeigt wurden** Hunderte von Stunden YouTube-Videos	✗
	PE	in diesem Experiment **wurden** Individuen Hunderte von Stunden Youtube-Videos **gezeigt**	
	NMT	in diesem Experiment **wurden** Individuen hunderte Stunden YouTube Videos **gezeigt**	✓
	PE	in diesem Experiment **wurden** Individuen hunderte Stunden YouTube Videos **gezeigt**	
*Verb in subordinate (adjunct) clause [neb:*V*]:*			
	SRC	... when coaches and managers and owners **look** at this information streaming ...	
(b)	PBSY	... wenn Trainer und Manager und Eigentümer **betrachten** diese Information Streaming ...	✗
	PE	... wenn Trainer und Manager und Eigentümer dieses Informations-Streaming **betrachten** ...	
	NMT	... wenn Trainer und Manager und Besitzer sich diese Informationen **anschauen** ...	✓
	PE	... wenn Trainer und Manager und Besitzer sich diese Informationen **anschauen** ...	
Prepositional phrase [pp:PREP det:ART pn:N] acting as temporal adjunct:			
	SRC	so like many of us , I 've lived in a few closets **in my life**	
(c)	SPB	so wie viele von uns , ich habe in ein paar Schränke **in meinem Leben** gelebt	✗
	PE	so habe ich wie viele von uns **während meines Lebens** in einigen Verstecken gelebt	
	NMT	wie viele von uns habe ich in ein paar Schränke **in meinem Leben** gelebt	✗
	PE	wie viele von uns habe ich **in meinem Leben** in ein paar Schränken gelebt	
*Negation particle [adv:*PTKNEG*]:*			
	SRC	but I eventually came to the conclusion that that just did **not** work for systematic reasons	
(d)	HPB	aber ich kam schlielich zu dem Schluss , dass nur aus systematischen Gründen **nicht** funktionieren	✓
	PE	aber ich kam schlielich zu dem Schluss , dass es einfach aus systematischen Gründen **nicht** funktioniert	
	NMT	aber letztendlich kam ich zu dem Schluss , dass das einfach **nicht** aus systematischen Gründen funktionierte	✗
	PE	ich musste aber einsehen , dass das aus systematischen Gründen **nicht** funktioniert	

Table 6: MT output and post-edit examples showing common types of reordering errors.

7 Conclusions

We analysed the output of four state-of-the-art MT systems that participated in the English-to-German task of the IWSLT 2015 evaluation campaign. Our selected runs were produced by three phrase-based MT systems and a neural MT system. The analysis leveraged high quality post-edits of the MT outputs, which allowed us to profile systems with respect to reliable measures of post-editing effort and translation error types.

The outcomes of the analysis confirm that NMT has significantly pushed ahead the state of the art, especially in a language pair involving rich morphology prediction and significant word reordering. To summarize our findings: *(i)* NMT generates outputs that considerably lower the overall post-edit effort with respect to the best PBMT system (-26%); *(ii)* NMT outperforms PBMT systems on all sentence lengths, although its performance degrades faster with the input length than its competitors; *(iii)* NMT seems to have an edge especially on lexically rich texts; *(iv)* NMT output contains less morphology errors (-19%), less lexical errors (-17%), and substantially less word order errors (-50%) than its closest competitor for each error type; *(v)* concerning word order, NMT shows an impressive improvement in the placement of verbs (-70% errors).

While NMT proved superior to PBMT with respect to all error types that were investigated, our analysis also pointed out some aspects of NMT that deserve further work, such as the handling of long sentences and the reordering of particular linguistic constituents requiring a deep semantic understanding of text. Machine translation is definitely not a solved problem, but the time is finally ripe to tackle its most intricate aspects.

Acknowledgments

FBK authors were supported by the CRACKER, QT21 and ModernMT projects, which received funding from the European Union's Horizon 2020 programme under grants No. 645357, 645452 and 645487. AB was funded in part by the NWO under projects 639.022.213 and 612.001.218.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*, San Diego, US-CA.

Alexandra Birch, Miles Osborne, and Phil Blunsom. 2010. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26.

Alexandra Birch. 2011. *Reordering Metrics for Statistical Machine Translation*. Ph.D. thesis, School of Informatics, University of Edinburgh, UK.

Arianna Bisazza and Marcello Federico. 2013. Efficient solutions for word reordering in German-English phrase-based statistical machine translation. In *Proc. of WMT*, Sofia, Bulgaria.

Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proc. of ACL-HLT*, Portland, US-OR.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proc. of WMT*, Lisbon, Portugal.

Ondrej Bojar. 2011. Analyzing error types in English-Czech machine translation. *The Prague Bulletin of Mathematical Linguistic*, (95):63–76.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web Inventory of Transcribed and Translated Talks. In *Proc. of EAMT*, Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proc. of IWSLT*, Da Nang, Vietnam.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: encoder–decoder approaches. In *Proc. of SSST-8*, Doha, Qatar.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, Doha, Qatar.

Joke Daems, Lieve Macken, and Sonia Vandepitte. 2014. On the origin of errors: a fine-grained analysis of MT and PE errors and their relationship. In *Proc. of LREC*, Reykjavik, Iceland.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. of NACL-HLT*, Atlanta, US-GA.

Mireia Farrús Cabeceran, Marta Ruiz Costa-Jussà, José Bernardo Mariño Acebal, and José Adrián Rodríguez Fonollosa. 2010. Linguistic-based evaluation criteria to identify statistical machine translation errors. In *Proc. of EAMT*, Saint-Raphaël, France.

Marcello Federico, Matteo Negri, Luisa Bentivogli, and Marco Turchi. 2014. Assessing the impact of translation errors on machine translation quality with mixed-effects models. In *Proc. of EMNLP*, Doha, Qatar.

Mark Fishel, Ondrej Bojar, and Maja Popović. 2012. Terra: a collection of translation error-annotated corpora. In *Proc. of LREC*, Istanbul, Turkey.

Mary Flanagan. 1994. Error classification for MT evaluation. In *Proc. of AMTA*, Columbia, US-MD.

Thomas François and Cédrick Fairon. 2012. An "AI readability" formula for French as a foreign language. In *Proc. of EMNLP-CoNLL*, Jeju Island, Korea.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Thanh-Le Ha, Jan Niehues, Eunah Cho, Mohammed Mediani, and Alex Waibel. 2015. The KIT translation systems for IWSLT 2015. In *Proc. of IWSLT*, Da Nang, Vietnam.

Matthias Huck and Alexandra Birch. 2015. The Edinburgh machine translation systems for IWSLT 2015. In *Proc. of IWSLT*, Da Nang, Vietnam.

Ann Irvine, John Morgan, Marine Carpuat, Hal Daumé III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*, 1:429–440.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015a. On using very large target vocabulary for neural machine translation. In *Proc. of ACL-IJCNLP*, Beijing, China.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015b. Montreal neural machine translation systems for WMT15. In *Proc. of WMT*, Lisbon, Portugal.

Laura Jehl, Patrick Simianer, Julian Hitschler, and Stefan Riezler. 2015. The Heidelberg university English-German translation system for IWSLT 2015. In *Proc. of IWSLT*, Da Nang, Vietnam.

Maarit Koponen. 2012. Comparing human perceptions of post-editing effort with post-editing operations. In *Proc. of WMT*, Montréal, Canada.

Arle Lommel, Aljoscha Burchardt, Maja Popović, Kim Harris, Eleftherios Avramidis, and Hans Uszkoreit. 2014. Using a new analytic measure for the annotation and analysis of MT errors on real data. In *Proc. of EAMT*, Dubrovnik, Croatia.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proc. of IWSLT*, Da Nang, Vietnam.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, Lisbon, Portugal.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015. In *Proc. of WAT2015*, Kyoto, Japan.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688.

Maja Popović, Eleftherios Avramidis, Aljoscha Burchardt, Sabine Hunsicker, Sven Schmeier, Cindy Tscherwinka, David Vilar, and Hans Uszkoreit. 2013. Learning from human judgments of machine translation output. In *Proc. of MT Summit*, Nice, France.

Maja Popović. 2011. Hjerson: an open source tool for automatic error classification of machine translation output. *The Prague Bulletin of Mathematical Linguistic*, (96):59–68.

Jean Pouget-Abadie, Dzmitry Bahdanau, Bart van Merrienboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proc. of SSST-8*, Doha, Qatar.

Nicholas Ruiz and Marcello Federico. 2014. Complexity of spoken versus written language for machine translation. In *Proc. of EAMT*, Dubrovnik, Croatia.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for German dependency parsing, POS-tagging, and morphological analysis. In *Proc. of RANLP*, Hissar, Bulgaria.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*, Boston, US-MA.

Sara Stymne and Lars Ahrenberg. 2012. On the practice of error analysis for machine translation evaluation. In *Proc. of LREC*, Istanbul, Turkey.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*, Montréal, Canada.

Christoph Tillmann, Stephan Vogel, Hermann Ney, Alexander Zubiaga, and Hassan Sawaf. 1997. Accelerated DP based search for statistical translation. In *Proc. of Eurospeech*, Rhodes, Greece.

David Vilar, Jia Xu, Luis Fernando d'Haro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proc. of LREC*, Genoa, Italy.

Daniel Zeman, Mark Fishel, Jan Berka, and Ondrej Bojar. 2011. Addicter: what is wrong with my translations? *The Prague Bulletin of Mathematical Linguistic*, (96):79–88.

Zero-Resource Translation with Multi-Lingual Neural Machine Translation

Orhan Firat[⋆]
Middle East Technical University
orhan.firat@ceng.metu.edu.tr

Baskaran Sankaran
IBM T.J. Watson Research Center

Yaser Al-onaizan
IBM T.J. Watson Research Center

Fatos T. Yarman Vural
Middle East Technical University

Kyunghyun Cho
New York University

Abstract

In this paper, we propose a novel finetuning algorithm for the recently introduced multi-way, multilingual neural machine translate that enables zero-resource machine translation. When used together with novel many-to-one translation strategies, we empirically show that this finetuning algorithm allows the multi-way, multilingual model to translate a zero-resource language pair (1) as well as a single-pair neural translation model trained with up to 1M direct parallel sentences of the same language pair and (2) better than pivot-based translation strategy, while keeping only one additional copy of attention-related parameters.

1 Introduction

A recently introduced neural machine translation (Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) has proven to be a platform for new opportunities in machine translation research. Rather than word-level translation with language-specific preprocessing, neural machine translation has found to work well with statistically segmented subword sequences as well as sequences of characters (Chung et al., 2016; Luong and Manning, 2016; Sennrich et al., 2015b; Ling et al., 2015). Also, recent works show that neural machine translation provides a seamless way to incorporate multiple modalities other than natural language text in translation (Luong et al., 2015a; Caglayan et al., 2016). Furthermore, neural machine translation has been found to translate between multiple languages, achieving better translation quality by exploiting positive language transfer (Dong et al., 2015; Firat et al., 2016; Zoph and Knight, 2016).

In this paper, we conduct in-depth investigation into the recently proposed multi-way, multilingual neural machine translation (Firat et al., 2016). Specifically, we are interested in its potential for zero-resource machine translation, in which there does not exist any direct parallel examples between a target language pair. Zero-resource translation has been addressed by pivot-based translation in traditional machine translation research (Wu and Wang, 2007; Utiyama and Isahara, 2007; Habash and Hu, 2009), but we explore a way to use the multi-way, multilingual neural model to translate directly from a source to target language.

In doing so, we begin by studying different translation strategies available in the multi-way, multilingual model in Sec. 3–4. The strategies include a usual one-to-one translation as well as variants of many-to-one translation for multi-source translation (Zoph and Knight, 2016). We empirically show that the many-to-one strategies significantly outperform the one-to-one strategy.

We move on to zero-resource translation by first evaluating a vanilla multi-way, multilingual model on a zero-resource language pair, which revealed that the vanilla model cannot do zero-resource translation in Sec. 6.1. Based on the many-to-one strategies we proposed earlier, we design a novel finetun-

[⋆] Work carried out while the author was at IBM Research.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 268–277,
Austin, Texas, November 1-5, 2016.

ing strategy that does not require any direct parallel corpus between a target, zero-resource language pair in Sec. 5.2, which uses the idea of generating a pseudo-parallel corpus (Sennrich et al., 2015a). This strategy makes an additional copy of the attention mechanism and finetunes only this small set of parameters.

Large-scale experiments with Spanish, French and English show that the proposed finetuning strategy allows the multi-way, multilingual neural translation model to perform zero-resource translation as well as a single-pair neural translation model trained with up to 1M true parallel sentences. This result re-confirms the potential of the multi-way, multilingual model for low/zero-resource language translation, which was earlier argued by Firat et al. (2016).

2 Multi-Way, Multilingual Neural Machine Translation

Recently Firat et al. (2016) proposed an extension of attention-based neural machine translation (Bahdanau et al., 2015) that can handle multi-way, multilingual translation with a shared attention mechanism. This model was designed to handle multiple source and target languages. In this section, we briefly overview this multi-way, multilingual model. For more detailed exposition, we refer the reader to (Firat et al., 2016).

2.1 Model Description

The goal of multi-way, multilingual model is to build a neural translation model that can translate a source sentence given in one of N languages into one of M target languages. Thus to handle those N source and M target languages, the model consists of N encoders and M decoders. Unlike these language-specific encoders and decoders, only a single attention mechanism is shared across all $M \times N$ language pairs.

Encoder An encoder for the n-th source language reads a source sentence $X = (x_1, \ldots, x_{T_x})$ as a sequence of linguistic symbols and returns a set of context vectors $C^n = \left\{\mathbf{h}^n_1, \ldots, \mathbf{h}^n_{T_x}\right\}$. The encoder is usually implemented as a bidirectional recurrent network (Schuster and Paliwal, 1997), and each context vector $\mathbf{h}^n_t$ is a concatenation of the forward and reverse recurrent networks' hidden states at time t. Without loss of generality, we assume that the dimensionalities of the context vector for all source languages are all same.

Decoder and Attention Mechanism A decoder for the m-th target language is a conditional recurrent language model (Mikolov et al., 2010). At each time step t', it updates its hidden state by

$$\mathbf{z}^m_{t'} = \varphi^m(\mathbf{z}^m_{t'-1}, \tilde{y}^m_{t'-1}, \mathbf{c}^m_{t'}),$$

based on the previous hidden state $\mathbf{z}^m_{t'-1}$, previous target symbol $\tilde{y}^m_{t'-1}$ and the time-dependent context vector $\mathbf{c}^m_{t'}$. φ^m is a gated recurrent unit (GRU, (Cho et al., 2014)).

The time-dependent context vector is computed by the shared attention mechanism as a weighted sum of the context vectors from the encoder C^n:

$$\mathbf{c}^m_{t'} = \mathbf{U} \sum_{t=1}^{T_x} \alpha^{m,n}_{t,t'} \mathbf{h}^n_t + \mathbf{b}, \tag{1}$$

where

$$\alpha^{m,n}_{t,t'} \propto \exp\left(f_{\text{score}}(\mathbf{W}^n \mathbf{h}^n_t, \mathbf{W}^m \mathbf{z}^m_{t'-1}, \tilde{y}^m_{t'-1})\right). \tag{2}$$

The scoring function f_{score} returns a scalar and is implemented as a feedforward neural network with a single hidden layer. For more variants of the attention mechanism for machine translation, see (Luong et al., 2015b).

The initial hidden state of the decoder is initialized as

$$\mathbf{z}^m_0 = \phi^m_{\text{init}}(\mathbf{W}^n \mathbf{h}^n_t). \tag{3}$$

With the new hidden state $\mathbf{z}^m_{t'}$, the probability distribution over the next symbol is computed by

$$p(y_t = w | \tilde{y}_{<t}, X^n) \propto \exp(g^m_w(\mathbf{z}^m_t, \mathbf{c}^m_t, \mathbf{E}^m_y[\tilde{y}_{t-1}]), \tag{4}$$

where g^m_w is a decoder specific parametric function that returns the unnormalized probability for the next target symbol being w.

2.2 Learning

Training this multi-way, multilingual model does not require multi-way parallel corpora but only a

set of bilingual corpora. For each bilingual pair, the conditional log-probability of a ground-truth translation given a source sentence is maximize by adjusting the relevant parameters following the gradient of the log-probability.

3 Translation Strategies

3.1 One-to-One Translation

In the original paper by Firat et al. (2016), only one translation strategy was evaluated, that is, *one-to-one translation*. This one-to-one strategy works on a source sentence given in one language by taking the encoder of that source language, the decoder of a target language and the shared attention mechanism. These three components are glued together as if they form a single-pair neural translation model and translates the source sentence into a target language.

We however notice that this is not the only translation strategy available with the multi-way, multilingual model. As we end up with multiple encoders, multiple decoders and a shared attention mechanism, this model naturally enables us to exploit a source sentence given in multiple languages, leading to a *many- to-one translation* strategy which was proposed recently by Zoph and Knight (2016) in the context of neural machine translation.

Unlike (Zoph and Knight, 2016), the multi-way, multilingual model is not trained with multi-way parallel corpora. This however does not necessarily imply that the model cannot be used in this way. In the remainder of this section, we propose two alternatives for doing multi-source translation with the multi-way, multilingual model, which eventually pave the way towards zero-resource translation.

3.2 Many-to-One Translation

In this section, we consider a case where a source sentence is given in two languages, X_1 and X_2. However, any of the approaches described below applies to more than two source languages trivially.

In this multi-way, multilingual model, multi-source translation can be thought of as averaging two separate translation paths. For instance, in the case of Es+Fr to En, we want to combine Es→En and Fr→En so as to get a better English translation. We notice that there are two points in the multi-way, multilingual model where this averaging may happen.

Early Average The first candidate is to averaging two translation paths when computing the time-dependent context vector (see Eq. (1).) At each time t in the decoder, we compute a time-dependent context vector for each source language, $\mathbf{c}_t^1$ and $\mathbf{c}_t^2$ respectively for the two source languages. In this early averaging strategy, we simply take the average of these two context vectors:

$$\mathbf{c}_t = \frac{\mathbf{c}_t^1 + \mathbf{c}_t^2}{2}. \tag{5}$$

Similarly, we initialize the decoder's hidden state to be the average of the initializers of the two encoders:

$$\mathbf{z}_0 = \frac{1}{2}\left(\phi_{\text{init}}(\phi_{\text{init}}^1(\mathbf{h}_{T_{x_1}}^1)) + \phi_{\text{init}}(\phi_{\text{init}}^2(\mathbf{h}_{T_{x_1}}^2))\right), \tag{6}$$

where ϕ_{init} is the decoder's initializer (see Eq. (3).)

Late Average Alternatively, we can average those two translation paths (e.g., Es→En and Fr→En) at the output level. At each time t, each translation path computes the distribution over the target vocabulary, i.e., $p(y_t = w|y_{<t}, X_1)$ and $p(y_t = w|y_{<t}, X_2)$. We then average them to get the multi-source output distribution:

$$p(y_t = w|y_{<t}, X_1, X_2) = \frac{1}{2}(p(y_t = w|y_{<t}, X_1) + p(y_t = w|y_{<t})). \tag{7}$$

An advantage of this late averaging strategy over the early averaging one is that this can work even when those two translation paths were not from a single multilingual model. They can be two separately trained single-pair models. In fact, if X_1 and X_2 are same and the two translation paths are simply two different models trained on the same language pair–direction, this is equivalent to constructing an ensemble, which was found to greatly improve translation quality (Sutskever et al., 2014; Jean et al., 2015)

Early+Late Average The two strategies above can be further combined by late-averaging the output distributions from the early averaged model and the late averaged one. We empirically evaluate this early+late average strategy as well.

4 Experiments: Translation Strategies and Multi-Source Translation

Before continuing on with zero-resource machine translation, we first evaluate the translation strategies described in the previous section on multi-source translation, as these translation strategies form a basic foundation on which we extend the multi-way, multilingual model for zero-resource machine translation.

4.1 Settings

When evaluating the multi-source translation strategies, we use English, Spanish and French, and focus on a scenario where only En-Es and En-Fr parallel corpora are available.

4.1.1 Corpora

En-Es We combine the following corpora to form 34.71m parallel Es-En sentence pairs: UN (8.8m), Europarl-v7 (1.8m), news-commentary-v7 (150k), LDC2011T07-T12 (2.9m) and internal technical-domain data (21.7m).

En-Fr We combine the following corpora to form 65.77m parallel En-Fr sentence pairs: UN (9.7m), Europarl-v7 (1.9m), news-commentary-v7 (1.2m), LDC2011T07-T10 (1.6m), ReutersUN (4.5m), internal technical-domain data (23.5m) and Gigaword R2 (20.66m).

Evaluation Sets We use newstest-2012 and newstest-2013 from WMT as development and test sets, respectively.

Monolingual Corpora We do not use any additional monolingual corpus.

Preprocessing All the sentences are tokenized using the tokenizer script from Moses (Koehn et al., 2007). We then replace special tokens, such as numbers, dates and URL's with predefined markers, which will be replaced back with the original tokens *after* decoding. After using byte pair encoding (BPE, (Sennrich et al., 2015b)) to get subword symbols, we end up with 37k, 43k and 45k unique tokens for English, Spanish and French, respectively. For training, we only use sentence pairs in which both sentences are only up to 50 symbols long.

See Table 1 for the detailed statistics.

# Sents	Train	Dev†	Test‡
En-Es	34.71m	3003	3000
En-Fr	65.77m	3003	3000
En-Es-Fr	11.32m	3003	3000

Table 1: Data statistics. †: newstest-2012. ‡: newstest-2013

4.2 Models and Training

We start from the code made publicly available as a part of (Firat et al., 2016)[1]. We made two changes to the original code. First, we replaced the decoder with the conditional gated recurrent network with the attention mechanism as outlines in (Firat and Cho, 2016). Second, we feed a binary indicator vector of which encoder(s) the source sentence was processed by to the output layer of each decoder (g_w^m in Eq. (4)). Each dimension of the indicator vector corresponds to one source language, and in the case of multi-source translation, there may be more than one dimensions set to 1.

We train the following models: four single-pair models (Es↔En and Fr↔En) and one multi-way, multilingual model (Es,Fr,En↔Es,Fr,En). As proposed by Firat et al. (2016), we share one attention mechanism for the latter case.

Training We closely follow the setup from (Firat et al., 2016). Each symbol is represented as a 620-dimensional vector. Any recurrent layer, be it in the encoder or decoder, consists of 1000 gated recurrent units (GRU, (Cho et al., 2014)), and the attention mechanism has a hidden layer of 1200 $\tanh$ units (f_{score} in Eq. (2)). We use Adam (Kingma and Ba, 2015) to train a model, and the gradient at each update is computed using a minibatch of at most 80 sentence pairs. The gradient is clipped to have the norm of at most 1 (Pascanu et al., 2012). We early-stop any training using the T-B score on a development set[2].

4.3 One-to-One Translation

We first confirm that the multi-way, multilingual translation model indeed works as well as single-pair models on the translation paths that were considered during training, which was the major claim

[1] https://github.com/nyu-dl/dl4mt-multi

[2] T-B score is defined as $\frac{\text{TER}-\text{BLEU}}{2}$ which we found to be more stable than either TER or BLEU alone for the purpose of early-stopping (Zhao and Chen, 2009).

	Src	Trgt	Multi Dev	Multi Test	Single Dev	Single Test
(a)	Es	En	30.73	28.32	29.74	27.48
(b)	Fr	En	26.93	27.93	26.00	27.21
(c)	En	Es	30.63	28.41	31.31	28.90
(d)	En	Fr	22.68	23.41	22.80	24.05

Table 2: One-to-one translation qualities using the multi-way, multilingual model and four separate single-pair models.

		Multi Dev	Multi Test	Single Dev	Single Test
(a)	Early	31.89	31.35	–	–
(b)	Late	32.04	31.57	32.00	31.46
(c)	E+L	32.61	31.88	–	–

Table 3: Many-to-one quality (Es+Fr→En) using three translation strategies. Compared to Table 2 (a–b) we observe a significant improvement (up to 3+ BLEU), although the model was never trained in these many-to-one settings. The second column shows the quality by the ensemble of two separate single-pair models.

in (Firat et al., 2016). In Table 2, we present the results on four language pair-directions (Es↔En and Fr↔En).

It is clear that the multi-way, multilingual model indeed performs comparably on all the four cases with less parameters (due to the shared attention mechanism.) As observed earlier in (Firat et al., 2016), we also see that the multilingual model performs better when a target language is English.

4.4 Many-to-One Translation

We consider translating from a pair of source sentences in Spanish (Es) and French (Fr) to English (En). It is important to note that the multilingual model was *not* trained with any multi-way parallel corpus. Despite this, we observe that the early averaging strategy improves the translation quality (measured in BLEU) by 3 points in the case of the test set (compare Table 2 (a–b) and Table 3 (a).) We conjecture that this happens as training the multilingual model has implicitly encouraged the model to find a *common context vector* space across multiple source languages.

The late averaging strategy however outperforms the early averaging in both cases of multilingual model and a pair of single-pair models (see Table 3 (b)) albeit marginally. The best quality was observed when the early and late averaging strategies were combined at the output level, achieving up to +3.5 BLEU (compare Table 2 (a) and Table 3 (c).)

We emphasize again that there was *no* multi-way parallel corpus consisting of Spanish, French and English during training[3]. The result presented in this section shows that the multi-way, multilingual model can exploit multiple sources effectively without requiring any multi-way parallel corpus, and we will rely on this property together with the proposed many-to-one translation strategies in the later sections where we propose and investigate zero-resource translation.

5 Zero-Resource Translation Strategies

The network architecture of multi-way, multilingual model suggests the potential for translating between two languages *without* any direct parallel corpus available. In the setting considered in this paper (see Sec. 4.1,) these translation paths correspond to Es↔Fr, as only parallel corpora used for training were Es↔En and Fr↔En.

The most naive approach for translating along a zero-resource path is to simply treat it as any other path that was included as a part of training. This corresponds to the one-to-one strategy from Sec. 3.1. In our experiments, it however turned out that this naive approach does not work at all, as can be seen in Table 4 (a).

In this section, we investigate this potential of zero-resource translation with the multi-way, multilingual model in depth. More specifically, we propose a number of approaches that enable zero-resource translation without requiring any additional bilingual or multi-way corpora.

5.1 Pivot-based Translation

The first set of approaches exploits the fact that the target zero-resource translation path can be decomposed into a sequence of high-resource translation paths (Wu and Wang, 2007; Utiyama and Isahara, 2007; Habash and Hu, 2009). For instance, in our

[3]We do not assume the availability of annotation on multi-way parallel sentence pairs. It is likely that there will be some sentence (or a set of very close variants of a single sentence) translated into multiple languages (eg. Europarl). One may decide to introduce a mechanism for exploiting these (Zoph and Knight, 2016), or as we present here, it may not be necessary at all to do so.

case, Es→Fr can be decomposed into a sequence of Es→En and En→Fr. In other words, we translate a source sentence (Es) into a pivot language (En) and then translate the English translation into a target language (Fr), all within the same multi-way, multilingual model trained by using bilingual corpora.

One-to-One Translation The most basic approach here is to perform each translation path in the decomposed sequence independently from each other. This one-to-one approach introduces only a minimal computational complexity (the multiplicative factor of two.) We can further improve this one-to-one pivot-based translation by maintaining a set of k-best translations from the first stage (Es→En), but this increase the overall computational complexity by the factor of k, making it impractical in practice. We therefore focus only on the former approach of keeping the best pivot translation in this paper.

Many-to-One Translation With the multi-way, multilingual model considered in this paper, we can extend the naive one-to-one pivot-based strategy by replacing the second stage (En→Fr) to be many-to-one translation from Sec. 4.4 using both the original source language and the pivot language as a pair of source languages. We first translate the source sentence (Es) into English, and use both the original source sentence and the English translation (Es+En) to translate into the final target language (Fr).

Both approaches described and proposed above do not require any additional action on an already-trained multilingual model. They are simply different translation strategies specifically aimed at zero-resource translation.

5.2 Finetuning with Pseudo Parallel Corpus

The failure of the naive zero-resource translation earlier (see Table 4 (a)) suggests that the context vectors returned by the encoder are not compatible with the decoder, when the combination was not included during training. The good translation qualities of the translation paths included in training however imply that the representations learned by the encoders and decoders are good. Based on these two observations, we conjecture that all that is needed for a zero-resource translation path is a simple adjustment that makes the context vectors from the encoder to be compatible with the target decoder. Thus, we propose to adjust this zero-resource translation path however without any additional parallel corpus.

First, we generate a small set of *pseudo bilingual pairs* of sentences for the zero-resource language pair (Es→Fr) in interest. We randomly select N sentences pairs from a parallel corpus between the target language (Fr) and a pivot language (En) and translate the pivot side (En) into the source language (Es). Then, the pivot side is discarded, and we construct a *pseudo* parallel corpus consisting of sentence pairs of the source and target languages (Es-Fr).

We make a copy of the existing attention mechanism, to which we refer as *target-specific attention mechanism*. We then finetune only this target-specific attention mechanism while keeping all the other parameters of the encoder and decoder intact, using the generated pseudo parallel corpus. We do not update any other parameters in the encoder and decoder, because they are already well-trained (evidenced by high translation qualities in Table 2) and we want to avoid disrupting the well-captured structures underlying each language.

Once the model has been finetuned with the pseudo parallel corpus, we can use any of the translation strategies described earlier in Sec. 3 for the finetuned zero-resource translation path. We expect a similar gain by using many-to-one translation, which we empirically confirm in the next section.

6 Experiments: Zero-Resource Translation

6.1 Without Finetuning

6.1.1 Settings

We use the same multi-way, multilingual model trained earlier in Sec. 4.2 to evaluate the zero-resource translation strategies. We emphasize here that this model was trained only using Es-En and Fr-En *bilingual* parallel corpora without any Es-Fr parallel corpus.

We evaluate the proposed approaches to zero-resource translation with the same multi-way, multilingual model from Sec. 4.1. We specifically select the path from Spanish to French (Es→Fr) as a target zero-resource translation path.

	Pivot	Many-to-1	Dev	Test
(a)			< 1	< 1
(b)	$\surd$		20.64	20.4
(c)	$\surd$	Early	9.24	10.42
(d)	$\surd$	Late	18.22	19.14
(e)	$\surd$	E+L	13.29	14.56

Table 4: Zero-resource translation from Spanish (Es) to French (Fr) *without* finetuning, using multi-way, multilingual model. When pivot is $\surd$, English is used as a pivot language.

6.1.2 Result and Analysis

As mentioned earlier, we observed that the multi-way, multilingual model *cannot* directly translate between two languages when the translation path between those two languages was not included in training (Table 4 (a).) On the other hand, the model was able to translate decently with the pivot-based one-to-one translation strategy, as can be seen in Table 4 (b). Unsurprisingly, all the many-to-one strategies resulted in worse translation quality, which is due to the inclusion of the useless translation path (direct path between the zero-resource pair, Es-Fr). Another interesting trend we observe is the Early+Late averaging (Table 4 (e)) seems to perform worse than Late averaging (Table 4 (d)) alone, opposite of the results in Table 3 (b-c). We conjecture that, by simply averaging two model outputs (as in E+L), when one of them is drastically worse than the other, has the effect of pulling down the performance of final results. But early averaging can still recover from this deficiency, upto some extent, since the decoder output probability function g_w^m (Eq. (4).) is a smooth function not only using the averaged context vectors (Eq. (5).).

These results clearly indicate that the multi-way, multilingual model trained with only bilingual parallel corpora is not capable of direct zero-resource translation *as it is*.

6.2 Finetuning with a Pseudo Parallel Corpus

6.2.1 Settings

The proposed finetuning strategy raises a number of questions. First, it is unclear how many pseudo sentence pairs are needed to achieve a decent translation quality. Because the purpose of this finetuning stage is simply to adjust the shared attention mechanism so that it can properly bridge from the source-side encoder to the target-side decoder, we expect it to work with only a small amount of pseudo pairs. We validate this by creating pseudo corpora of different sizes–1k, 10k, 100k and 1m.

Second, we want to know how detrimental it is to use the generated pseudo sentence pairs compared to using true sentence pairs between the target language pair. In order to answer this question, we compiled a true multi-way parallel corpus by combining the subsets of UN (7.8m), Europarl-v7 (1.8m), OpenSubtitles-2013 (1m), news-commentary-v7 (174k), LDC2011T07 (335k) and news-crawl (310k), and use it to finetune the model[4]. This allows us to evaluate the effect of the pseudo and true parallel corpora on finetuning for zero-resource translation.

Lastly, we train single-pair models translating directly from Spanish to French by using the true parallel corpora. These models work as a baseline against which we compare the multi-way, multilingual models.

Training Unlike the usual training procedure described in Sec. 4.2, we compute the gradient for each update using 60 sentence pairs only, when finetuning the model with the multi-way parallel corpus (either pseudo or true.)

6.2.2 Result and Analysis

Table 5 summarizes all the result. The most important observation is that the proposed finetuning strategy with *pseudo*-parallel sentence pairs outperforms the pivot-based approach (using the early averaging strategy from Sec. 4.4) even when we used only 10k such pairs (compare (b) and (d).) As we increase the size of the pseudo-parallel corpus, we observe a clear improvement. Furthermore, these models perform comparably to or better than the single-pair model trained with 1M *true* parallel sentence pairs, *although they never saw a single true bilingual sentence pair* of Spanish and French (compare (a) and (d).)

Another interesting finding is that it is only beneficial to use true parallel pairs for finetuning the multi-way, mulitilingual models when there are enough of them (1m or more). When there are only a small number of true parallel sentence pairs, we

[4] See the last row of Table 1.

	Pivot	Many-to-1		Pseudo Parallel Corpus 1k	10k	100k	1m	True Parallel Corpus 1k	10k	100k	1m
(a)	Single-Pair Models		Dev	–	–	–	–	–	–	11.25	21.32
			Test	–	–	–	–	–	–	10.43	20.35
(b)	√	No Finetuning		Dev: 20.64, Test: 20.4				–			
(c)			Dev	0.28	10.16	15.61	17.59	0.1	8.45	16.2	20.59
			Test	0.47	10.14	15.41	17.61	0.12	8.18	15.8	19.97
(d)	√	Early	Dev	19.42	21.08	21.7	21.81	8.89	16.89	20.77	22.08
			Test	19.43	20.72	21.23	21.46	9.77	16.61	20.40	21.7
(e)	√	Early+ Late	Dev	20.89	20.93	21.35	21.33	14.86	18.28	20.31	21.33
			Test	20.5	20.71	21.06	21.19	15.42	17.95	20.16	20.9

Table 5: Zero-resource translation from Spanish (Es) to French (Fr) *with* finetuning. When pivot is √, English is used as a pivot language. Row (b) is from Table 4 (b).

even found using pseudo pairs to be more beneficial than true ones. This effective as more apparent, when the direct one-to-one translation of the zero-resource pair was considered (see (c) in Table 5.) This applies that the misalignment between the encoder and decoder can be largely fixed by using pseudo-parallel pairs only, and we conjecture that it is easier to learn from pseudo-parallel pairs as they better reflect the inductive bias of the trained model and as the pseudo- parallel corpus is expected to be more noisy, this may be an implicit regularization effect. When there is a large amount of true parallel sentence pairs available, however, our results indicate that it is better to exploit them.

Unlike we observed with the multi-source translation in Sec. 3.2, we were not able to see any improvement by further averaging the early-averaged and late-average decoding schemes (compare (d) and (e).) This may be explained by the fact that the context vectors computed when creating a pseudo source (e.g., En from Es when Es→Fr) already contains all the information about the pseudo source. It is simply enough to take those context vectors into account via the early averaging scheme.

These results clearly indicate and verify the potential of the multi-way, multilingual neural translation model in performing zero-resource machine translation. More specifically, it has been shown that the translation quality can be improved even without any direct parallel corpus available, and if there is a small amount of direct parallel pairs available, the quality may improve even further.

7 Conclusion: Implications and Limitations

Implications There are two main results in this paper. First, we showed that the multi-way, multilingual neural translation model by Firat et al. (2016) is able to exploit common, underlying structures across many languages in order to better translate when a source sentence is given in multiple languages. This confirms the usefulness of positive language transfer, which has been believed to be an important factor in human language learning (Odlin, 1989; Ringbom, 2007), in machine translation. Furthermore, our result significantly expands the applicability of multi-source translation (Zoph and Knight, 2016), as it does not assume the availability of multi-way parallel corpora for training and relies only on *bilingual* parallel corpora.

Second, the experiments on zero-resource translation revealed that it is not necessary to have a direct parallel corpus, or deep linguistic knowledge, between two languages in order to build a machine translation system. Importantly we observed that the proposed approach of zero-resource translation is better both in terms of translation quality and data efficiency than a more traditional pivot-based translation (Wu and Wang, 2007; Utiyama and Isahara, 2007). Considering that this is the first attempt at such zero-resource, or extremely low-resource, translation using neural machine translation, we expect a large progress in near future.

Limitations Despite the promising empirical results presented in this paper, there are a number of shortcomings that needs to addressed in follow-up research. First, our experiments have been done only with three European languages–Spanish, French and English. More investigation with a diverse set of languages needs to be done in order to make a more solid conclusion, such as was done in (Firat et al., 2016; Chung et al., 2016). Furthermore, the effect of varying sizes of available parallel corpora on the performance of zero-resource translation must be studied more in the future.

Second, although the proposed many-to-one translation is indeed generally applicable to any number of source languages, we have only tested a source sentence in two languages. We expect even higher improvement with more languages, but it must be tested thoroughly in the future.

Lastly, the proposed finetuning strategy requires the model to have an additional set of parameters relevant to the attention mechanism for a target, zero-resource pair. This implies that the number of parameters may grow linearly with respect to the number of target language pairs. We expect future research to address this issue by, for instance, mixing in the parallel corpora of high-resource language pairs during finetuning as well.

Acknowledgments

OF thanks Iulian Vlad Serban and Georgiana Dinu for insightful discussions. KC thanks the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016. Does multimodality help human and machine for translation and image captioning? *arXiv preprint arXiv:1605.09186*.

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

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *ACL*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. ACL.

Orhan Firat and Kyunghyun Cho. 2016. DL4MT-Tutorial: Conditional gated recurrent unit with attention mechanism.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Mikel L Forcada and Ramón P Ñeco. 1997. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology*, pages 453–462. Springer.

Nizar Habash and Jun Hu. 2009. Improving arabic-chinese statistical machine translation using english as pivot language. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 173–181. Association for Computational Linguistics.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal, September. Association for Computational Linguistics.

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

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv:1511.04586*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv:1604.00788*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH*, 2:3.

Terence Odlin. 1989. *Language Transfer*. Cambridge University Press. Cambridge Books Online.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Håkan Ringbom. 2007. *Cross-linguistic similarity in foreign language learning*, volume 21.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *HLT-NAACL*, pages 484–491.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Bing Zhao and Shengyuan Chen. 2009. A simplex armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *HLT-NAACL*, pages 21–24.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL*.

Memory-enhanced Decoder for Neural Machine Translation

Mingxuan Wang[1] **Zhengdong Lu**[2] **Hang Li**[2] **Qun Liu**[3,1]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
{wangmingxuan,liuqun}@ict.ac.cn
[2]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong,HangLi.HL}@huawei.com
[3]ADAPT Centre, School of Computing, Dublin City University

Abstract

We propose to enhance the RNN decoder in a neural machine translator (NMT) with external memory, as a natural but powerful extension to the state in the decoding RNN. This memory-enhanced RNN decoder is called MEMDEC. At each time during decoding, MEMDEC will read from this memory and write to this memory once, both with content-based addressing. Unlike the unbounded memory in previous work(Bahdanau et al., 2014) to store the representation of source sentence, the memory in MEMDEC is a matrix with pre-determined size designed to better capture the information important for the decoding process at each time step. Our empirical study on Chinese-English translation shows that it can improve by 4.8 BLEU upon Groundhog and 5.3 BLEU upon on Moses, yielding the best performance achieved with the same training set.

1 Introduction

The introduction of external memory has greatly expanded the representational capability of neural network-based model on modeling sequences(Graves et al., 2014), by providing flexible ways of storing and accessing information. More specifically, in neural machine translation, one great improvement came from using an array of vectors to represent the source in a sentence-level memory and dynamically accessing relevant segments of them (alignment) (Bahdanau et al., 2014) through content-based addressing (Graves et al., 2014). The success of RNNsearch demonstrated the advantage of saving the entire sentence of arbitrary length in an unbounded memory for operations of next stage (e.g., decoding).

In this paper, we show that an external memory can be used to facilitate the decoding/generation process thorough a memory-enhanced RNN decoder, called MEMDEC. The memory in MEMDEC is a direct extension to the state in the decoding, therefore functionally closer to the memory cell in LSTM(Hochreiter and Schmidhuber, 1997). It takes the form of a matrix with pre-determined size, each column ("a memory cell") can be accessed by the decoding RNN with content-based addressing for both reading and writing during the decoding process. This memory is designed to provide a more flexible way to select, represent and synthesize the information of source sentence and previously generated words of target relevant to the decoding. This is in contrast to the set of hidden states of the entire source sentence (which can viewed as another form of memory) in (Bahdanau et al., 2014) for attentive read, but can be combined with it to greatly improve the performance of neural machine translator. We apply our model on English-Chinese translation tasks, achieving performance superior to any published results, SMT or NMT, on the same training data (Xie et al., 2011; Meng et al., 2015; Tu et al., 2016; Hu et al., 2015)

Our contributions are mainly two-folds

- we propose a memory-enhanced decoder for

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 278–286,
Austin, Texas, November 1-5, 2016.

neural machine translator which naturally extends the RNN with vector state.

- our empirical study on Chinese-English translation tasks show the efficacy of the proposed model.

Roadmap In the remainder of this paper, we will first give a brief introduction to attention-based neural machine translation in Section 2, presented from the view of encoder-decoder, which treats the hidden states of source as an unbounded memory and the attention model as a content-based reading. In Section 3, we will elaborate on the memory-enhanced decoder MEMDEC. In Section 4, we will apply NMT with MEMDEC to a Chinese-English task. Then in Section 5 and 6, we will give related work and conclude the paper.

2 Neural machine translation with attention

Our work is built on attention-based NMT(Bahdanau et al., 2014), which represents the source sentence as a sequence of vectors after being processed by RNN or bi-directional RNNs, and then conducts dynamic alignment and generation of the target sentence with another RNN simultaneously.

Attention-based NMT, with RNNsearch as its most popular representative, generalizes the conventional notion of encoder-decoder in using a unbounded memory for the intermediate representation of source sentence and content-based addressing read in decoding, as illustrated in Figure 1. More specifically, at time step t, RNNsearch first get context vector $\mathbf{c}_t$ after reading from the source representation $\mathbf{M}^{\mathrm{s}}$, which is then used to update the state, and generate the word y_t (along with the current hidden state $\mathbf{s}_t$, and the previously generated word y_{i-1}).

Formally, given an input sequence $\mathbf{x} = [x_1, x_2, \ldots, x_{T_x}]$ and the previously generated sequence $\mathbf{y}_{<t} = [y_1, y_2, \ldots, y_{t-1}]$, the probability of next word y_t is

$$p(y_t|\mathbf{y}_{<t}; \mathbf{x}) = f(\mathbf{c}_t, y_{t-1}, \mathbf{s}_t), \qquad (1)$$

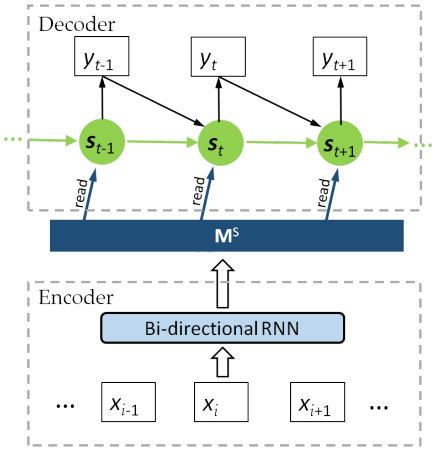


Figure 1: RNNsearch in the encoder-decoder view.

where $\mathbf{s}_t$ is state of decoder RNN at time step t calculated as

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t). \qquad (2)$$

where $g(\cdot)$ can be an be any activation function, here we adopt a more sophisticated dynamic operator as in Gated Recurrent Unit (GRU, (Cho et al., 2014)). In the remainder of the paper, we will also use GRU to stand for the operator. The reading $\mathbf{c}_t$ is calculated as

$$\mathbf{c}_t = \sum_{j=1}^{j=T_x} \alpha_{t,j}\mathbf{h}_j, \qquad (3)$$

where $\mathbf{h}_j$ is the j^{th} cell in memory $\mathbf{M}^{\mathrm{s}}$. More formally, $\mathbf{h}_j = [\overleftarrow{\mathbf{h}_j}^\top, \overrightarrow{\mathbf{h}_j}^\top]^\top$ is the annotations of x_j and contains information about the whole input sequence with a strong focus on the parts surrounding x_j, which is computed by a bidirectional RNN. The weight $\alpha_{t,j}$ is computed by

$$\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^{k=T_x} \exp(e_{t,k})}.$$

where $e_{i,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a\mathbf{s}_{t-1} + \mathbf{U}_a\mathbf{h}_j)$ scores how well $\mathbf{s}_{t-1}$ and the memory cell $\mathbf{h}_j$ match. This is called automatic alignment (Bahdanau et al., 2014) or attention model (Luong et al., 2015), but it is essentially reading with content-based addressing defined in (Graves et al., 2014). With this addressing strategy the decoder can attend to the source representation that is most relevant to the stage of decoding.

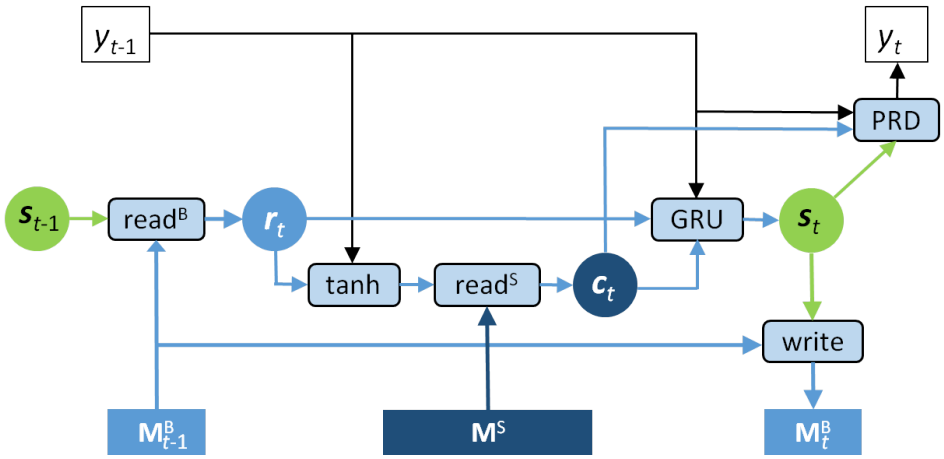


Figure 2: Diagram of the proposed decoder MEMDEC with details.

2.1 Improved Attention Model

The alignment model $\alpha_{t,j}$ scores how well the output at position t matches the inputs around position j based on $\mathbf{s}_{t-1}$ and h_j. It is intuitively beneficial to exploit the information of y_{t-1} when reading from $\mathbf{M}^S$, which is missing from the implementation of attention-based NMT in (Bahdanau et al., 2014). In this work, we build a more effective alignment path by feeding both previous hidden state $\mathbf{s}_{t-1}$ and the context word y_{t-1} to the attention model, inspired by the recent implementation of attention-based NMT[1]. Formally, the calculation of $e_{t,j}$ becomes

$$e_{t,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \tilde{\mathbf{s}}_{t-1} + \mathbf{U}_a \mathbf{h}_j),$$

where

- $\tilde{\mathbf{s}}_{t-1} = \mathcal{H}(\mathbf{s}_{t-1}, \mathbf{e}_{y_{t-1}})$ is an intermediate state tailored for reading from $\mathbf{M}^S$ with the information of y_{t-1} (its word embedding being $\mathbf{e}_{y_{t-1}}$) added;

- $\mathcal{H}$ is a nonlinear function, which can be as simple as tanh or as complex as GRU. In our preliminary experiments, we found GRU works slightly better than tanh function, but we chose the latter for simplicity.

[1] github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

3 Decoder with External Memory

In this section we will elaborate on the proposed memory-enhanced decoder MEMDEC. In addition to the source memory $\mathbf{M}^S$, MEMDEC is equipped with a buffer memory $\mathbf{M}^B$ as an extension to the conventional state vector. Figure 3 contrasts MEMDEC with the decoder in RNNsearch (Figure 1) on a high level.

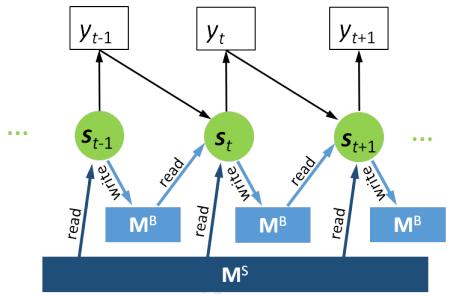


Figure 3: High level digram of MEMDEC.

In the remainder of the paper, we will refer to the conventional state as vector-state (denoted $\mathbf{s}_t$) and its memory extension as memory-state (denoted as $\mathbf{M}^B_t$). Both states are updated at each time step in a interweaving fashion, while the output symbol y_t is predicted based solely on vector-state $\mathbf{s}_t$ (along with $\mathbf{c}_t$ and y_{t-1}). The diagram of this memory-enhanced decoder is given in Figure 2.

Vector-State Update At time t, the vector-state $\mathbf{s}_t$ is first used to read $\mathbf{M}^{\text{B}}$

$$\mathbf{r}_{t-1} = \text{read}^{\text{B}}(\mathbf{s}_{t-1}, \mathbf{M}^{\text{B}}_{t-1}) \quad (4)$$

which then meets the previous prediction y_{t-1} to form an "intermediate" state-vector

$$\widetilde{\mathbf{s}}_t = \tanh(\mathbf{W}^r \mathbf{r}_{t-1} + \mathbf{W}^y \mathbf{e}_{y_{t-1}}). \quad (5)$$

where $\mathbf{e}_{y_{t-1}}$ is the word-embedding associated with the previous prediction y_{t-1}. This pre-state $\widetilde{\mathbf{s}}_t$ is used to read the source memory $\mathbf{M}^{\text{S}}$

$$\mathbf{c}_t = \text{read}^{\text{S}}(\widetilde{\mathbf{s}}_t, \mathbf{M}^{\text{S}}). \quad (6)$$

Both readings in Eq. (4) & (6) follow content-based addressing(Graves et al., 2014) (details later in Section 3.1). After that, $\mathbf{r}_{t-1}$ is combined with output symbol y_{t-1} and $\mathbf{c}_t$ to update the new vector-state

$$\mathbf{s}_t = \text{GRU}(\mathbf{r}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t) \quad (7)$$

The update of vector-state is illustrated in Figure 4.

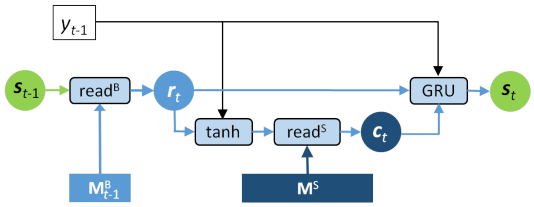


Figure 4: Vector-state update at time t.

Memory-State Update As illustrated in Figure 5, the update for memory-state is simple after the update of vector-state: with the vector-state $\mathbf{s}_{t+1}$ the updated memory-state will be

$$\mathbf{M}^{\text{B}}_t = \text{write}(\mathbf{s}_t, \mathbf{M}^{\text{B}}_{t-1}) \quad (8)$$

The writing to the memory-state is also content-based, with same forgetting mechanism suggested in (Graves et al., 2014), which we will elaborate with more details later in this section.

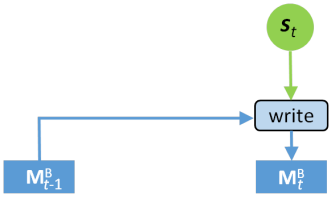


Figure 5: Memory-state update at time t.

Prediction As illustrated in Figure 6, the prediction model is same as in (Bahdanau et al., 2014), where the score for word y is given by

$$\text{score}(y) = \text{DNN}([\mathbf{s}_t, \mathbf{c}_t, \mathbf{e}_{y_{t-1}}])^\top \omega_y \quad (9)$$

where ω_y is the parameters associated with the word y. The probability of generating word y at time t is then given by a softmax over the scores

$$p(y|\mathbf{s}_t, \mathbf{c}_t, y_{t-1}) = \frac{\exp(\text{score}(y))}{\sum_{y'} \exp(\text{score}(y'))}.$$

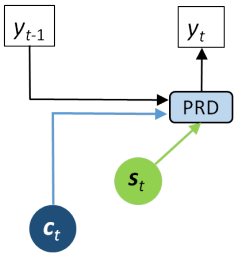


Figure 6: Prediction at time t.

3.1 Reading Memory-State

Formally $\mathbf{M}^{\text{B}}_{t'} \in \mathbb{R}^{n \times m}$ is the memory-state at time t' after the memory-state update, where n is the number of memory cells and m is the dimension of vector in each cell. Before the vector-state update at time t, the output of reading $\mathbf{r}_t$ is given by

$$\mathbf{r}_t = \sum_{j=1}^{j=n} \mathbf{w}^{\text{R}}_t(j) \mathbf{M}^{\text{B}}_{t-1}(j)$$

where $\mathbf{w}^{\text{R}}_t \in \mathbb{R}^n$ specifies the *normalized* weights assigned to the cells in $\mathbf{M}^{\text{B}}_t$. Similar with the reading from $\mathbf{M}^{\text{S}}$ (a.k.a. attention model), we use content-based addressing in determining $\mathbf{w}^{\text{R}}_t$.

More specifically, $\mathbf{w}_t^{\text{R}}$ is also updated from the one from previous time $\mathbf{w}_{t-1}^{\text{R}}$ as

$$\mathbf{w}_t^{\text{R}} = g_t^{\text{R}} \mathbf{w}_{t-1}^{\text{R}} + (1 - g_t^{\text{R}}) \widetilde{\mathbf{w}}_t^{\text{R}}, \qquad (10)$$

where

- $g_t^{\text{R}} = \sigma(\mathbf{w}_g^{\text{R}} \mathbf{s}_t)$ is the gate function, with parameters $\mathbf{w}_g^{\text{R}} \in \mathbb{R}^m$;
- $\widetilde{\mathbf{w}}_t$ gives the contribution based on the current vector-state $\mathbf{s}_t$

$$\widetilde{\mathbf{w}}_t^{\text{R}} = \text{softmax}(\mathbf{a}_t^{\text{R}}) \qquad (11)$$

$$\mathbf{a}_t^{\text{R}}(i) = \mathbf{v}^\top (\mathbf{W}_a^{\text{R}} \mathbf{M}_{t-1}^{\text{B}}(i) + \mathbf{U}_a^{\text{R}} \mathbf{s}_{t-1}), \qquad (12)$$

with parameters $\mathbf{W}_a^{\text{R}}, \mathbf{U}_a^{\text{R}} \in \mathbb{R}^{m \times m}$ and $\mathbf{v} \in \mathbb{R}^m$.

3.2 Writing to Memory-State

There are two types of operation on writing to memory-state: ERASE and ADD. Erasion is similar to the forget gate in LSTM or GRU, which determines the content to be remove from memory cells. More specifically, the vector $\mu_t^{\text{ERS}} \in \mathbb{R}^m$ specifies the values to be removed on each dimension in memory cells, which is than assigned to each cell through normalized weights $\mathbf{w}_t^{\text{W}}$. Formally, the memory-state after ERASE is given by

$$\widetilde{\mathbf{M}}_t^{\text{B}}(i) = \mathbf{M}_{t-1}^{\text{B}}(i)(1 - \mathbf{w}_t^{\text{W}}(i) \cdot \mu_t^{\text{ERS}}) \qquad (13)$$

$$i = 1, \cdots, n$$

where

- $\mu_t^{\text{ERS}} = \sigma(\mathbf{W}^{\text{ERS}} \mathbf{s}_t)$ is parametrized with $\mathbf{W}^{\text{ERS}} \in \mathbb{R}^{m \times m}$;
- $\mathbf{w}_t^{\text{W}}(i)$ specifies the weight associated with the i^{th} cell in the same parametric form as in Eq. (10)-(12) with generally different parameters.

ADD operation is similar with the update gate in LSTM or GRU, deciding how much current information should be written to the memory.

$$\mathbf{M}_t^{\text{B}}(i) = \widetilde{\mathbf{M}}_t^{\text{B}}(i) + \mathbf{w}_t^{\text{W}}(i) \mu_t^{\text{ADD}}$$

$$\mu_t^{\text{ADD}} = \sigma(\mathbf{W}^{\text{ADD}} \mathbf{s}_t)$$

where $\mu_t^{\text{ADD}} \in \mathbb{R}^m$ and $\mathbf{W}^{\text{ADD}} \in \mathbb{R}^{m \times m}$.

In our experiments, we have a peculiar but interesting observation: it is often beneficial to use the same weights for both reading (i.e., $\mathbf{w}_t^{\text{R}}$ in Section 3.1) and writing (i.e., $\mathbf{w}_t^{\text{W}}$ in Section 3.2) for the same vector-state $\mathbf{s}_t$. We conjecture that this acts like a regularization mechanism to encourage the content of reading and writing to be similar to each other.

3.3 Some Analysis

The writing operation in Eq. (13) at time t can be viewed as an nonlinear way to combine the previous memory-state $\mathbf{M}_{t-1}^{\text{B}}$ and the newly updated vector-state $\mathbf{s}_t$, where the nonlinearity comes from both the content-based addressing and the gating. This is in a way similar to the update of states in regular RNN, while we conjecture that the addressing strategy in MEMDEC makes it easier to selectively change some content updated (e.g., the relatively short-term content) while keeping other content less modified (e.g., the relatively long-term content).

The reading operation in Eq. (10) can "extract" the content from $\mathbf{M}_t^{\text{B}}$ relevant to the alignment (reading from $\mathbf{M}^{\text{S}}$) and prediction task at time t. This is in contrast with the regular RNN decoder including its gated variants, which takes the entire state vector to for this purpose. As one advantage, although only part of the information in $\mathbf{M}_t^{\text{B}}$ is used at t, the entire memory-state, which may store other information useful for later, will be carry over to time $t+1$ for memory-state update (writing).

4 Experiments on Chinese-English Translation

We test the memory-enhanced decoder to task of Chinese-to-English translation, where MEMDEC is put on the top of encoder same as in (Bahdanau et al., 2014).

4.1 Datasets and Evaluation metrics

Our training data for the translation task consists of 1.25M sentence pairs extracted from LDC corpora[2], with 27.9M Chinese words and 34.5M

[2]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07,

English words respectively. We choose NIST 2002 (MT02) dataset as our development set, and the NIST 2003 (MT03), 2004 (MT04) 2005 (MT05) and 2006 (MT06) datasets as our test sets. We use the case-insensitive 4-gram NIST BLEU score as our evaluation metric as our evaluation metric (Papineni et al., 2002).

4.2 Experiment settings

Hyper parameters In training of the neural networks, we limit the source and target vocabularies to the most frequent 30K words in both Chinese and English, covering approximately 97.7% and 99.3% of the two corpora respectively. The dimensions of word embedding is 512 and the size of the hidden layer is 1024. The dimemsion of each cell in $\mathbf{M}^{\text{B}}$ is set to 1024 and the number of cells n is set to 8.

Training details We initialize the recurrent weight matrices as random orthogonal matrices. All the bias vectors were initialize to zero. For other parameters, we initialize them by sampling each element from the Gaussian distribution of mean 0 and variance 0.01^2. Parameter optimization is performed using stochastic gradient descent. Adadelta (Zeiler, 2012) is used to automatically adapt the learning rate of each parameter ($\epsilon = 10^{-6}$ and $\rho = 0.95$). To avoid gradients explosion, the gradients of the cost function which had ℓ_2 norm larger than a predefined threshold 1.0 was normalized to the threshold (Pascanu et al., 2013). Each SGD is of a minibatch of 80 sentences. We train our NMT model with the sentences of length up to 50 words in training data, while for moses system we use the full training data.

Memory Initialization Each memory cell is initialized with the source sentence hidden state computed as

$$\mathbf{M}^{\text{B}}(i) = \mathbf{m} + \nu_i \quad (14)$$

$$\mathbf{m} = \sigma(\mathbf{W}_{\text{INI}} \sum_{i=0}^{i=T_x} \mathbf{h}_i)/T_x \quad (15)$$

where $\mathbf{W}_{\text{INI}} \in \mathbb{R}^{m \times 2 \cdot m}$; σ is $\tanh$ function. $\mathbf{m}$ makes a nonlinear transformation of the source sentence information. ν_i is a random vector sampled from $\mathcal{N}(0, 0.1)$.

Dropout we also use dropout for our NMT baseline model and MEMDEC to avoid overfitting (Hinton et al., 2012). The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. In the simplest case, each unit is omitted with a fixed probability p, namely dropout rate. In our experiments, dropout was applied only on the output layer and the dropout rate is set to 0.5. We also try other strategy such as dropout at word embeddings or RNN hidden states but fail to get further improvements.

Pre-training For MEMDEC, the objective function is a highly non-convex function of the parameters with more complicated landscape than that for decoder without external memory, rendering direct optimization over all the parameters rather difficult. Inspired by the effort on easing the training of very deep architectures (Hinton and Salakhutdinov, 2006), we propose a simple pre-training strategyFirst we train a regular attention-based NMT model without external memory. Then we use the trained NMT model to initialize the parameters of encoder and parameters of MEMDEC, except those related to memory-state (i.e., $\{\mathbf{W}^{\text{R}}_a, \mathbf{U}^{\text{R}}_a, \mathbf{v}, \mathbf{w}^{\text{R}}_g, \mathbf{W}^{\text{ERS}}, \mathbf{W}^{\text{ADD}}\}$). After that, we fine-tune all the parameters of NMT with MEMDEC decoder, including the parameters initialized with pre-training and those associated with accessing memory-state.

4.3 Comparison systems

We compare our method with three state-of-the-art systems:

- **Moses:** an open source phrase-based translation system [3]: with default configuration and a 4-gram language model trained on the target portion of training data.

LDC2004T08 and LDC2005T06.

[3] http://www.statmt.org/moses/

SYSTEM	MT03	MT04	MT05	MT06	AVE.
Groundhog	31.92	34.09	31.56	31.12	32.17
RNNsearch⋆	33.11	37.11	33.04	32.99	34.06
RNNsearch⋆ + coverage	34.49	38.34	34.91	34.25	35.49
MEMDEC	**36.16**	**39.81**	**35.91**	**35.98**	**36.95**
Moses	31.61	33.48	30.75	30.85	31.67

Table 1: Case-insensitive BLEU scores on Chinese-English translation. Moses is the state-of-the-art phrase-based statistical machine translation system. For RNNsearch, we use the open source system Groundhog as our baseline. The strong baseline, denoted RNNsearch⋆, also adopts *feedback attention* and *dropout*. The *coverage* model on top of RNNsearch⋆ has significantly improved upon its published version (Tu et al., 2016), which achieves the best published result on this training set. For MEMDEC the number of cells is set to 8.

pre-training	n	MT03	MT04	MT05	MT06	Ave.
N	4	35.29	37.36	34.58	33.32	35.11
Y	4	35.39	39.16	35.33	35.02	36.22
Y	6	35.63	39.29	35.61	34.92	36.58
Y	8	36.16	39.81	35.91	35.98	36.95
Y	10	36.46	38.86	34.46	35.00	36.19
Y	12	35.92	39.09	35.31	35.12	36.37

Table 2: MEMDEC performances of different memory size.

- **RNNSearch:** an attention-based NMT model with default settings. We use the open source system GroundHog as our NMT baseline[4].
- **Coverage model**: a state-of-the-art variant of attention-based NMT model (Tu et al., 2016) which improves the attention mechanism through modelling a soft coverage on the source representation.

4.4 Results

The main results of different models are given in Table 1. Clearly MEMDEC leads to remarkable improvement over Moses (+5.28 BLEU) and Groundhog (+4.78 BLEU). The *feedback attention* gains +1.06 BLEU score on top of Groundhog on average, while together with *dropout* adds another +0.83 BLEU score, which constitute the 1.89 BLEU gain of RNNsearch⋆ over Groundhog. Compared to RNNsearch⋆ MEMDEC is +2.89 BLEU score higher, showing the modeling power gained from the external memory. Finally, we also compare MEMDEC with the state-of-the-art attention-based NMT with COVERAGE mechanism(Tu et al., 2016), which is about 2 BLEU over than the published result after adding fast attention and dropout. In this comparison MEMDEC wins with big margin (+1.46 BLEU score).

[4] https://github.com/lisa-groundhog/GroundHog

4.5 Model selection

Pre-training plays an important role in optimizing the memory model. As can be seen in Tab.2, pre-training improves upon our baseline +1.11 BLEU score on average, but even without pre-training our model still gains +1.04 BLEU score on average. Our model is rather robust to the memory size: with merely four cells, our model will be over 2 BLEU higher than RNNsearch⋆. This further verifies our conjecture the the external memory is mostly used to store part of the source and history of target sentence.

4.6 Case study

We show in Table 5 sample translations from Chinese to English, comparing mainly MEMDEC

src	恩达依兹耶说:“签署(2003年11月停火)协定的各方,最迟必须在元月五日以前把战士的驻扎地点安顿完毕 。”
ref	“All *parties* that signed the *(November 2003 ceasefire)* accord should finish the cantoning of their fighters by January 5, 2004, at the latest,” Ndayizeye said.
MEMDEC	UNK said, “ the *parties involved in the ceasefire agreement on November 2003* will have to be completed by January 5, 2004. ”
base	“The signing of the **agreement (UNK-fire) agreement in the November 2003 ceasefire** must be completed by January 5, 2004.
src	代表团成员告诉今日美国报说,布希政府已批准美国代表团预定元月六日至十日展开的北韩之行。
ref	Members of the delegation told *US Today* that the Bush administration had *approved the US delegation' s visit* to North Korea from January 6 to 10.
MEMDEC	The delegation told the *US today* that the Bush administration has *approved the US delegation's visit* to north Korea from 6 to 10 january .
base	The delegation told the **US** that the Bush administration has **approved the US to begin his visit** to north Korea from 6 to 10 January.

Table 3: Sample translations-for each example, we show the source(src), the human translation (ref),the translation from our memory model MEMDEC and the translation from RNNsearch(equipped with fast attention and dropout).We italicise some *correct* translation segments and highlight a few **wrong** ones in bold.

and the RNNsearch model for its pre-training. It is appealing to observe that MEMDEC can produce more fluent translation results and better grasp the semantic information of the sentence.

5 Related Work

There is a long thread of work aiming to improve the ability of RNN in remembering long sequences, with the long short-term memory RNN (LSTM) (Hochreiter and Schmidhuber, 1997) being the most salient examples and GRU (Cho et al., 2014) being the most recent one. Those works focus on designing the dynamics of the RNN through new dynamic operators and appropriate gating, while still keeping vector form RNN states. MEMDEC, on top of the gated RNN, explicitly adds matrix-form memory equipped with content-based addressing to the system, hence greatly improving the power of the decoder RNN in representing the information important for the translation task.

MEMDEC is obviously related to the recent effort on attaching an external memory to neural networks, with two most salient examples being Neural Turing Machine (NTM) (Graves et al., 2014) and Memory Network (Weston et al., 2014). In fact MEMDEC can be viewed as a special case of NTM, with specifically designed reading (from two different types of memory) and writing mechanism for the translation task. Quite remarkably MEMDEC is among the rare instances of NTM which significantly improves upon state-of-the-arts on a real-world NLP task with large training corpus.

Our work is also related to the recent work on machine reading (Cheng et al., 2016), in which the machine reader is equipped with a memory tape, enabling the model to directly read all the previous hidden state with an attention mechanism. Different from their work, we use an external bounded memory and make an abstraction of previous information. In (Meng et al., 2015), Meng et. al. also proposed a deep architecture for sequence-to-sequence learning with stacked layers of memory to store the intermediate representations, while our external memory was applied within a sequence.

6 Conclusion

We propose to enhance the RNN decoder in a neural machine translator (NMT) with external memory. Our empirical study on Chinese-English translation shows that it can significantly improve the performance of NMT.

References

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

[Cheng et al.2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

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

[Graves et al.2014] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[Hinton and Salakhutdinov2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Hinton et al.2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hu et al.2015] Baotian Hu, Zhaopeng Tu, Zhengdong Lu, and Hang Li. 2015. Context-dependent translation selection using convolutional neural network.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Meng et al.2015] Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. *arXiv preprint arXiv:1506.06442*.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Pascanu et al.2013] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

[Tu et al.2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ArXiv eprints, January*.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

[Xie et al.2011] Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Semi-Supervised Learning of Sequence Models with the Method of Moments

Zita Marinho[*♯] **André F. T. Martins**[†♡◇] **Shay B. Cohen**[♣] **Noah A. Smith**[♠]

[*]Instituto de Sistemas e Robótica, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[♯]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[♡]Unbabel Lda, Rua Visconde de Santarém, 67-B, 1000-286 Lisboa, Portugal
[◇]Priberam Labs, Alameda D. Afonso Henriques, 41, 2º, 1000-123 Lisboa, Portugal
[♣]School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK
[♠]Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA

zmarinho@cmu.edu, andre.martins@unbabel.com,
scohen@inf.ed.ac.uk, nasmith@cs.washington.edu

Abstract

We propose a fast and scalable method for semi-supervised learning of sequence models, based on anchor words and moment matching. Our method can handle hidden Markov models with feature-based log-linear emissions. Unlike other semi-supervised methods, no decoding passes are necessary on the unlabeled data and no graph needs to be constructed—only one pass is necessary to collect moment statistics. The model parameters are estimated by solving a small quadratic program for each feature. Experiments on part-of-speech (POS) tagging for Twitter and for a low-resource language (Malagasy) show that our method can learn from very few annotated sentences.

1 Introduction

Statistical learning of NLP models is often limited by the scarcity of annotated data. Weakly supervised methods have been proposed as an alternative to laborious manual annotation, combining large amounts of unlabeled data with limited resources, such as tag dictionaries or small annotated datasets (Merialdo, 1994; Smith and Eisner, 2005; Garrette et al., 2013). Unfortunately, most semi-supervised learning algorithms for the structured problems found in NLP are computationally expensive, requiring multiple decoding passes through the unlabeled data, or expensive similarity graphs. More scalable learning algorithms are in demand.

In this paper, we propose a moment-matching method for semi-supervised learning of sequence models. Spectral learning and moment-matching approaches have recently proved a viable alternative to expectation-maximization (EM) for unsupervised learning (Hsu et al., 2012; Balle and Mohri, 2012; Bailly et al., 2013), supervised learning with latent variables (Cohen and Collins, 2014; Quattoni et al., 2014; Stratos et al., 2013) and topic modeling (Arora et al., 2013; Nguyen et al., 2015). These methods have learnability guarantees, do not suffer from local optima, and are computationally less demanding.

Unlike spectral methods, ours does not require an orthogonal decomposition of any matrix or tensor. Instead, it considers a more restricted form of supervision: words that have unambiguous annotations, so-called **anchor words** (Arora et al., 2013). Rather than identifying anchor words from unlabeled data (Stratos et al., 2016), we extract them from a small labeled dataset or from a dictionary. Given the anchor words, the estimation of the model parameters can be made efficient by collecting moment statistics from unlabeled data, then solving a small quadratic program for each word.

Our contributions are as follows:

- We adapt anchor methods to semi-supervised learning of generative sequence models.
- We show how our method can also handle log-linear feature-based emissions.
- We apply this model to POS tagging. Our experiments on the Twitter dataset introduced by Gimpel et al. (2011) and on the dataset introduced by Garrette et al. (2013) for Malagasy, a low-resource language, show that our method does particularly well with very little labeled data, outperforming semi-supervised EM and self-training.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 287–296,
Austin, Texas, November 1-5, 2016.

2 Sequence Labeling

In this paper, we address the problem of sequence labeling. Let $\boldsymbol{x}_{1:L} = \langle x_1, \ldots, x_L \rangle$ be a sequence of L input observations (for example, words in a sentence). The goal is to predict a sequence of labels $\boldsymbol{h}_{1:L} = \langle h_1, \ldots, h_L \rangle$, where each h_i is a label for the observation x_i (for example, the word's POS tag).

We start by describing two generative sequence models: hidden Markov models (HMMs, §2.1), and their generalization with emission features (§2.2). Later, we propose a weakly-supervised method for estimating these models' parameters (§3–§4) based only on observed statistics of words and contexts.

2.1 Hidden Markov Models

We define random variables $\boldsymbol{X} := \langle X_1, \ldots, X_L \rangle$ and $\boldsymbol{H} := \langle H_1, \ldots, H_L \rangle$, corresponding to observations and labels, respectively. Each X_i is a random variable over a set $\mathcal{X}$ (the vocabulary), and each H_i ranges over $\mathcal{H}$ (a finite set of "states" or "labels"). We denote the vocabulary size by $V = |\mathcal{X}|$, and the number of labels by $K = |\mathcal{H}|$. A first-order HMM has the following generative scheme:

$$p(\boldsymbol{X} = \boldsymbol{x}_{1:L}, \boldsymbol{H} = \boldsymbol{h}_{1:L}) := \quad (1)$$
$$\prod_{\ell=1}^{L} p(X_\ell{=}x_\ell \mid H_\ell{=}h_\ell) \prod_{\ell=0}^{L} p(H_{\ell+1}{=}h_{\ell+1} \mid H_\ell{=}h_\ell),$$

where we have defined $h_0 = \text{START}$ and $h_{L+1} = \text{STOP}$. We adopt the following notation for the parameters:

- The **emission matrix** $\mathbf{O} \in \mathbb{R}^{V \times K}$, defined as $O_{x,h} := p(X_\ell = x \mid H_\ell = h), \forall h \in \mathcal{H}, x \in \mathcal{X}$.
- The **transition matrix** $\mathbf{T} \in \mathbb{R}^{(K+2)\times(K+2)}$, defined as $T_{h,h'} := p(H_{\ell+1} = h \mid H_\ell = h')$, for every $h, h' \in \mathcal{H} \cup \{\text{START}, \text{STOP}\}$. This matrix satisfies $\mathbf{T}^\top \mathbf{1} = \mathbf{1}$.[1]

Throughout the rest of the paper we will adopt $X \equiv X_\ell$ and $H \equiv H_\ell$ to simplify notation, whenever the index ℓ is clear from the context. Under this generative process, predicting the most probable label sequence $\boldsymbol{h}_{1:L}$ given observations $\boldsymbol{x}_{1:L}$ is accomplished with the Viterbi algorithm in $O(LK^2)$ time.

If labeled data are available, the model parameters $\mathbf{O}$ and $\mathbf{T}$ can be estimated with the maximum likelihood principle, which boils down to a simple counting of events and normalization. If we only have unlabeled data, the traditional approach is the expectation-maximization (EM) algorithm, which alternately decodes the unlabeled examples and updates the model parameters, requiring multiple passes over the data. The same algorithm can be used in semi-supervised learning when labeled and unlabeled data are combined, by initializing the model parameters with the supervised estimates and interpolating the estimates in the M-step.

2.2 Feature-Based Hidden Markov Models

Sequence models with log-linear emissions have been considered by Smith and Eisner (2005), in a discriminative setting, and by Berg-Kirkpatrick et al. (2010), as generative models for POS induction. Feature-based HMMs (FHMMs) define a feature function for words, $\boldsymbol{\phi}(X) \in \mathbb{R}^W$, which can be discrete or continuous. This allows, for example, to indicate whether an observation, corresponding to a word, starts with an uppercase letter, contains digits or has specific affixes. More generally, it helps with the treatment of out-of-vocabulary words. The emission probabilities are modeled as K conditional distributions parametrized by a log-linear model, where the $\boldsymbol{\theta}_h \in \mathbb{R}^W$ represent feature weights:

$$p(X = x \mid H = h) := \exp(\boldsymbol{\theta}_h^\top \boldsymbol{\phi}(x)) / Z(\boldsymbol{\theta}_h). \quad (2)$$

Above, $Z(\boldsymbol{\theta}_h) := \sum_{x' \in \mathcal{X}} \exp(\boldsymbol{\theta}_h^\top \boldsymbol{\phi}(x'))$ is a normalization factor. We will show in §4 how our moment-based semi-supervised method can also be used to learn the feature weights $\boldsymbol{\theta}_h$.

3 Semi-Supervised Learning via Moments

We now describe our moment-based semi-supervised learning method for HMMs. Throughout, we assume the availability of a small labeled dataset $\mathcal{D}_L$ and a large unlabeled dataset $\mathcal{D}_U$.

The full roadmap of our method is shown as Algorithm 1. Key to our method is the decomposition of a **context-word moment matrix** $\mathbf{Q} \in \mathbb{R}^{C \times V}$, which counts co-occurrences of words and contexts,

[1] That is, it satisfies $\sum_{h=1}^{K} p(H_{\ell+1} = h \mid H_\ell = h') + p(H_{\ell+1} = \text{STOP} \mid H_\ell = h') = 1$; and also $\sum_{h=1}^{K} p(H_1 = h \mid H_0 = \text{START}) = 1$.

Algorithm 1 Semi-Supervised Learning of HMMs with Moments

Input: Labeled dataset $\mathcal{D}_L$, unlabeled dataset $\mathcal{D}_U$
Output: Estimates of emissions $\mathbf{O}$ and transitions $\mathbf{T}$

1: Estimate context-word moments $\widehat{\mathbf{Q}}$ from $\mathcal{D}_U$ (Eq. 5)
2: **for** each label $h \in \mathcal{H}$ **do**
3: Extract set of anchor words $\mathcal{A}(h)$ from $\mathcal{D}_L$ (§3.2)
4: **end for**
5: Estimate context-label moments $\widehat{\mathbf{R}}$ from anchors and $\mathcal{D}_U$ (Eq. 12)
6: **for** each word $w \in [V]$ **do**
7: Solve the QP in Eq. 14 to obtain $\boldsymbol{\gamma}_w$ from $\widehat{\mathbf{Q}}, \widehat{\mathbf{R}}$
8: **end for**
9: Estimate emissions $\mathbf{O}$ from $\boldsymbol{\Gamma}$ via Eq. 15
10: Estimate transitions $\mathbf{T}$ from $\mathcal{D}_L$
11: Return $\langle \mathbf{O}, \mathbf{T} \rangle$

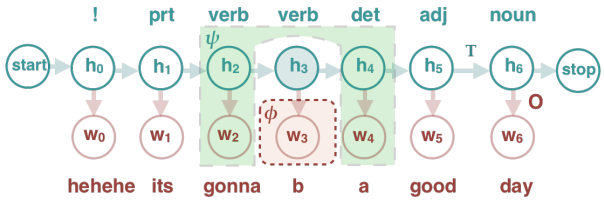


Figure 1: HMM, context (green) conditionally independent of present (red) w_ℓ given state h_ℓ.

and will be formally defined in §3.1. Such co-occurrence matrices are often collected in NLP, for various problems, ranging from dimensionality reduction of documents using latent semantic indexing (Deerwester et al., 1990; Landauer et al., 1998), distributional semantics (Schütze, 1998; Levy et al., 2015) and word embedding generation (Dhillon et al., 2015; Osborne et al., 2016). We can build such a moment matrix entirely from the unlabeled data $\mathcal{D}_U$. The same unlabeled data is used to build an estimate of a **context-label moment matrix** $\mathbf{R} \in \mathbb{R}^{C \times K}$, as explained in §3.3. This is done by first identifying words that are unambiguously associated with each label h, called **anchor words**, with the aid of a few labeled data; this is outlined in §3.2. Finally, given empirical estimates of $\mathbf{Q}$ and $\mathbf{R}$, we estimate the emission matrix $\mathbf{O}$ by solving a small optimization problem independently per word (§3.4). The transition matrix $\mathbf{T}$ is obtained directly from the labeled dataset $\mathcal{D}_L$ by maximizing the likelihood.

3.1 Moments of Contexts and Words

To formalize the notion of "context," we introduce the shorthand $\boldsymbol{Z}_\ell := \langle \boldsymbol{X}_{1:(\ell-1)}, \boldsymbol{X}_{(\ell+1):L} \rangle$. Importantly, the HMM in Eq. 1 entails the following conditional independence assumption: X_ℓ is conditionally independent of the surrounding context Z_ℓ given the hidden state H_ℓ. This is illustrated in Figure 1, using POS tagging as an example task.

We introduce a vector of **context features** $\boldsymbol{\psi}(\boldsymbol{Z}_\ell) \in \mathbb{R}^C$, which may look arbitrarily within the context $\boldsymbol{Z}_\ell$ (left or right), but not at X_ℓ itself. These features could be "one-hot" representations or other reduced-dimensionality embeddings (as described later in §5). Consider the word $w \in \mathcal{X}$ an instance of $X \equiv X_\ell$. A pivotal matrix in our formulation is the matrix $\mathbf{Q} \in \mathbb{R}^{C \times V}$, defined as:

$$Q_{c,w} := \mathbb{E}[\psi_c(\boldsymbol{Z}) \mid X = w]. \tag{3}$$

Expectations here are taken with respect to the probabilistic model in Eq. 1 that generates the data. The following quantities will also be necessary:

$$q_c := \mathbb{E}[\psi_c(\boldsymbol{Z})], \quad p_w := p(X = w). \tag{4}$$

Since all the variables in Eqs. 3–4 are observed, we can easily obtain empirical estimates by taking expectations over the unlabeled data:

$$\widehat{Q}_{c,w} = \frac{\sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \psi_c(\boldsymbol{z}) \mathbb{1}(x = w)}{\sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \mathbb{1}(x = w)}, \tag{5}$$

$$\widehat{q}_c = \sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \psi_c(\boldsymbol{z}) \Big/ |\mathcal{D}_U|, \tag{6}$$

$$\widehat{p}_w = \sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \mathbb{1}(x = w) \Big/ |\mathcal{D}_U|. \tag{7}$$

where we take $\mathbb{1}(x = w)$ to be the indicator for word w. Note that, under our modeling assumptions, $\mathbf{Q}$ decomposes in terms of its hidden states:

$$\mathbb{E}[\psi_c(\boldsymbol{Z}) \mid X = w] = \sum_{h \in \mathcal{H}} p(H = h \mid X = w) \mathbb{E}[\psi_c(\boldsymbol{Z}) \mid H = h] \tag{8}$$

The reason why this holds is that, as stated above, $\boldsymbol{Z}$ and X are conditionally independent given H.

3.2 Anchor Words

Following Arora et al. (2013) and Cohen and Collins (2014), we identify **anchor words** whose hidden

state is assumed to be deterministic, regardless of context. In this work, we generalize this notion to more than one anchor word per label, for improved context estimates. This allows for more flexible forms of anchors with weak supervision. For each state $h \in \mathcal{H}$, let its set of anchor words be

$$\begin{aligned}\mathcal{A}(h) &= \{w \in \mathcal{X} : p(H = h \mid X = w) = 1\} \quad (9)\\ &= \left\{w \in \mathcal{X} : O_{w,h}{>}0 \wedge O_{w,h'}{=}0, \forall h' {\neq} h\right\}.\end{aligned}$$

That is, $\mathcal{A}(h)$ is the set of unambiguous words that always take the label h. This can be estimated from the labeled dataset $\mathcal{D}_L$ by collecting the most frequent unambiguous words for each label.

Algorithms for identifying $\mathcal{A}(h)$ from unlabeled data alone were proposed by Arora et al. (2013) and Zhou et al. (2014), with application to topic models. Our work differs in which we do not aim to discover anchor words from pure unlabeled data, but rather exploit the fact that small amounts of labeled data are commonly available in many NLP tasks—better anchors can be extracted easily from such small labeled datasets. In §5 we give a more detailed description of the selection process.

3.3 Moments of Contexts and Labels

We define the matrix $\mathbf{R} \in \mathbb{R}^{C \times K}$ as follows:

$$R_{c,h} := \mathbb{E}[\psi_c(\boldsymbol{Z}) \mid H = h]. \quad (10)$$

Since the expectation in Eq. 10 is conditioned on the (unobserved) label h, we cannot directly estimate it using moments of observed variables, as we do for $\mathbf{Q}$. However, if we have identified sets of anchor words for each label $h \in \mathcal{H}$, we have:

$$\begin{aligned}&\mathbb{E}[\psi_c(\boldsymbol{Z}) \mid X \in \mathcal{A}(h)] =\\ &= \sum_{h'} \mathbb{E}[\psi_c(\boldsymbol{Z}) \mid H = h'] \underbrace{p(H = h' \mid X \in \mathcal{A}(h))}_{=\mathbb{1}(h'=h)}\\ &= R_{c,h}. \quad (11)\end{aligned}$$

Therefore, given the set of anchor words $\mathcal{A}(h)$, the hth column of $\mathbf{R}$ can be estimated in a single pass over the unlabeled data, as follows:

$$\widehat{R}_{c,h} = \frac{\sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \psi_c(\boldsymbol{z}) \mathbb{1}(x \in \mathcal{A}(h))}{\sum_{x,\boldsymbol{z} \in \mathcal{D}_U} \mathbb{1}(x \in \mathcal{A}(h))} \quad (12)$$

3.4 Emission Distributions

We can now put all the ingredients above together to estimate the emission probability matrix $\mathbf{O}$. The procedure we propose here is computationally very efficient, since only one pass is required over the unlabeled data, to collect the co-occurrence statistics $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{R}}$. The emissions will be estimated from these moments by solving a small problem independently for each word. Unlike EM and self-training, no decoding is necessary, only counting and normalizing; and unlike label propagation methods, there is requirement to build a graph with the unlabeled data.

The crux of our method is the decomposition in Eq. 8, which is combined with the one-to-one correspondence between labels h and anchor words $\mathcal{A}(h)$. We can rewrite Eq. 8 as:

$$Q_{c,w} = \sum_h R_{c,h}\ p(H = h \mid X = w). \quad (13)$$

In matrix notation, we have $\mathbf{Q} = \mathbf{R}\boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma} \in \mathbb{R}^{K \times V}$ is defined as $\Gamma_{h,w} := p(H = h \mid X = w)$.

If we had infinite unlabeled data, our moment estimates $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{R}}$ would be perfect and we could solve the system of equations in Eq. 13 to obtain $\boldsymbol{\Gamma}$ exactly. Since we have finite data, we resort to a least squares solution. This corresponds to solving a simple quadratic program (QP) per word, independent from all the other words, as follows. Denote by $\mathbf{q}_w := \mathbb{E}[\boldsymbol{\psi}(\boldsymbol{Z}) \mid X = w] \in \mathbb{R}^C$ and by $\boldsymbol{\gamma}_w := p(H = \cdot \mid X = w) \in \mathbb{R}^K$ the wth columns of $\mathbf{Q}$ and $\boldsymbol{\Gamma}$, respectively. We estimate the latter distribution following Arora et al. (2013):

$$\begin{aligned}\widehat{\boldsymbol{\gamma}}_w \;=\; &\arg\min_{\boldsymbol{\gamma}_w} \quad \|\mathbf{q}_w - \mathbf{R}\boldsymbol{\gamma}_w\|_2^2\\ &\text{s.t.} \qquad \mathbf{1}^\top \boldsymbol{\gamma}_w = 1,\ \boldsymbol{\gamma}_w \geq \mathbf{0}.\end{aligned} \quad (14)$$

Note that this QP is very small—it has only K variables—hence, we can solve it very quickly (1.7 ms on average, in Gurobi, with $K = 12$).

Given the probability tables for $p(H = h \mid X = w)$, we can estimate the emission probabilities $\mathbf{O}$ by direct application of Bayes rule:

$$\widehat{O}_{w,h} = \frac{p(H = h \mid X = w) \times p(X = w)}{p(H = h)} \quad (15)$$

$$= \frac{\widehat{\gamma}_{w,c} \times \overbrace{\widehat{p}_w}^{\text{Eq. 7}}}{\sum_{w'} \widehat{\gamma}_{w',c} \times \widehat{p}_{w'}}. \quad (16)$$

These parameters are guaranteed to lie in the probability simplex, avoiding the need of heuristics for dealing with "negative" and "unnormalized" probabilities required by prior work in spectral learning (Cohen et al., 2013).

3.5 Transition Distributions

It remains to estimate the transition matrix $\mathbf{T}$. For the problems tackled in this paper, the number of labels K is small, compared to the vocabulary size V. The transition matrix has only $O(K^2)$ degrees of freedom, and we found it effective to estimate it using the labeled sequences in $\mathcal{D}_L$ alone, without any refinement. This was done by smoothed maximum likelihood estimation on the labeled data, which boils down to counting occurrences of consecutive labels, applying add-one smoothing to avoid zero probabilities for unobserved transitions, and normalizing.

For problems with numerous labels, a possible alternative is the composite likelihood method (Chaganty and Liang, 2014). Given $\widehat{\mathbf{O}}$, the maximization of the composite log-likelihood function leads to a convex optimization problem that can be efficiently optimized with an EM algorithm. A similar procedure was carried out by Cohen and Collins (2014).[2]

4 Feature-Based Emissions

Next, we extend our method to estimate the parameters of the FHMM in §2.2. Other than contextual features $\boldsymbol{\psi}(\boldsymbol{Z}) \in \mathbb{R}^C$, we also assume a feature encoding function for words, $\boldsymbol{\phi}(X) \in \mathbb{R}^W$. Our framework, illustrated in Algorithm 2, allows for both discrete and continuous word and context features. Lines 2–5 are the same as in Algorithm 1, replacing word occurrences with expected values of word features (we redefine $\mathbf{Q}$ and $\mathbf{\Gamma}$ to cope with features instead of words). The main difference with respect to Algorithm 1 is that we do not estimate emission probabilities; rather, we first estimate the **mean parameters** (feature expectations $\mathbb{E}[\boldsymbol{\phi}(X) \mid H = h]$), by solving one QP for each emission feature; and then we solve a convex optimization problem, for each label h, to recover the log-linear weights over emission features (called **canonical parameters**).

[2] In preliminary experiments, the compositional likelihood method was not competitive with estimating the transition matrices directly from the labeled data, on the datasets described in §6; results are omitted due to lack of space. However, this may be a viable alternative if there is no labeled data and the anchors are extracted from gazetteers or a dictionary.

Algorithm 2 Semi-Supervised Learning of Feature-Based HMMs with Moments

Input: Labeled dataset $\mathcal{D}_L$, unlabeled dataset $\mathcal{D}_U$
Output: Emission log-linear parameters $\mathbf{\Theta}$ and transitions $\mathbf{T}$

1: Estimate context-word moments $\widehat{\mathbf{Q}}$ from $\mathcal{D}_U$ (Eq. 20)
2: **for** each label $h \in \mathcal{H}$ **do**
3: Extract set of anchor words $\mathcal{A}(h)$ from $\mathcal{D}_L$ (§3.2)
4: **end for**
5: Estimate context-label moments $\widehat{\mathbf{R}}$ from the anchors and $\mathcal{D}_U$ (Eq. 12)
6: **for** each word feature $j \in [W]$ **do**
7: Solve the QP in Eq. 22 to obtain $\boldsymbol{\gamma}_j$ from $\widehat{\mathbf{Q}}, \widehat{\mathbf{R}}$
8: **end for**
9: **for** each label $h \in \mathcal{H}$ **do**
10: Estimate the mean parameters $\boldsymbol{\mu}_h$ from $\mathbf{\Gamma}$ (Eq. 24)
11: Estimate the canonical parameters $\boldsymbol{\theta}_h$ from $\boldsymbol{\mu}_h$ by solving Eq. 25
12: **end for**
13: Estimate transitions $\mathbf{T}$ from $\mathcal{D}_L$
14: Return $\langle \mathbf{\Theta}, \mathbf{T} \rangle$

4.1 Estimation of Mean Parameters

First of all, we replace word probabilities by expectations over word features. We redefine the matrix $\mathbf{\Gamma} \in \mathbb{R}^{K \times W}$ as follows:

$$\Gamma_{h,j} := \frac{p(H = h) \times \mathbb{E}[\phi_j(X) \mid H = h]}{\mathbb{E}[\phi_j(X)]}. \quad (17)$$

Note that, with one-hot word features, we have $\mathbb{E}[\phi_w(X) \mid H = h] = P(X = w \mid H = h)$, $\mathbb{E}[\phi_w(X)] = p(X = w)$, and therefore $\Gamma_{h,w} = p(H = h \mid X = w)$, so this can be regarded as a generalization of the framework in §3.4.

Second, we redefine the context-word moment matrix $\mathbf{Q}$ as the following matrix in $\mathbb{R}^{C \times W}$:

$$Q_{c,j} = \mathbb{E}\left[\psi_c(\boldsymbol{Z}) \times \phi_j(X)\right] / \mathbb{E}[\phi_j(X)]. \quad (18)$$

Again, note that we recover the previous $\mathbf{Q}$ if we use one-hot word features. We then have the following generalization of Eq. 13:

$$\mathbb{E}\left[\psi_c(\boldsymbol{Z}) \times \phi_j(X)\right] / \mathbb{E}[\phi_j(X)] = \quad (19)$$
$$\textstyle\sum_h \mathbb{E}\left[\psi_c(\boldsymbol{Z}) \mid H = h\right] \frac{P(H=h)\mathbb{E}[\phi_j(X)|H=h]}{\mathbb{E}[\phi_j(X)]},$$

or, in matrix notation, $\mathbf{Q} = \mathbf{R}\mathbf{\Gamma}$.

Again, matrices $\mathbf{Q}$ and $\mathbf{R}$ can be estimated from data by collecting empirical feature expectations over unlabeled sequences of observations. For $\mathbf{R}$ use Eq. 12 with no change; for $\mathbf{Q}$ replace Eq. 5 by

$$\widehat{Q}_{c,j} = \sum_{x,\boldsymbol{z}\in\mathcal{D}_U} \psi_c(\boldsymbol{z})\phi_j(x) \Big/ \sum_{x,\boldsymbol{z}\in\mathcal{D}_U} \phi_j(x). \quad (20)$$

Let $\mathbf{q}_j \in \mathbb{R}^C$ and $\boldsymbol{\gamma}_j \in \mathbb{R}^K$ be columns of $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{\Gamma}}$, respectively. Note that we must have

$$\mathbf{1}^\top \boldsymbol{\gamma}_j = \sum_h \frac{P(H=h)\mathbb{E}[\phi_j(X)|H=h]}{\mathbb{E}[\phi_j(X)]} = 1, \quad (21)$$

since $\mathbb{E}[\phi_j(X)] = \sum_h P(H = h)\mathbb{E}\left[\phi_j(X) \mid H = h\right]$. We rewrite the QP to minimize the squared difference for each dimension j independently:

$$\widehat{\boldsymbol{\gamma}}_j = \arg\min_{\boldsymbol{\gamma}_j} \left\|\mathbf{q}_j - \mathbf{R}\boldsymbol{\gamma}_j\right\|_2^2 \text{ s.t. } \mathbf{1}^\top\boldsymbol{\gamma}_j = 1. \quad (22)$$

Note that, if $\boldsymbol{\phi}(x) \geq \mathbf{0}$ for all $x \in \mathcal{X}$, then we must have $\boldsymbol{\gamma}_j \geq \mathbf{0}$, and therefore we may impose this inequality as an additional constraint.

Let $\bar{\boldsymbol{\gamma}} \in \mathbb{R}^K$ be the vector of state probabilities, with entries $\bar{\gamma}_h := p(H = h)$ for $h \in \mathcal{H}$. This vector can also be recovered from the unlabeled dataset and the set of anchors, by solving another QP that aggregates information for all words:

$$\bar{\boldsymbol{\gamma}} = \arg\min_{\bar{\boldsymbol{\gamma}}} \|\bar{\boldsymbol{q}} - \mathbf{R}\bar{\boldsymbol{\gamma}}\|_2^2 \text{ s.t. } \mathbf{1}^\top\bar{\boldsymbol{\gamma}} = 1,\ \bar{\boldsymbol{\gamma}} \geq \mathbf{0}. \quad (23)$$

where $\bar{\boldsymbol{q}} := \widehat{\mathbb{E}}[\boldsymbol{\psi}(\boldsymbol{Z})] \in \mathbb{R}^C$ is the vector whose entries are defined in Eq. 6.

Let $\boldsymbol{\mu}_h := \mathbb{E}[\boldsymbol{\phi}(X) \mid H = h] \in \mathbb{R}^W$ be the mean parameters of the distribution for each state h. These parameters are computed by solving W independent QPs (Eq. 22), yielding the matrix $\mathbf{\Gamma}$ defined in Eq. 17, and then applying the formula:

$$\mu_{h,j} = \Gamma_{j,h} \times \mathbb{E}[\phi_j(X)] / \bar{\gamma}_h, \quad (24)$$

with $\bar{\gamma}_h = p(H = h)$ estimated as in Eq. 23.

4.2 Estimation of Canonical Parameters

To compute a mapping from mean parameters $\boldsymbol{\mu}_h$ to canonical parameters $\boldsymbol{\theta}_h$, we use the well-known Fenchel-Legendre duality between the entropy and the log-partition function (Wainwright and Jordan, 2008). Namely, we need to solve the following convex optimization problem:

$$\widehat{\boldsymbol{\theta}}_h = \arg\max_{\boldsymbol{\theta}_h} \boldsymbol{\theta}_h^\top \boldsymbol{\mu}_h - \log Z(\boldsymbol{\theta}_h) + \epsilon\|\boldsymbol{\theta}_h\|, \quad (25)$$

where ϵ is a regularization constant.[3] In practice, this regularization is important, since it prevents $\boldsymbol{\theta}_h$ from growing unbounded whenever $\boldsymbol{\mu}_h$ falls outside the marginal polytope of possible mean parameters. We solve Eq. 25 with the limited-memory BFGS algorithm (Liu and Nocedal, 1989).

5 Method Improvements

In this section we detail three improvements to our moment-based method that had a practical impact.

Supervised Regularization. We add a supervised penalty term to Eq. 22 to keep the label posteriors $\boldsymbol{\gamma}_j$ close to the label posteriors estimated in the labeled set, $\boldsymbol{\gamma}'_j$, for every feature $j \in [W]$. The regularized least-squares problem becomes:

$$\min_{\boldsymbol{\gamma}_j} (1-\lambda)\|\mathbf{q}_j - \mathbf{R}\boldsymbol{\gamma}_j\|^2 + \lambda\|\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j\|^2 \text{ s.t. } \mathbf{1}^\top\boldsymbol{\gamma}_j = 1. \quad (26)$$

CCA Projections. A one-hot feature representation of words and contexts has the disadvantage that it grows with the vocabulary size, making the moment matrix $\mathbf{Q}$ too sparse. The number of contextual features and words can grow rapidly on large text corpora. Similarly to Cohen and Collins (2014) and Dhillon et al. (2015), we use canonical correlation analysis (CCA) to reduce the dimension of these vectors. We use CCA to form low-dimensional projection matrices for features of words $\mathbf{P}_W \in \mathbb{R}^{W\times D}$ and features of contexts $\mathbf{P}_C \in \mathbb{R}^{C\times D}$, with $D \ll \min\{W, C\}$. We use these projections on the original feature vectors and replace the these vectors with their projections.

Selecting Anchors. We collect counts of each word-label pair, and select up to 500 anchors with high conditional probability on the anchoring state $\widehat{p}(h \mid w)$. We tuned the probability threshold to

[3] As shown by Xiaojin Zhu (1999) and Yasemin Altun (2006), this regularization is equivalent, in the dual, to a "soft" constraint $\|\mathbb{E}_{\boldsymbol{\theta}_h}[\boldsymbol{\phi}(X) \mid H = h] - \boldsymbol{\mu}_h\|_2 \leq \epsilon$, as opposed to a strict equality.

select the anchors on the validation set, using steps of 0.1 in the unit interval, and making sure that all tags have at least one anchor. We also considered a frequency threshold, constraining anchors to occur more than 500 times in the unlabeled corpus, and four times in the labeled corpus. Note that past work used a single anchor word per state (i.e., $|\mathcal{A}(h)| = 1$). We found that much better results are obtained when $|\mathcal{A}(h)| \gg 1$, as choosing more anchors increases the number of samples used to estimate the context-label moment matrix $\widehat{\mathbf{R}}$, reducing noise.

6 Experiments

We evaluated our method on two tasks: POS tagging of Twitter text (in English), and POS tagging for a low-resource language (Malagasy). For all the experiments, we used the universal POS tagset (Petrov et al., 2012), which consists of $K = 12$ tags. We compared our method against supervised baselines (HMM and FHMM), which use the labeled data only, and two semi-supervised baselines that exploit the unlabeled data: self-training and EM. For the Twitter experiments, we also evaluated a stacked architecture in which we derived features from our model's predictions to improve a state-of-the-art POS tagger (MEMM).[4]

6.1 Twitter POS Tagging

For the Twitter experiment, we used the *Oct27* dataset of Gimpel et al. (2011), with the provided partitions (1,000 tweets for training and 328 for validation), and tested on the *Daily547* dataset (547 tweets). Anchor words were selected from the training partition as described in §5. We used 2.7M unlabeled tweets (O'Connor et al., 2010) to train the semi-supervised methods, filtering the English tweets as in Lui and Baldwin (2012), tokenizing them as in Owoputi et al. (2013), and normalizing at-mentions, URLs, and emoticons.

We used as word features $\phi(X)$ the word iself, as well as binary features for capitalization, titles, and digits (Berg-Kirkpatrick et al., 2010), the word shape, and the Unicode class of each character. Similarly to Owoputi et al. (2013), we also used suffixes and prefixes (up to length 3), and Twitter-specific features: whether the word starts with @, #, or *http://*. As contextual features $\psi(Z)$, we derive analogous features for the preceding and following words, before reducing dimensionality with CCA. We collect feature expectations for words and contexts that occur more than 20 times in the unlabeled corpus. We tuned hyperparameters on the development set: the supervised interpolation coefficient in Eq. 26, $\lambda \in \{0, 0.1, \ldots, \underline{1.0}\}$, and, for all systems, the regularization coefficient $\epsilon \in \{\underline{0.0001}, 0.001, 0.01, 0.1, 1, 10\}$. (Underlines indicate selected values.) The former controls how much we rely on the supervised vs. unsupervised estimates. For $\lambda = 1.0$ we used supervised estimates only for words that occur in the labeled corpus, all the remaining words rely solely on unsupervised estimates.

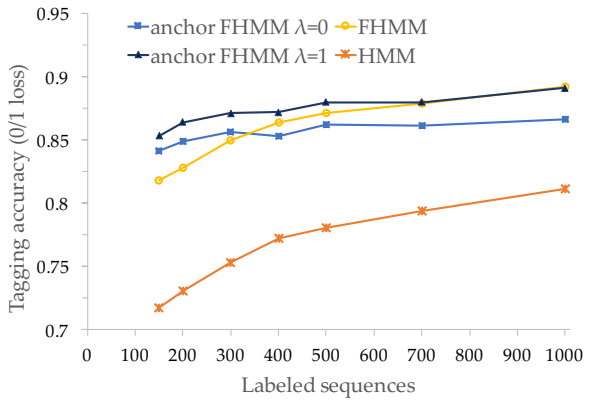


Figure 2: POS tagging accuracy in the Twitter data versus the number of labeled training sequences.

Varying supervision. Figure 2 compares the learning curves of our anchor-word method for the FHMM with the supervised baselines. We show the performance of the anchor methods without interpolation ($\lambda = 0$), and with supervised interpolation coefficient ($\lambda = 1$). When the amount of supervision is small, our method with and without interpolation outperforms all the supervised baselines. This improvement is gradually attenuated when more labeled sequences are used, with the supervised FHMM catching up when the full labeled dataset is used. The best model $\lambda = 1.0$ relies on supervised estimates for words that occur in the labeled corpus, and on anchor estimates for words that occur only in the unlabeled corpus. The unregular-

[4] http://www.ark.cs.cmu.edu/TweetNLP/

	HMM		FHMM	
Models / #sequences	150	1000	150	1000
Supervised baseline				
HMM	71.7	81.1	81.8	89.1
Semi-supervised baselines				
EM	77.2	83.1	81.8	89.1
self-training	78.2	86.1	83.4	**89.4**
Anchor Models				
anchors, $\lambda = 0.0$	83.0	85.5	84.1	86.7
anchors, $\lambda = 1.0$	**84.3**	**88.0**	**85.3**	89.1

Table 1: Tagging accuracies on Twitter. Shown are the supervised and semi-supervised baselines, and our moment-based method, trained with 150 training labeled sequences, and the full labeled corpus (1000 sequences).

ized model $\lambda = 0.0$ relies solely on unsupervised estimates given the set of anchors.

Semi-supervised comparison. Next, we compare our method to two other semi-supervised baselines, using both HMMs and FHMMs: EM and self-training. EM requires decoding and counting in multiple passes over the full unlabeled corpus. We initialized the parameters with the supervised estimates, and selected the iteration with the best accuracy on the development set.[5] The self-training baseline uses the supervised system to tag the unlabeled data, and then retrains on all the data.

Results are shown in Table 1. We observe that, for small amounts of labeled data (150 tweets), our method outperforms all the supervised and semi-supervised baselines, yielding accuracies 6.1 points above the best semi-supervised baseline for a simple HMM, and 1.9 points above for the FHMM. With more labeled data (1,000 instances), our method outperforms all the baselines for the HMM, but not with the more sophisticated FHMM, in which our accuracies are 0.3 points below the self-training method.[6] These results suggest that our method is more effective when the amount of labeled data is small.

[5]The FHMM with EM did not perform better than the supervised baseline, so we consider the initial value as the best accuracy under this model.

[6]According to a word-level paired Kolmogorov-Smirnov test, for the FHMM with 1,000 tweets, the self-training method outperforms the other methods with statistical significance at $p < 0.01$; and for the FHMM with 150 tweets the anchor-based and self-training methods outperform the other baselines with the same p-value. Our best HMM outperforms the other baselines at a significance level of $p < 0.01$ for 150 and 1000 sequences.

	150	1000
MEMM (same+clusters)	89.57	**93.36**
MEMM (same+clusters+posteriors)	**91.14**	93.18
MEMM (all+clusters)	91.55	**94.17**
MEMM (all+clusters+posteriors)	**92.06**	94.11

Table 2: Tagging accuracy for the MEMM POS tagger of Owoputi et al. (2013) with additional features from our model's posteriors.

Stacking features. We also evaluated a stacked architecture in which we use our model's predictions as an additional feature to improve the state-of-the-art Twitter POS tagger of Owoputi et al. (2013). This system is based on a semi-supervised discriminative model with Brown cluster features (Brown et al., 1992). We provide results using their full set of features (*all*), and using the same set of features in our anchor model (*same*). We compare tagging accuracy on a model with these features plus Brown clusters (+*clusters*) against a model that also incorporates the posteriors from the anchor method as an additional feature in the MEMM (+*clusters*+*posteriors*). The results in Table 2 show that using our model's posteriors are beneficial in the small labeled case, but not if the entire labeled data is used.

Runtime comparison. The training time of anchor FHMM is 3.8h (hours), for self-training HMM 10.3h, for EM HMM 14.9h and for Twitter MEMM (all+clusters) 42h. As such, the anchor method is much more efficient than all the baselines because it requires a single pass over the corpus to collect the moment statistics, followed by the QPs, without the need to decode the unlabeled data. EM and the Brown clustering method (the latter used to extract features for the Twitter MEMM) require several passes over the data; and the self-training method involves decoding the full unlabeled corpus, which is expensive when the corpus is large. Our analysis adds to previous evidence that spectral methods are more scalable than learning algorithms that require inference (Parikh et al., 2012; Cohen et al., 2013).

6.2 Malagasy POS Tagging

For the Malagasy experiment, we used the small labeled dataset from Garrette et al. (2013), which consists of 176 sentences and 4,230 tokens. We also make use of their tag dictionaries with 2,773 types

Models	Accuracies
supervised FHMM	90.5
EM FHMM	90.5
self-training FHMM	88.7
anchors FHMM (token), λ=1.0	89.4
anchors FHMM (type+token), λ=1.0	**90.9**

Table 3: Tagging accuracies for the Malagasy dataset.

and 23 tags, and their unlabeled data (43.6K sequences, 777K tokens). We converted all the original POS tags to universal tags using the mapping proposed in Garrette et al. (2013).

Table 3 compares our method with semi-supervised EM and self-training, for the FHMM.We tested two supervision settings: token only, and type+token annotations, analogous to Garrette et al. (2013). The anchor method outperformed the baselines when both type and token annotations were used to build the set of anchor words.[7]

7 Conclusion

We proposed an efficient semi-supervised sequence labeling method using a generative log-linear model. We use contextual information from a set of *anchor* observations to disambiguate state, and build a weakly supervised method from this set. Our method outperforms other supervised and semi-supervised methods, with small supervision in POS-tagging for Malagasy, a scarcely annotated language, and for Twitter. Our anchor method is most competitive for learning with large amounts of unlabeled data, under weak supervision, while training an order of magnitude faster than any of the baselines.

Acknowledgments

Support for this research was provided by Fundação para a Ciência e Tecnologia (FCT) through the CMU Portugal Program under grant SFRH/BD/52015/2012. This work has also been partially supported by the European Union under H2020 project SUMMA, grant 688139, and by FCT, through contracts UID/EEA/50008/2013, through the LearnBig project (PTDC/EEI-SII/7092/2014), and the GoLocal project (grant CMUPERI/TIC/0046/2014).

[7] Note that the accuracies are not directly comparable to Garrette et al. (2013), who use a different tag set. However, our supervised baseline trained on those tags is already superior to the best semi-supervised system in Garrette et al. (2013), as we get 86.9% against the 81.2% reported in Garrette et al. (2013) using their tagset.

References

Sanjeev Arora, Rong Ge, Yoni Halpern, David Mimno, David Sontag Ankur Moitra, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proc. of International Conference of Machine Learning*.

Raphaël Bailly, Xavier Carreras, Franco M. Luque, and Ariadna Quattoni. 2013. Unsupervised spectral learning of WCFG as low-rank matrix completion. In *Proc. of Empirical Methods in Natural Language Processing*, pages 624–635.

Borja Balle and Mehryar Mohri. 2012. Spectral learning of general weighted automata via constrained matrix completion. In *Advances in Neural Information Processing Systems*, pages 2168–2176.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: Conference of the North American Association of Computational Linguistics*.

Peter F. Brown, Peter V. de Souza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Arun T. Chaganty and Percy Liang. 2014. Estimating latent-variable graphical models using moments and likelihoods. In *Proc. of International Conference on Machine Learning*.

Shay B. Cohen and Michael Collins. 2014. A provably correct learning algorithm for latent-variable PCFGs. In *Proc. of Association for Computational Linguistics*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proc. of North American Association of Computational Linguistics*.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

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

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of POS-taggers

for low-resource languages. In *Proc. of Association for Computational Linguistics*.

Gimpel, Schneider, O'Connor, Das, Mills, Eisenstein, Heilman, Yogatama, Flanigan, and Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proc. of Association of Computational Linguistics*.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2012. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes 25*, pages 259–284.

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

Dong Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proc. of Association of Computational Linguistics System Demonstrations*, pages 25–30.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Thang Nguyen, Jordan Boyd-Graber, Jeff Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *Proc. of North American Association for Computational Linguistics*.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory search and topic summarization for Twitter. In *Proc. of AAAI Conference on Weblogs and Social Media*.

Dominique Osborne, Shashi Narayan, and Shay B. Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the Association of Computational Linguistics*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of North American Association for Computational Linguistics*.

Ankur P. Parikh, Lee Song, Mariya Ishteva, Gabi Teodoru, and Eric P. Xing. 2012. A spectral algorithm for latent junction trees. In *Proc. of Uncertainty in Artificial Intelligence*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of International Conference on Language Resources and Evaluation (LREC)*.

Ariadna Quattoni, Borja Balle, Xavier Carreras, and Amir Globerson. 2014. Spectral regularization for max-margin sequence tagging. In *Proc. of International Conference of Machine Learning*, pages 1710–1718.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of Association for Computational Linguistics*, pages 354–362.

Karl Stratos, Alexander M. Rush, Shay B. Cohen, and Michael Collins. 2013. Spectral learning of refinement hmms. In *Proc. of Computational Natural Language Learning*.

Karl Stratos, Michael Collins, and Daniel J. Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*, 4:245–257.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(2):1–305.

Roni Rosenfeld Xiaojin Zhu, Stanley F. Chen. 1999. Linguistic features for whole sentence maximum entropy language models. In *European Conference on Speech Communication and Technology*.

Alexander J. Smola Yasemin Altun. 2006. Unifying divergence minimization and statistical inference via convex duality. In *Proc. of Conference on Learning Theory*.

Tianyi Zhou, Jeff A. Bilmes, and Carlos Guestrin. 2014. Divide-and-conquer learning by anchoring a conical hull. In *Advances in Neural Information Processing Systems*.

Learning from Explicit and Implicit Supervision Jointly For Algebra Word Problems

Shyam Upadhyay[1] **Ming-Wei Chang**[2] **Kai-Wei Chang**[3] **Wen-tau Yih**[2]
[1]University of Illinois at Urbana-Champaign, Urbana, IL, USA
[2]Microsoft Research, Redmond, WA, USA
[3]University of Virginia, Charlottesville, VA, USA

Abstract

Automatically solving algebra word problems has raised considerable interest recently. Existing state-of-the-art approaches mainly rely on learning from human annotated equations. In this paper, we demonstrate that it is possible to efficiently mine algebra problems and their numerical solutions with little to no manual effort. To leverage the mined dataset, we propose a novel structured-output learning algorithm that aims to learn from both explicit (e.g., equations) and implicit (e.g., solutions) supervision signals *jointly*. Enabled by this new algorithm, our model gains 4.6% absolute improvement in accuracy on the ALG-514 benchmark compared to the one without using implicit supervision. The final model also outperforms the current state-of-the-art approach by 3%.

1 Introduction

Algebra word problems express mathematical relationships via narratives set in a real-world scenario, such as the one below:

> Maria is now four times as old as Kate. Four years ago, Maria was six times as old as Kate. Find their ages now.

The desired output is an *equation system* which expresses the mathematical relationship symbolically: $m = 4 \times n$ and $m - 4 = 6 \times (n - 4)$ where m and n represent the age of Maria and Kate, respectively. The *solution* (i.e., $m = 40, n = 10$) can be found by a *mathematical engine* given the equation systems. Building efficient automatic algebra word problem solvers have clear values for online education scenarios. The challenge itself also provides a good test bed for evaluating an intelligent agent that understands natural languages, a direction advocated by artificial intelligence researchers (Clark and Etzioni, 2016).

One key challenge of solving algebra word problems is the lack of fully annotated data (i.e., the annotated equation system associated with each problem). In contrast to annotating problems with binary or categorical labels, manually solving algebra word problems to provide correct equations is time consuming. As a result, existing benchmark datasets are small, limiting the performance of supervised learning approaches. However, thousands of algebra word problems have been posted and discussed in online forums, where the solutions can be easily mined, despite the fact that some of them could be incorrect. It is thus interesting to ask whether a better algebra problem solver can be learned by leveraging these *noisy* and *implicit* supervision signals, namely the solutions.

In this work, we address the technical difficulty of leveraging implicit supervision in learning an algebra word problem solver. We argue that the effective strategy is to learn from both explicit and implicit supervision signals *jointly*. In particular, we design a novel online learning algorithm based on structured-output Perceptron. By taking both kinds of training signals together as input, the algorithm iteratively improves the model, while at the same time it uses the intermediate model to find candidate equation systems for problems with only numerical solutions.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 297–306,
Austin, Texas, November 1-5, 2016.

Our contributions are summarized as follows.

- We propose a novel learning algorithm (Section 3 and 4) that jointly learns from both explicit and implicit supervision. Under different settings, the proposed algorithm outperforms the existing supervised and weakly supervised algorithms (Section 6) for algebra word problems.
- We mine the problem-solution pairs for algebra word problems from an online forum and show that we can effectively obtain the implicit supervision with little to no manual effort (Section 5).[1]
- By leveraging both implicit and explicit supervision signals, our final solver outperforms the state-of-the-art system by 3% on ALG-514, a popular benchmark data set proposed by (Kushman et al., 2014).

2 Related Work

Automatically solving mathematical reasoning problems expressed in natural language has been a long-studied problem (Bobrow, 1964; Newell et al., 1959; Mukherjee and Garain, 2008). Recently, Kushman et al. (2014) created a template-base search procedure to map word problems into equations. Then, several following papers studied different aspects of the task: Hosseini et al. (2014) focused on improving the generalization ability of the solvers by leveraging extra annotations; Roy and Roth (2015) focused on how to solve arithmetic problems without using any pre-defined template. In (Shi et al., 2015), the authors focused on number word problems and proposed a system that is created using semi-automatically generated rules. In Zhou et al. (2015), the authors simplified the inference procedure and pushed the state-of-the-art benchmark accuracy. The idea of learning from implicit supervision is discussed in (Kushman et al., 2014; Zhou et al., 2015; Koncel-Kedziorski et al., 2015), where the authors train the algebra solvers using only the solutions with little or no annoated equation systems. We discuss this in detail in Section 4.

[1]The new resource and the dataset we used for training is available soon on `https://aka.ms/dataimplicit` and `https://aka.ms/datadraw`

Solving automatic algebra word problems can be viewed as a semantic parsing task. In the semantic parsing community, the technique of learning from implicit supervision signals has been applied (under the name *response-driven learning* (Clarke et al., 2010)) to knowledge base question answering tasks such as Geoquery (Zelle and Mooney, 1996) and WebQuestions (Berant et al., 2013) or mapping instructions to actions (Artzi and Zettlemoyer, 2013). In these tasks, researchers have shown that it is possible to train a semantic parser only from question-answer pairs, such as "*What is the largest state bordering Texas?*" and "*New Mexico*" (Clarke et al., 2010; Liang et al., 2013; Yih et al., 2015).

One key reason that such implicit supervision is effective is because the correct semantic parses of the questions can often be found using the answers and the knowledge base alone, with the help of heuristics developed for the specific domain. For instance, when the question is relatively simple and does not have complex compositional structure, paths in the knowledge graph that connect the answers and the entities in the narrative can be interpreted as legitimate semantic parses. However, as we will show in our experiments, learning from implicit supervision alone is not a viable strategy for algebra word problems. Compared to the knowledge base question answering problems, one key difference is that a large number (potentially infinitely many) of different equation systems could end up having the same solutions. Without a database or special rules for combining variables and coefficients, the number of candidate equation systems cannot be trimmed effectively, given only the solutions.

From the algorithmic point of view, our proposed learning framework is related to several lines of work. Similar efforts have been made to develop latent structured prediction models (Yu and Joachims, 2009; Chang et al., 2013; Zettlemoyer and Collins, 2007) to find latent semantic structures which best explain the answer given the question. Our algorithm is also influenced by the discriminative reranking algorithms (Collins, 2000; Ge and Mooney, 2006; Charniak and Johnson, 2005) and models for learning from intractable supervision (Steinhardt and Liang, 2015).

Recently, Huang et al. (2016) collected a large

number of noisily annotated word problems from online forums. While they collected a large-scale dataset, unlike our work, they did not demonstrate how to utilize the newly crawled dataset to improve existing systems. It will be interesting to see if our proposed algorithm can make further improvements using their newly collected dataset.[2]

3 Problem Definition

Table 1 lists all the symbols representing the components in the process. The input algebra word problem is denoted by $\mathbf{x}$, and the output $\mathbf{y} = (T, A)$ is called a *derivation*, which consists of an *equation system template* T and an *alignment* A. A template T is a family of equation systems parameterized by a set of coefficients $\mathcal{C}(T) = \{c_i\}_{i=1}^{k}$, where each coefficient c_i aligns to a textual number (e.g., *four*) in a word problem. Let $\mathcal{Q}(x)$ be all the *textual numbers* in the problem $\mathbf{x}$, and $\mathcal{C}(T)$ be the coefficients to be determined in the template T. An alignment is a set of tuples $A = \{(q, c) \mid q \in \mathcal{Q}(x), c \in \mathcal{C}(T) \cup \{\epsilon\}\}$, where the tuple (q, ϵ) indicates that the number q is not relevant to the final equation system. By specifying the value of each coefficient, it identifies an equation system belonging to the family represented by template T. Together, T and A generate a complete equation system, and the solution $\mathbf{z}$ can be derived by the mathematical engine E.

Following (Kushman et al., 2014; Zhou et al., 2015), our strategy of mapping a word problem to an equation system is to first choose a template that consists of variables and coefficients, and then align each coefficient to a textual number mentioned in the problem. We formulate the mapping between an algebra word problem and an equation system as a structured learning problem. The output space is the set of all possible derivations using templates that are observed in the training data. Our model maps $\mathbf{x}$ to $\mathbf{y} = (T, A)$ by a linear scoring function $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{w}$ is the model parameters and Φ is the feature functions. At test time, our model scores all the derivation candidates and picks the best one according to the model score. We often refer to $\mathbf{y}$ as a semantic parse, as it represents the semantics of the algebra word problem.

[2]The dataset has not been made public at the time of publication.

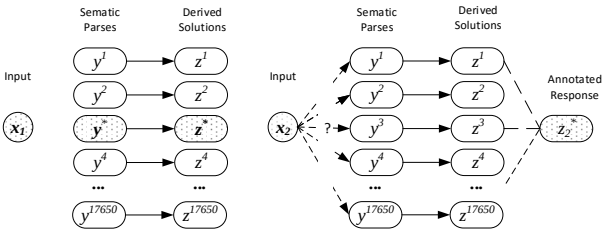


Figure 1: **Left:** Explicit supervision signals. Note that the solution $\mathbf{z}$ can be derived by the semantic parses $\mathbf{y}$. **Right:** Implicit supervision signals. In this case, we only have the annotated response $\mathbf{z}_2^*$. It is difficult to use $\mathbf{z}_2^*$ to find the correct derivation, as multiple derivations may lead to the same solution. Therefore, the learning algorithm has to explore the output space to guide the model in order to match the annotated response.

Properties of Implicit Supervision Signals We discuss some key properties of the implicit supervision signal to explain several design choices of our algorithm. Figure 1 illustrates the main differences between implicit and explicit supervision signals.

Algorithms that learn from implicit supervision signals face the following challenges. First, the learning system usually does not model directly the correlations between the input $\mathbf{x}$ and the solution $\mathbf{z}$. Instead, the mapping is handled by an external procedure such as a mathematical engine. Therefore, $E(\mathbf{y})$ is effectively a one-directional function. As a result, finding semantic parses (derivations) from responses (solutions) $E^{-1}(\mathbf{z})$ can sometimes be very slow or even intractable. Second, in many cases, even if we could find a semantic parse from responses, multiple combinations of templates and alignments could end up with the same solution set (e.g., the solutions of equations $2 + x = 4$ and $2 \times x = 4$ are the same). Therefore, the implicit supervision signals may be incomplete and noisy, and using the solutions alone to guide the training procedure might not be sufficient. Finally, since we need to have a complete derivation before we can observe the response of the mathematical engine E, we cannot design efficient inference methods such as dynamic programming algorithms based on partial feedback. As a result, we have to perform exploration during learning to search for fully constructed semantic parses that can generate the correct solution.

Term	Symbol	Example
Word Problem	$\mathbf{x}$	*Maria is now four times as old as Kate. Four years ago, Maria was six times as old as Kate. Find their ages now.*
Derivation (Semantic Parse)	$\mathbf{y} = (T, A)$	$(\{m - a \times n = -1 \times a \times b + b, m - c \times n = 0\}, A)$
Solution	$\mathbf{z}$	$n = 10, m = 40$
Mathematical Engine	$E : \mathbf{y} \rightarrow \mathbf{z}$	After determining the coefficients, the equation system is $\{m = 4 \times n, m - 4 = 6 \times (n - 4)\}$. The solution is thus $n = 10$, $m = 40$.
Variables	v	m, n
Textual Number[3]	$\mathcal{Q}(x)$	{four, Four, six}
Equation System Template	T	$\{m - a \times n = -1 \times a \times b + b, m - c \times n = 0\}$
Coefficients	$\mathcal{C}(T)$	a, b, c
Alignment	A	six $\rightarrow a$, Four $\rightarrow b$, four $\rightarrow c$

Table 1: Notation used in this paper to formally describe the problem of mapping algebra word problems to equations.

4 Learning from Mixed Supervision

We assume that we have two sets: $D_e = \{(\mathbf{x}_e, \mathbf{y}_e)\}$ and $D_m = \{(\mathbf{x}_m, \mathbf{z}_m)\}$. D_e contains the fully annotated equation system $\mathbf{y}_e$ for each algebra word problem $\mathbf{x}_e$, whereas in D_m, we have access to the numerical solution $\mathbf{z}_m$ to each problem, but not the equation system ($\mathbf{y}_m = \emptyset$). We refer to D_e as the *explicit set* and D_m as the *implicit set*. For the sake of simplicity, we explain our approach by modifying the training procedure of the structured Perceptron algorithm (Collins, 2002).[4]

As discussed in Section 3, the key challenge of learning from implicit supervision is that the mapping $E(\mathbf{y})$ is one-directional. Therefore, the correct equation system cannot be easily derived from the numerical solution. Intuitively, for data with only implicit supervision, we can explore the structure space Y and find the best possible derivation $\tilde{\mathbf{y}} \in Y$ according to the current model. If $E(\tilde{\mathbf{y}})$ matches $\mathbf{z}$, then we can update the model based on $\tilde{\mathbf{y}}$. Following this intuition, we propose *MixedSP* (Algorithm 1).

For each example, we use an approximate search algorithm to collect top scoring candidate structures. The algorithm first ranks the top-K templates according to the model score, and forms a candidate set by expanding all possible derivations that use the K templates (Line 3). The final candidate set is $\Omega = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^K\} \subset Y$.

When the explicit supervision is available (i.e., $(\mathbf{x}_i, \mathbf{y}_i) \in D_e$), our algorithm follows the standard structured prediction update procedure. We find the best scoring structure $\hat{\mathbf{y}}$ in Ω and then update the model using the difference of the feature vectors between the gold output structure $\mathbf{y}_i$ and the best scoring structure $\hat{\mathbf{y}}$ (Line 6).

When only implicit supervision is available (i.e., $(\mathbf{x}_i, \mathbf{z}_i) \in D_m$), our algorithm uses the current model to conduct a guided exploration, which iteratively finds structures that best explain the implicit supervision, and use the explanatory structure for making updates. As mentioned in Section 3, we have to explore and examine each structure in the candidate set Ω. This is due the fact that partial structure cannot be used for finding the right response, as getting response $E(\mathbf{y})$ requires complete derivations. In Line 9, we want to find the derivations $\mathbf{y}$ where its solution $E(\mathbf{y})$ matches the implicit supervision $\mathbf{z}_i$. More specifically,

$$\tilde{\mathbf{y}} = \arg\min_{\mathbf{y} \in \Omega} \Delta(E(\mathbf{y}), \mathbf{z}_i), \quad (1)$$

where Δ is a loss function to estimate the disagreement between $E(\mathbf{y})$ and $\mathbf{z}_i$. In our experiments, we simply set $\Delta(E(\mathbf{y}), \mathbf{z}_i)$ to be 0 if the solution partially matches, and 1 otherwise.[5] If more than one derivation achieves the minimal value of $\Delta(E(\mathbf{y}), \mathbf{z}_i)$, we break ties by choosing the derivation with higher score $\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$. This tie-

[4] Our approach can be easily extended to other structured learning algorithms such as Structured SVM (Taskar et al., 2004; Tsochantaridis et al., 2004).

[5] The mined solutions are often incomplete for some variables (e.g. solution `y=6` but no value for `x` could be mined). We allow partial matches so that the model can learn from the incomplete implicit signals as well.

Algorithm 1 Structured Perceptron with Mixed Supervision. (**MixedSP**)

Input: D_e, D_m, $L = |D_e| + |D_m|$, T, K, $\gamma \in [0, 1)$

1: **for** $t = 1 \dots N$ **do** ▷ training epochs
2: **for** $i = 1 \dots L$ **do**
3: $\Omega \leftarrow$ find top-K structures $\{\mathbf{y}\}$ approximately
4: **if** $\mathbf{y}_i \neq \emptyset$ **then** ▷ explicit supervision
5: $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y} \in \Omega} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$
6: $\mathbf{w} \leftarrow \mathbf{w} + \eta \left(\phi(\mathbf{x}, \mathbf{y}_i) - \phi(\mathbf{x}, \hat{\mathbf{y}})\right)$
7: **else if** $t \geq \gamma N$ **then** ▷ implicit supervision
8: $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y} \in \Omega} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$
9: Pick $\tilde{\mathbf{y}}$ from Ω by Eq. (1). ▷ exploration
10: $\mathbf{w} \leftarrow \mathbf{w} + \eta \left(\phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \hat{\mathbf{y}})\right)$
11: Return the average of $\mathbf{w}$

breaking strategy is important – in practice, several derivations may lead to the gold numerical solution; however, only few of them are correct. The tie-breaking strategy relies on the current model and the structured features $\phi(\mathbf{x}_i, \mathbf{y})$ to filter out incorrect derivations during training. Finally, the model is updated using $\tilde{\mathbf{y}}$ in Line 10.

Similar to curriculum learning (Bengio et al., 2009), it is important to control when the algorithm starts exploring the output space using weak supervision. Exploring too early may mislead the model, as the structured feature weights $\mathbf{w}$ may not be able to help filter out incorrect derivations, while exploring too late may lead to under-utilization of the implicit supervision. We use the parameter γ to control when the model starts to learn from implicit supervision signals. The parameter γ denotes the fraction of the training time that the model uses purely explicit supervision.

Key Properties of Our Algorithm The idea of using solutions to train algebra word problem solvers has been discussed in (Kushman et al., 2014) and (Zhou et al., 2015). However, their implicit supervision signals are created from clean, fully supervised data, and the experiments use little to no explicit supervision examples.[6] While their algorithms are interesting, the experimental setting is somewhat unrealistic as the implicit signals are simulated.

[6]Prior work (Kushman et al., 2014) has used only 5 explicit supervision examples when training with solutions.

On the other hand, the goal of our algorithm is to significantly improve a strong solver with a large quantity of unlabeled data. Moreover, our implicit supervision signals are noisier given that we crawled the data automatically, and the clean labeled equation systems are not available to us. As a result, we have made several design choices to address issues of learning from noisy implicit supervision signals in practice.

First, the algorithm is designed to perform updates *conservatively*. Indeed, in Line 10, the algorithm will not perform an update if the model could not find any parses matching the implicit signals in Line 9. That is, if $\Delta(E(\mathbf{y}), \mathbf{z}_i) = 1$ for all $\mathbf{y} \in \Omega$, $\tilde{\mathbf{y}} = \hat{\mathbf{y}}$ due to the tie-breaking mechanism. This ensures that the algorithm drives the learning using only those structures which lead to the correct solution, avoiding undesirable effects of noise.

Second, the algorithm does *not* use implicit supervision signals in the early stage of model training. Learning only on clean and explicit supervision helps derive a better intermediate model, which later allows exploring the output space more efficiently using the implicit supervision signals.

Existing semantic parsing algorithms typically use either implicit or explicit supervision signals *exclusively* (Zettlemoyer and Collins, 2007; Berant et al., 2013; Artzi and Zettlemoyer, 2013). In contrast, *MixedSP* makes use of both explicit and implicit supervised examples mixed at the training time.

5 Mining Implicit Supervision Signals

In this section, we describe the process of collecting SOL-2K, a data set containing question-solution pairs of algebra word problems from a Web forum[7], where students and tutors interact to solve math problems.

A word problem posted on the forum is often accompanied by a detailed explanation provided by tutors, which includes a list of the relevant equations. However, these posted equations are not suitable for direct use as labeled data, as they are often imprecise or incomplete. For instance, tutors often omit many simplification steps when writing the equations. A commonly observed example is that `(5-3)x+2y` would be directly written as `2x+2y`. Despite being

[7]`http://www.algebra.com`

mathematically equivalent, learning from the latter equation is not desirable as the model may learn that 5 and 3 appearing the text are irrelevant. An extreme case of this is when tutors directly post the solution (such as `x=2` and `y=5`), without writing any equations. Another observation is that tutors often write two-variable equation systems with only one variable. For example, instead of writing `x+y=10, x-y=2`, many tutors pre-compute `x=10-y` using the first equation and substitute it in the second one, which results in `10-y-y=2`. It is also possible that the tutor wrote the incorrect equation system, but while explaining the steps, made corrections to get the right answer. These practical issues make it difficult to use the crawled equations for explicit supervision directly.

On the other hand, it is relatively easy to obtain question-solution pairs with simple heuristics. We use a simple strategy to generate the solution from the extracted equations. We greedily select equations in a top-down manner, declaring success as soon as we find an equation system that can be solved by a mathematical engine (we used SymPy (Sympy Development Team, 2016)). Equations that cause an exception in the solver (due to improper extraction) are rejected. Note that the solution thus found may be incorrect (making the mined supervision noisy), as the equation system used by the solver may contain an incorrect equation. To ensure the quality of the mined supervision, we use several simple rules to further filter the problems. For example, we remove questions that have more than 15 numbers. We found that usually such questions were not a single word problem, but instead concatenations of several problems.

Note that our approach relies only on a few rules and a mathematical engine to generate (noisy) implicit supervision from crawled problems, with no human involvement. Once the solutions are generated, we discarded the equation systems used to obtain them. Using this procedure, we collected 2,039 question-solution pairs. For example, the solution to the following mined problem was "6" (The correct solutions are 6 and 12.):

> Roz is twice as old as Grace. In 5 years the sum of their ages will be 28. How old are they now?

Settings	Explicit sets		Implicit sets
	(D_e)		(D_m)
Dataset	ALG-514	DRAW-1K	SOL-2K
# temp.	24	224	Unknown
# prob.	514	1,000	2,039
Vocab.	1.83k	2.2k	6.8k

Table 2: The statistics of the data sets.

6 Experiments

In this section, we demonstrate the effectiveness of the proposed approach and empirically verify the design choices of the algorithm. We show that our joint learning approach leverages mined implicit supervision effectively, improving system performance without using additional manual annotations (Section 6.1). We also compare our approach to existing methods under different supervision settings (Section 6.2).

Experimental Settings Table 2 shows the statistics of the datasets. The ALG-514 dataset (Kushman et al., 2014) consists of 514 algebra word problems, ranging over a variety of narrative scenarios (object counting, simple interest, etc.). Although it is a popular benchmark for evaluating algebra word solvers, ALG-514 has only 24 templates. To test the generality of different approaches, we thus conduct experiments on a newly released data set, DRAW-1K[8] (Upadhyay and Chang, 2016), which covers more than 200 templates and contains 1,000 algebra word problems. The data is split into training, development, and test sets, with 600/200/200 examples, respectively.

The SOL-2K dataset contains the word problem-solution pairs we mined from online forum (see Section 5). Unlike ALG-514 and DRAW-1K, there are no annotated equation systems in this dataset, and only the solutions are available. Also, no preprocessing or cleaning is performed, so the problem descriptions might contain some irrelevant phrases such as "please help me". Since all the datasets are generated from online forums, we carefully examined and removed problems from SOL-2K that are identical to problems in ALG-514 and DRAW-1K, to ensure fairness. We set the number of iterations

[8] `https://aka.ms/datadraw`

to 15 and the learning rate η to be 1.

For all experiments, we report *solution accuracy* (whether the solution was correct). Following Kushman et al. (2014), we ignore the ordering of answers when calculating the solution accuracy. We report the 5-fold cross validation accuracy on ALG-514 in order to have a fair comparison with previous work. For DRAW-1K, we report the results on the test set. In all the experiments, we only use the templates that appear in the corresponding explicit supervision.

Following (Zhou et al., 2015), we do not model the alignments between noun phrases and variables. We use a similar set of features introduced in (Zhou et al., 2015), except that our solver does not use rich NLP features from dependency parsing or coreference-resolution systems. We follow (Kushman et al., 2014) and set the beam-size K to 10, unless stated otherwise.

6.1 Joint Learning from Mixed Supervision

Supervision Protocols We compare the following training protocols:

- *Explicit* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\}$): the standard setting, where fully annotated examples are used to train the model (we use the structured Perceptron algorithm as our training algorithm here).

- *Implicit* ($D = \{(\mathbf{x}_m, \mathbf{z}_m))\}$): the model is trained on SOL-2K dataset only (i.e., only implicit supervision). This setting is similar to the one in (Liang et al., 2013; Clarke et al., 2010).

- *Pseudo* ($D = \{(\mathbf{x}_m, \tilde{Z}^{-1}(\mathbf{z}_m, \mathbf{x}_m))\}$): where we use $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ to denote a pseudo derivation whose solutions match the mined solutions. Similar to the approach in (Yih et al., 2015) for question answering, here we attempts to recover (possibly incorrect) explicit supervision from the implicit supervision by finding parses whose solution matches the mined solution. For each word problem, we generated a pseudo derivation $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ by finding the equation systems whose solutions that match the mined solutions. We conduct a brute force search to find $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ by enumerating all possible derivations. Note that this process can be very slow for datasets like DRAW-1K because the brute-force search needs to examine more than 200 templates for each word problem. Ties are broken by random.

- *E+P* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\} \cup \{(\mathbf{x}_m, \tilde{Z}^{-1}(\mathbf{z}_m, \mathbf{x}_m))\}$): a baseline approach that jointly learns by combining the dataset generated by *Pseudo* with the *Explicit* supervision.

- *MixedSP* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\} \cup \{(\mathbf{x}_m, \mathbf{z}_m))\}$): the setting used by our proposed algorithm. The algorithm trained the word problem solver using both explicit and implicit supervision jointly. We set the parameter γ to 0.5 unless otherwise stated. In other words, the first half of the training iterations use only explicit supervision.

Note that *Explicit*, *E+P*, and *MixedSP* use the same amount of labeled equations, although *E+P* and *MixedSP* use additional implicit supervised resources.

Results Table 3 lists the main results. With implicit supervision from mined question-solution pairs, *MixedSP* outperforms *Explicit* by around 4.5% on both datasets. This verifies the claim that the joint learning approach can benefit from the noisy implicit supervision. Note that with the same amount of supervision signals, *E+P* performs poorly and even under-performs *Explicit*. The reason is that the derived derivations in SOL-2K can be noisy. Indeed, we found that about 70% of the problems in the implicit set have more than one template that can produce a derivation which matches the mined solutions. Therefore, the pseudo derivation selected by the system might be wrong, even if they generate the correct answers. As a result, *E+P* can commit to the possibly incorrect pseudo derivations before training, and suffer from error propagation. In contrast, *MixedSP* does not commit to a derivation and allows the model to choose the one best explaining the implicit signals as training progresses.

As expected, using only the implicit set D_m performs poorly. The reason is that in both *Implicit* and *Pseudo* settings, the algorithm needs to select one from many derivations that match the labeled solutions, and use the selected derivation to update the model. When there are no explicit supervision

Dataset	D_e	D_m		D_e and D_m	
	Expl.	Pseudo	Impl.	E+P	MixedSP
ALG-514	78.4	54.1	63.7	73.3	**83.0**
DRAW-1K	55.0	33.5	39.0	48.5	**59.5**

Table 3: The solution accuracies of different protocols on ALG-514 and DRAW-1K.

signals, the model can use incorrect derivations to update the model. As a result, models on both *Implicit* and *Pseudo* settings perform significantly worse than the *Explicit* baseline in both datasets, even if the size of SOL-2K is larger than the fully supervised data.

6.2 Comparisons to Previous Work

We now compare to previous approaches for solving algebra word problems, both in fully supervised and weakly supervised settings.

Comparisons of Overall Systems We first compare our systems to the systems that use the same level of explicit supervision (fully labeled examples). The comparison between our system and existing systems are in Fig 2a and 2b. Compared to previous systems that were trained only on explicit signals, our *Explicit* baseline is quite competitive. On ALG-514, the accuracy of our baseline system is 78.4%, which is 1.3% lower than the best reported accuracy achieved by the system ZDC15 (Zhou et al., 2015). We suspect that this is due to the richer feature set used by ZDC15, which includes features based on POS tags, coreference and dependency parses, whereas our system only uses features based on POS tags. Our system is also the best system on DRAW-1K, and performs much better than the system KAZB14 (Kushman et al., 2014). Note that we could not run the system ZDC15 on DRAW-1K because it can only handle limited types of equation systems. Although the *Explicit* baseline is strong, the *MixedSP* algorithm is still able to improve the solver significantly through noisy implicit supervision signals without using manual annotation of equation systems.

Comparisons of Weakly Supervised Algorithms In the above comparisons, *MixedSP* benefits from the mined implicit supervision as well as using Algorithm 1. Since there are several practical limitations for us to run previously proposed weakly supervised algorithms in our settings, in the following, we perform a direct comparison between *MixedSP* and existing algorithms in their corresponding settings. Note that the implicit supervision in weak supervision settings proposed in earlier work is noise-free, as it was simulated by hiding equation systems of a manually annotated dataset.

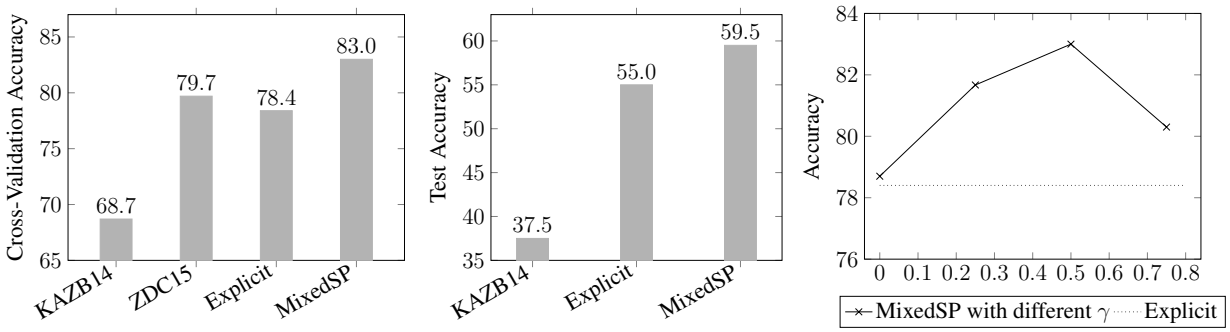
Zhou et al. (2015) proposed a weak supervision setting where the system was provided with the set of all templates, as well as the solutions of all problems during training. Under this setting, they reported 72.3% accuracy on ALG-514. Note that such high accuracy can be achieved mainly because that the complete and correct templates were supplied.

In this setting, running the *MixedSP* algorithm is equivalent to using the *Implicit* setting with clean implicit supervision signals. Surprisingly, *MixedSP* can obtain 74.3% accuracy, surpassing the weakly supervised model in (Zhou et al., 2015) on ALG-514. Compared to the results in Table 3, note that when using noisy implicit signals, it cannot obtain the same level of results, even though we had more training problems (2,000 mined problems instead of 514 problems). This shows that working with real, noisy weak supervision is much more challenging than working on simulated, noise-free, weak supervision.

Kushman et al. (2014) proposed another weak supervision setting (5EQ+ANS in the paper), in which explicit supervision is provided for only 5 problems in the training data. For the rest of problems, only their solutions are provided. The 5 problems are chosen such that their templates constitute the 5 most common templates in the dataset. This weak supervision setting is harder than that of (Zhou et al., 2015), as the solver only has the templates for 5 problems, instead of the templates for all problems. Under this setting, our *MixedSP* algorithm achieves 53.8%, which is better than 46.1% reported in (Kushman et al., 2014).

6.3 Analysis

In Figure 2c, we investigate the impact of tuning γ in *MixedSP* on the dataset ALG-514. Recall that γ controls the fraction of the training time that the model uses solely explicit supervision. At first glance, it may appear that we should utilize the im-



(a) (b) (c)

Figure 2: **(a)** Comparisons between our system to state-of-the-art systems on ALG-514. ZDC15 is the system proposed in (Zhou et al., 2015), and KAZB14 is the system proposed in (Kushman et al., 2014). **(b)** Comparisons between our system and other systems on DRAW-1K. Note that we are not able to run ZDC15 on DRAW-1K because it cannot handle some equation systems in the dataset. **(c)** Analysis of the impact of γ in MixedSP.

plicit supervision throughout training (set $\gamma = 0$). But setting γ to 0 hurts overall performance, suggesting in this setting that the algorithm uses a weak model to guide the exploration for using implicit supervision. On the other hand, by delaying exploration ($\gamma > 0.5$) for too long, the model could not fully utilize the implicit supervision. We observe similar trend on DRAW-1K as well. We found $\gamma = 0.5$ works well across the experiments.

We also analyze the impact of the parameter K, which controls the size of the candidate set Ω in *MixedSP*. Specifically, for DRAW-1K, when setting K to 5 and 10, the accuracy of *MixedSP* is at 59.5%. On setting K to 15, the accuracy of *MixedSP* improves to 61%. We suspect that enlarging K increases the chance to have good structures in the candidate set that can match the correct responses.

7 Conclusion

In this paper, we propose an algorithmic approach for training a word problem solver based on both explicit and implicit supervision signals. By extracting the question answer pairs from a Web-forum, we show that the algebra word problem solver can be improved significantly using our proposed technique, surpassing the current state-of-the-art.

Recent advances in deep learning techniques demonstrate that the error rate of machine learning models can decrease dramatically when large quantities of labeled data are presented (Krizhevsky et al., 2012). However, labeling natural language data has been shown to be expensive, and it has become one of the major bottleneck for advancing natural language understanding techniques (Clarke et al., 2010). We hope the proposed approach can shed light on how to leverage data on the web, and eventually improves other semantic parsing tasks such as knowledge base question answering and mapping natural instructions to actions.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *Proc. of TACL*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc. of ICML*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proc. of EMNLP*.

Daniel G. Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*.

K.-W. Chang, R. Samdani, and D. Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student-but how intelligent is it? Standardized tests as a measure of AI. *AI Magazine.*, 37(1).

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of CoNLL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

R. Ge and R. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proc. of ACL*.

Javad Mohammad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proc. of EMNLP*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? Large-scale dataset construction and evaluation. In *Proc. of ACL*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Proc. of TACL*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. of ACL*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artif. Intell. Rev.*, 29(2):93–122.

Allen Newell, John C Shaw, and Herbert A Simon. 1959. Report on a general problem-solving program. In *IFIP Congress*, pages 256–264.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proc. of EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proc. of EMNLP*.

J. Steinhardt and P. Liang. 2015. Learning with relaxed supervision. In *Proc. of NIPS*.

Sympy Development Team, 2016. *SymPy: Python library for symbolic mathematics*.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin markov networks. In *Proc. of NIPS*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *https://aka.ms/derivationpaper*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proc. of ACL*.

C. Yu and T. Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. of ICML*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Proc. of AAAI*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proc. of EMNLP*.

TweeTime: A Minimally Supervised Method for Recognizing and Normalizing Time Expressions in Twitter

Jeniya Tabassum, Alan Ritter and Wei Xu
Computer Science and Engineering
Ohio State University
{bintejafar.1,ritter.1492,xu.1265}@osu.edu

Abstract

We describe TweeTIME, a temporal tagger for recognizing and normalizing time expressions in Twitter. Most previous work in social media analysis has to rely on temporal resolvers that are designed for well-edited text, and therefore suffer from reduced performance due to domain mismatch. We present a minimally supervised method that learns from large quantities of unlabeled data and requires no hand-engineered rules or hand-annotated training corpora. TweeTIME achieves 0.68 F1 score on the end-to-end task of resolving date expressions, outperforming a broad range of state-of-the-art systems.[1]

1 Introduction

Temporal expressions are words or phrases that refer to dates, times or durations. Resolving time expressions is an important task in information extraction (IE) that enables downstream applications such as calendars or timelines of events (Derczynski and Gaizauskas, 2013; Do et al., 2012; Ritter et al., 2012; Ling and Weld, 2010), knowledge base population (Ji et al., 2011), information retrieval (Alonso et al., 2007), automatically scheduling meetings from email and more. Previous work in this area has applied rule-based systems (Mani and Wilson, 2000; Bethard, 2013b; Chambers, 2013) or supervised machine learning on small collections of hand-annotated news documents (Angeli et al., 2012; Lee et al., 2014).

[1]Our code and data are publicly available at https://github.com/jeniyat/TweeTime.

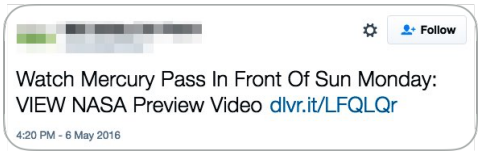


Figure 1: A tweet published on *Friday 5/6/2016* that contains the temporal expression *Monday* referring to the date of the event (*5/9/2016*), which a generic temporal tagger failed to resolve correctly.

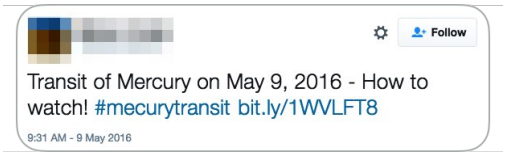


Figure 2: A tweet that contains a simple explicit time mention and an event (*Mercury, 5/9/2016*) that can be identified by an open-domain information extraction system.

Social media especially contains time-sensitive information and requires accurate temporal analysis, for example, for detecting real-time cybersecurity events (Ritter et al., 2015; Chang et al., 2016), disease outbreaks (Kanhabua et al., 2012) and extracting personal information (Schwartz et al., 2015). However, most work on social media simply uses generic temporal resolvers and therefore suffers from suboptimal performance. Recent work on temporal resolution focuses primarily on news articles and clinical texts (UzZaman et al., 2013; Bethard and Savova, 2016).

Resolving time expressions in social media is a non-trivial problem. Besides many spelling variations, time expressions are more likely to refer to future dates than in newswire. For the example in

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 307–318,
Austin, Texas, November 1-5, 2016.

Figure 1, we need to recognize that *Monday* refers to the upcoming Monday and not the previous one to resolve to its correct normalized date (*5/9/2016*). We also need to identify that the word *Sun* is not referring to a Sunday in this context.

In this paper, we present a new minimally supervised approach to temporal resolution that requires no in-domain annotation or hand-crafted rules, instead learning from large quantities of unlabeled text in conjunction with a database of known events. Our approach is capable of learning robust time expression models adapted to the informal style of text found on social media.

For popular events, some related tweets (e.g. Figure 2) may contain explicit or other simple time mentions that can be captured by a generic temporal tagger. An open-domain information extraction system (Ritter et al., 2012) can then identify events (e.g. [*Mercury*, *5/9/2016*]) by aggregating those tweets. To automatically generate temporally annotated data for training, we make the following novel *distant supervision assumption*:[2]

> Tweets posted near the time of a known event that mention central entities are likely to contain time expressions that refer to the date of the event.

Based on this assumption, tweets that contain the same named entity (e.g. Figure 1) are heuristically labeled as training data. Each tweet is associated with multiple overlapping labels that indicate the day of the week, day of the month, whether the event is in the past or future and other time properties of the event date in relation to the tweet's creation date. In order to learn a tagger that can recognize temporal expressions at the word-level, we present a multiple-instance learning approach to model sentence and word-level tags jointly and handle overlapping labels. Using heuristically labeled data and the temporal tags predicted by the multiple-instance learning model as input, we then train a log-linear model that normalizes time expressions to calendar dates.

Building on top of the multiple-instance learning model, we further improve performance using a missing data model that addresses the problem of errors introduced during the heuristic labeling process. Our best model achieves a 0.68 F1 score when resolving date mentions in Twitter. This is a 17% increase over SUTime (Chang and Manning, 2012), outperforming other state-of-the-art time expression resolvers HeidelTime (Strötgen and Gertz, 2013), TempEX (Mani and Wilson, 2000) and UWTime (Lee et al., 2014) as well. Our approach also produces a confidence score that allows us to trade recall for precision. To the best of our knowledge, TweeTIME is the first time resolver designed specifically for social media data.[3] This is also the first time that distant supervision is successfully applied for end-to-end temporal recognition and normalization. Previous distant supervision approaches (Angeli et al., 2012; Angeli and Uszkoreit, 2013) only address the normalization problem, assuming gold time mentions are available at test time.

2 System Overview

Our TweeTIME system consists of two major components as shown in Figure 3:

1. A **Temporal Recognizer** which identifies time expressions (e.g. *Monday*) in English text and outputs 5 different temporal types (described in Table 1) indicating timeline direction, month of year, date of month, day of week or no temporal information (NA). It is realized as a multiple-instance learning model, and in an enhanced version, as a missing data model.

2. A **Temporal Normalizer** that takes a tweet with its creation time and temporal expressions tagged by the above step as input, and outputs their normalized forms (e.g. *Monday* $\rightarrow$ *5/9/2016*). It is a log-linear model that uses both lexical features and temporal tags.

To train these two models without corpora manually annotated with time expressions, we leverage a large database of known events as distant supervision. The event database is extracted automatically from Twitter using the open-domain IE system

[2] We focus on resolving dates, arguably the most important and frequent category of time expressions in social media data, and leave other phenomenon such as times and durations to traditional methods or future work.

[3] The closest work is HeidelTime's colloquial English version (Strötgen and Gertz, 2012) developed from annotated SMS data and slang dictionary. Our TweeTIME significantly outperforms on Twitter data.

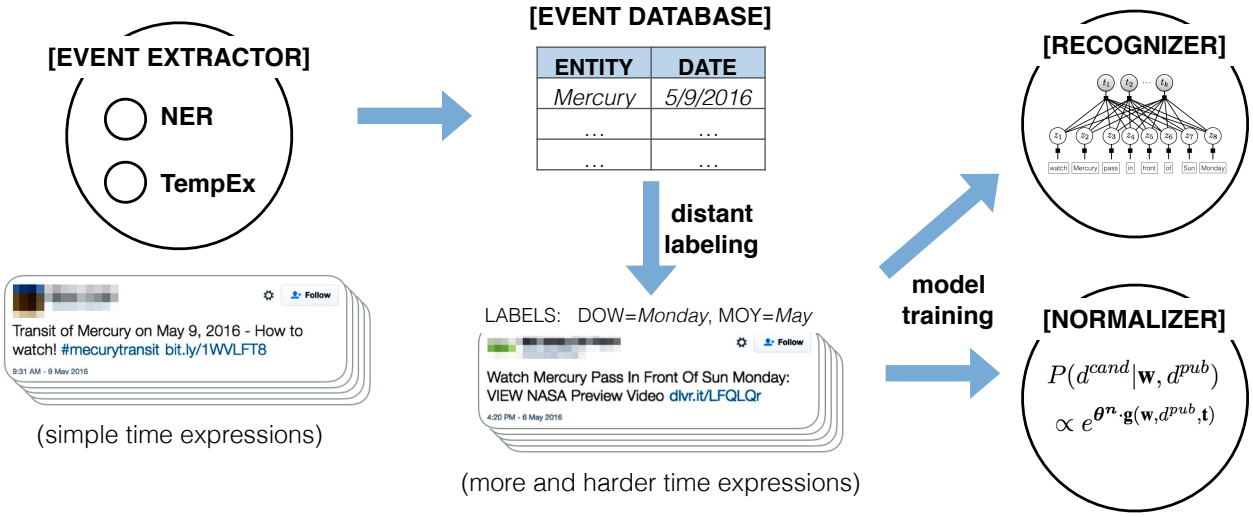


Figure 3: TweeTIME system diagram of model training.

Temporal Types	Possible Values (tags)
Timeline (TL)	$past, present, future$
Day of Week (DOW)	$Mon, Tue, \ldots, Sun$
Day of Month (DOM)	$1, 2, 3, \ldots, 31$
Month of Year (MOY)	$Jan, Feb, \ldots, Dec$
None (NA)	NA

Table 1: Our Temporal Recognizer can extract five different temporal types and assign one of their values to each word of a tweet.

proposed by Ritter et al. (2012). Each event consists of one or more named entities, in addition to the date on which the event takes place, for example [*Mercury*, *5/9/2016*]. Tweets are first processed by a Twitter named entity recognizer (Ritter et al., 2011), and a generic date resolver (Mani and Wilson, 2000). Events are then extracted based on the strength of association between each named entity and calendar date, as measured by a G^2 test on their co-occurrence counts. More details of the **Event Extractor** can be found in Section 5.1.

The following two sections describe the details of our **Temporal Recognizer** and **Temporal Normalizer** separately.

3 Distant Supervision for Recognizing Time Expressions

The goal of the recognizer is to predict the temporal tag of each word, given a sentence (or a tweet) $\mathbf{w} = w_1, \ldots, w_n$. We propose a multiple-instance learning model and a missing data model that are capable of learning word-level taggers given only sentence-level labels.

Our recognizer module in is built using a database of known events as *distant supervision*. We assume tweets published around the time of a known event that mention a central entity are also likely to contain time expressions referring to the event's date. For each event, such as [*Mercury*, *5/9/2016*], we gather all tweets that contain the central entity *Mercury* and are posted within 7 days of *5/9/2016*. We then label each tweet based on the event date in addition to the tweet's creation date. The sentence-level temporal tags for the tweet in Figure 1 are: TL=$future$, DOW=Mon, DOM=9, MOY=May.

3.1 Multiple-Instance Learning Temporal Tagging Model (MultiT)

Unlike supervised learning, where labeled instances are provided to the learner, in multiple instance learning scenarios (Dietterich et al., 1997), the learner is only provided with bags of instances labeled as either positive (where at least one instance is positive) or all negative. This is a close match to our problem setting, in which sentences are labeled with tags that should be assigned to one or more words.

We represent sentences and their labels using a graphical model that is divided into word-level and sentence-level variables (as shown in Figure 4). Unlike the standard supervised tagging prob-

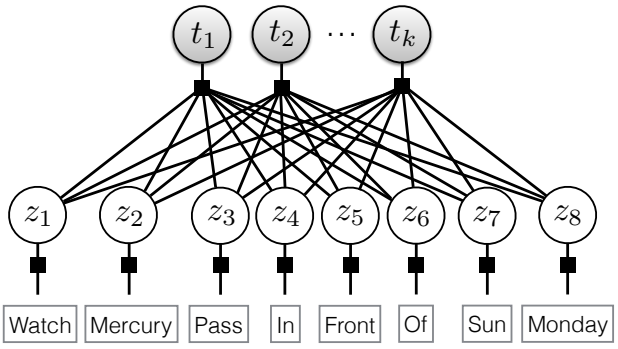


Figure 4: Multiple-Instance Learning Temporal Tagging Model – our approach to learn a word-level tagging model given only sentence-level labels. In this example a sentence-level variable $t_a = 1$ indicates the temporal tag DOW=Mon must be present and $t_b = 1$ indicates that the target date is in the future (TL=$future$). The multiple instance learning assumption implies that at least one word must be tagged with each of these present temporal tags. For example, ideally after training, the model will learn to assign z_8 to tag a and z_1 to tag b.

lem, we never directly observe the words' tags ($\mathbf{z} = z_1, \ldots, z_n$) during learning. Instead, they are latent and we only observe the date of an event mentioned in the text, from which we derive sentence-level binary variables $\mathbf{t} = t_1, \ldots, t_k$ corresponding to temporal tags for the sentence. Following previous work on multiple-instance learning (Hoffmann et al., 2011a; Xu et al., 2014), we model the connection between sentence-level labels and word-level tags using a set of deterministic-OR factors ϕ^{sent}.

The overall conditional probability of our model is defined as:

$$\begin{aligned} & P(\mathbf{t}, \mathbf{z}|\mathbf{w}; \boldsymbol{\theta^r}) \\ &= \frac{1}{Z} \prod_{i=1}^{k} \phi^{sent}(t_i, \mathbf{z}) \times \prod_{j=1}^{n} \phi^{word}(z_j, w_j) \\ &= \frac{1}{Z} \prod_{i=1}^{k} \phi^{sent}(t_i, \mathbf{z}) \times \prod_{j=1}^{n} e^{\boldsymbol{\theta^r} \cdot \mathbf{f}(z_j, w_j)} \end{aligned} \tag{1}$$

where $\mathbf{f}(z_j, w_j)$ is a feature vector and

$$\phi^{sent}(t_i, \mathbf{z}) = \begin{cases} 1 & \text{if } t_i = true \wedge \exists j : z_j = i \\ 1 & \text{if } t_i = false \wedge \forall j : z_j \neq i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We include a standard set of tagging features that includes word shape and identity in addition to prefixes and suffixes. To learn parameters $\boldsymbol{\theta^r}$ of the Temporal Tagger, we maximize the likelihood of the sentence-level heuristic labels conditioned on observed words over all tweets in the training corpus. Given a training instance **w** with label **t**, the gradient of the conditional log-likelihood with respect to the parameters is:

$$\begin{aligned} \nabla P(\mathbf{t}|\mathbf{w}) = & \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{w}, \mathbf{t}; \boldsymbol{\theta^r}) \cdot \mathbf{f}(\mathbf{z}, \mathbf{w}) \\ & - \sum_{\mathbf{t},\mathbf{z}} P(\mathbf{t}, \mathbf{z}|\mathbf{w}; \boldsymbol{\theta^r}) \cdot \mathbf{f}(\mathbf{z}, \mathbf{w}) \end{aligned} \tag{3}$$

This gradient is the difference of two conditional expectations over the feature vector **f**: a "clamped" expectation that is conditioned on the observed words and tags (**w**, **t**) and a "free" expectation that is only conditioned on the words in the text, **w**, and ignores the sentence-level labels. To make the inference tractable, we use a Viterbi approximation that replaces the expectations with maximization. Because each sentence corresponds to more than one temporal tag, the maximization of the "clamped" maximization is somewhat challenging to compute. We use the approximate inference algorithm of Hoffmann et al. (2011a), that views inference as a weighted set cover problem, with worst case running time ($|T| \cdot |W|$), where $|T|$ is the number of all possible temporal tag values and $|W|$ is the number of words in a sentence.

3.2 Missing Data Temporal Tagging Model (MiDaT)

While the multiple-instance learning assumption works well much of the time, it can easily be violated – there are many tweets that mention entities involved in an event but that never explicitly mention its date.

The missing data modeling approach to weakly supervised learning proposed by Ritter et. al. (2013) addresses this problem by relaxing the hard constraints of deterministic-OR factors, such as those described above, as soft constraints. Our missing-data model for weakly supervised tagging splits the sentence-level variables, t into two parts : m which represents whether a temporal tag is mentioned by at least one word of the tweet, and t' which represents

whether a temporal tag can be derived from the event date. A set of pairwise potentials $\psi(m_j, t'_j)$ are introduced that encourage (but don't strictly require) agreement between m_j and t'_j, that is:

$$\psi(m_j, t'_j) = \begin{cases} \alpha_p, \text{ if } t'_j \neq m_j \\ \alpha_r, \text{ if } t'_j = m_j \end{cases} \quad (4)$$

Here, α_p (Penalty), and α_r (Reward) are parameters for the MiDaT model. α_p is the penalty for extracting a temporal tag that is not related to the event-date and α_r is the reward for extracting a tag that matches the date.

During learning, if the local classifier is very confident, it is possible for a word to be labeled with a tag that is not derived from the event-date, and also for a sentence-level tag to be ignored, although either case will be penalized by the agreement potentials, $\psi(m_j, t'_j)$, in the global objective. We use a local-search approach to inference that was empirically demonstrated to nearly always yield exact solutions by Ritter et. al. (2013).

4 A Log-Linear Model for Normalizing Time Expressions

The Temporal Normalizer is built using a log-linear model which takes the tags **t** produced by the Temporal Recognizer as input and outputs one or more dates mentioned in a tweet. We formulate date normalization as a binary classification problem: given a tweet **w** published on date d^{pub}, we consider 22 candidate target dates (**w**, d_l^{cand}) such that $d_l^{cand} = d^{pub} + l$, where $l = -10, \ldots, -1, 0, +1, \ldots, +10$, limiting the possible date references that are considered within 10 days before or after the tweet creation date, in addition to $d_l^{cand} = null$ (the tweet does not mention a date). [4] While our basic approach has the limitation, that it is only able to predict dates within ± 10 days of the target date, we found that in practice the majority of date references on social media fall within this window. Our approach is also able to score dates outside this range that are generated by traditional approaches to resolving time expressions, as described in Section 5.3.3.

[4] Although the temporal recognizer is trained with tweets from ± 7 days around the event date, we found that extending the candidate date range to ± 10 days for the temporal normalizer increased the performance of TweeTIME in the dev set.

The normalizer is similarly trained using the event database as distant supervision. The probability that a tweet mentions a candidate date is estimated using a log-linear model:

$$P(d^{cand}|\mathbf{w}, d^{pub}) \propto e^{\boldsymbol{\theta^n} \cdot \mathbf{g}(\mathbf{w}, d^{pub}, \mathbf{t})} \quad (5)$$

where $\boldsymbol{\theta^n}$ and **g** are the parameter and feature vector respectively in the Temporal Normalizer. For every tweet and candidate date pair (**w**, d_l^{cand}), we extract the following set of features:

Temporal Tag Features that indicate whether the candidate date agrees with the temporal tags extracted by the Temporal Recognizer. Three cases can happen here: The recognizer can extract a tag that can not be derived from the candidate date; The recognizer can miss a tag derived from the candidate date; The recognizer can extract a tag that is derived from the candidate date.

Lexical Features that include two types of binary features from the tweet: 1) **Word Tag** features consist of conjunctions of words in the tweet and tags associated with the candidate date. We remove URLs, stop words and punctuation; 2) **Word POS** features that are the same as above, but include conjunctions of POS tags, words and temporal tags derived from the candidate date.

Time Difference Features are numerical features that indicate the distance between the creation date and the candidate date. They include difference of day ranges form -10 to 10 and the difference of week ranges from -2 to 2.

5 Experiments

In the following sub-sections we present experimental results on learning to resolve time expressions in Twitter using minimal supervision. We start by describing our dataset, and proceed to present our results, including a large-scale evaluation on heuristically-labeled data and an evaluation comparing against human judgements.

5.1 Data Collection

We collected around 120 million tweets posted in a one year window starting from April 2011 to May 2012. These tweets were automatically annotated with named entities, POS tags and TempEx dates (Ritter et al., 2011).

From this automatically-annotated corpus we extract the top $10,000$ events and their corresponding dates using the G^2 test, which measures the strength of association between an entity and date using the log-likelihood ratio between a model in which the entity is conditioned on the date and a model of independence (Ritter et al., 2012). Events extracted using this approach then simply consist of the highest-scoring entity-date pairs, for example [*Mercury, 5/9/2016*].

After automatically extracting the database of events, we next gather all tweets that mention an entity from the list that are also written within ± 7 days of the event. These tweets and the dates of the known events serve as labeled examples that are likely to mention a known date.

We also include a set of pseudo-negative examples, that are unlikely to refer to any event, by gathering a random sample of tweets that do not mention any of the top $10,000$ events and where TempEx does not extract any date.

5.2 Large-Scale Heuristic Evaluation

We first evaluate our tagging model, by testing how well it can predict the heuristically generated labels. As noted in previous work on distant supervision (Mintz et al., 2009a), this type of evaluation usually under-estimates precision, however it provides us with a useful intrinsic measure of performance.

In order to provide even coverage of months in the training and test set, we divide the twitter corpus into 3 subsets based on the mod-5 week of each tweet's creation date. To train system we use tweets that are created in $1st$, $2nd$ or $3rd$ weeks. To tune parameters of the MiDaT model we used tweets from $5th$ weeks, and to evaluate the performance of the trained model we used tweets from $4th$ weeks.

	Precision	Recall	F-value
MultiT	0.61	0.21	0.32
MiDaT	0.67	0.31	0.42

Table 2: Performance comparison of MultiT and MiDaT at predicting heuristically generated tags on the dev set.

The performance of the MiDaT model varies with the penalty and reward parameters. To find a (near) optimal setting of the values we performed a grid search on the dev set and found that a penalty of -25 and reward of 500 works best. A comparison of MultiT and MiDaT's performance at predicting heuristically generated labels is shown in Table 2.

The word level tags predicted by the temporal recognizer are used as the input to the temporal normalizer, which predicts the referenced date from each tweet. The overall system's performance at predicting event dates on the automatically generated test set, compared against SUTime, is shown in Table 3.

	System	Prec.	Recall	F-value
dev set	TweeTIME	0.93	0.69	0.79
	SUTime	0.89	0.64	0.75
test set	TweeTIME	0.97	0.94	0.96
	SUTime	0.85	0.75	0.80

Table 3: Performance comparison of TweeTIME and SUTime at predicting heuristically labeled normalized dates.

5.3 Evaluation Against Human Judgements

In addition to automatically evaluating our tagger on a large corpus of heuristically-labeled tweets, we also evaluate the performance of our tagging and date-resolution models on a random sample of tweets taken from a much later time period, that were manually annotated by the authors.

5.3.1 Word-Level Tags

To evaluate the performance of the MiDaT-tagger we randomly selected 50 tweets and labeled each word with its corresponding tag. Against this hand annotated test set, MiDaT achieves Precision=0.54, Recall=0.45 and F-value=0.49. A few examples of word-level tags predicted by MiDaT are shown in Table 4. We found that because the tags are learned as latent variables inferred by our model, they sometimes don't line up exactly with our intuitions but still provide useful predictions, for example in Table 4, *Christmas* is labeled with the tag MOY=dec.

5.3.2 End-to-end Date Resolution

To evaluate the final performance of our system and compare against existing state-of-the art time resolvers, we randomly sampled 250 tweets from 2014-2016 and manually annotated them with normalized dates; note that this is a separate date range from our weakly-labeled training data which is taken from 2011-2012. We use 50 tweets as a development set and the remaining 200 as a final test set.

Tweets and their corresponding word tags (word^{tag})
Im^{NA} hella^{NA} excited^{future} for^{NA} tomorrow^{future}
Kick^{NA} off^{NA} the^{NA} New^{future} Year^{future} Right^{NA} @^{NA} #ClubLacura^{NA} #FRIDAY^{fri} !^{NA} HOSTED^{NA} BY^{NA} [[^{NA} DC^{NA} Young^{NA} Fly^{NA}]]^{NA}
@OxfordTownHall^{NA} Thks^{NA} for^{NA} a^{NA} top^{NA} night^{NA} at^{NA} our^{NA} Christmas^{dec} party^{NA} on^{NA} Fri!^{fri} Compliments^{NA} to^{NA} chef!^{NA} (Rose^{NA} melon^{NA} *cantaloupe*^{NA} :)^{NA}
Im^{NA} proud^{NA} to^{NA} say^{NA} that^{NA} I^{NA} breathed^{past} the^{NA} same^{NA} air^{NA} as^{NA} Harry^{NA} on^{NA} March^{mar} 21,^{21} 2015.^{NA} #KCA^{NA} #Vote1DUK^{NA}
C'mon^{present} let's^{present} jack^{NA} Tonight^{present} will^{NA} be^{present} a^{NA} night^{NA} to^{NA} remember.^{NA}

Table 4: Example MiDaT tagging output on the test set.

	Precision	**Recall**	**F-value**
TweeTIME	0.61	0.81	0.70
- Day Diff.	0.46	0.72	0.56
- Lexical&POS	0.48	0.80	0.60
- Week Diff.	0.49	0.85	0.62
- Lexical	0.50	0.88	0.64
- Temporal Tag	0.57	0.83	0.68

Table 5: Feature ablation of the Temporal Resolver by removing each individual feature group from the full set.

	System	**Prec.**	**Recall**	**F-value**
dev set	TweeTIME	**0.61**	0.81	**0.70**
	TweeTIME+SU	**0.67**	**0.83**	**0.74**
	SUTime	0.51	**0.86**	0.64
	TempEx	0.58	0.64	0.61
	HeidelTime	0.57	0.63	0.60
	UWTime	0.49	0.57	0.53
test set	TweeTIME	**0.58**	**0.70**	**0.63**
	TweeTIME+SU	**0.62**	**0.76**	**0.68**
	SUTime	0.54	0.64	0.58
	TempEx	0.56	0.58	0.57
	HeidelTime	0.43	0.52	0.47
	UWTime	0.39	0.50	0.44

Table 6: Performance comparison of TweeTIME against state-of-the-art temporal taggers. TweeTIME+SU uses our proposed approach to system combination, re-scoring output from SUTime using extracted features and learned parameters from TweeTIME.

We experimented with different feature sets on the development data. Feature ablation experiments are presented in Table 5.

The final performance of our system, compared against a range of state-of-the-art time resolvers is presented in Table 6. We see that TweeTIME outperforms SUTime, Tempex, HeidelTime (using its COLLOQUIAL mode, which is designed for SMS text) and UWTime. Brief descriptions of each system can be found in Section 6.

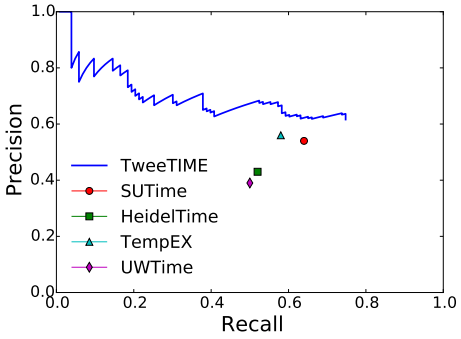


Figure 5: Precision and recall at resolving time expressions compared against human judgements. TweeTIME achieves higher precision at comparable recall than other state-of-the-art systems.

5.3.3 System Combination with SUTime

As our basic TweeTIME system is designed to predict dates within ± 10 days of the creation date, it fails when a tweet refers to a date outside this range. To overcome this limitation we append the date predicted by SUTime in the list of candidate days. We then re-rank SUTime's predictions using our log-linear model, and include its output as a predicted date if the confidence of our normalizer is sufficiently high.

5.3.4 Error Analysis

We manually examined the system outputs and found 7 typical categories of errors (see examples in Table 7):

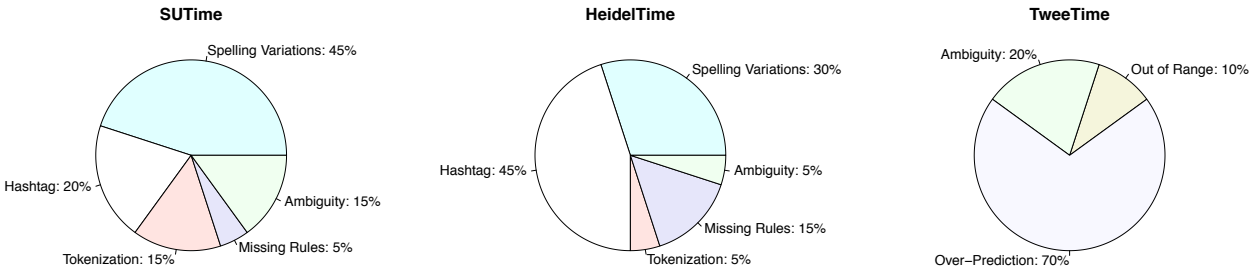


Figure 6: Error analyses for different temporal resolvers

Spelling Variation: Twitter users are very creative in their use of spelling and abbreviations. For example, a large number of variations of the word *tomorrow* can be found in tweets, including *2morrow*, *2mrw*, *tmrw*, *2mrow* and so on. Previous temporal resolvers often fail in these cases, while TweeTIME significantly reduces such errors.

Ambiguity: In many cases, temporal words like *Friday* in the tweet *Is it Friday yet?* may not refer to any specific event or date, but are often predicted incorrectly. Also included in this category are cases where the future and past are confused. For example, predicting the past Friday, when it is actually the coming Friday.

Missing Rule: Cases where specific temporal keywords, such as *April Fools*, are not covered by the rule-based systems.

Tokenization: Traditional systems tend to be very sensitive to incorrect tokenization and have trouble to handle expressions such as *9th-december*, *May 9,2015* or *Jan1*. For the following Tweet:

> JUST IN Delhi high court asks state government to submit data on changes in pollution level since #OddEven rule came into effect on *Jan1*

TweeTIME is able to correctly extract *01/01/2016*, whereas HeidelTime, SUTime, TempEX and UWTime all failed to extract any dates.

Hashtag: Hashtags can carry temporal information, for example, #September11. Only our system that is adapted to social media can resolve these cases.

Out of Range: TweeTIME only predicts dates within 10 days before or after the tweet. Time expressions referring to dates outside this range will not be predicted correctly. System combination with SUTime (Section 5.3.3) only partially addressed this problem.

Over-Prediction: Unlike rule-based systems, TweeTIME has a tendency to over-predict when there is no explicit time expression in the tweets, possibly because of the presence of present tense verbs. Such mistakes could also happen in some past tense verbs.

Because TweeTIME resolves time expressions using a very different approach compared to traditional methods, its distribution of errors is quite distinct, as illustrated in Figure 6.

6 Related Work

Temporal Resolvers primarily utilize either rule-based or probabilistic approaches. Notable rule-based systems such as TempEx (Mani and Wilson, 2000), SUTime (Chang and Manning, 2012) and HeidelTime (Strötgen and Gertz, 2013) provide particularly competitive performance compared to the state-of-the-art machine learning methods. Probabilistic approaches use supervised classifiers trained on in-domain annotated data (Kolomiyets and Moens, 2010; Bethard, 2013a; Filannino et al., 2013) or hybrid with hand-engineered rules (UzZaman and Allen, 2010; Lee et al., 2014). UWTime (Lee et al., 2014) is one of the most recent and competitive systems and uses Combinatory Categorial Grammar (CCG).

Although the recent research challenge TempEval (UzZaman et al., 2013; Bethard and Savova, 2016) offers an evaluation in the clinical domain besides newswire, most participants used the provided annotated corpus to train supervised models in addition to employing hand-coded rules. Previous work on adapting temporal taggers primarily focus on scaling up to more languages. HeidelTime was extended to multilingual (Strötgen and Gertz, 2015), colloquial (SMS) and scientific texts (Strötgen and Gertz, 2012) using dictionaries and additional in-domain

Error Category	Tweet	Gold Date	Predicted Date
Spelling	I cant believe *tmrw* is *fri*..the week flys by	2015-03-06	None (SUTime, HeidelTime)
Ambiguity	RT @Iyaimkatie: Is it Friday yet?????	None	2015-12-04 (TweeTime, SUTime, HeidelTime)
Missing Rule	#49ers #sanfrancisco 49ers fans should be oh so wary of *April Fools* pranks	2015-04-01	None (HeidelTime)
Tokenization	100000 - still waiting for that reply from *9th-december* lmao. you're pretty funny and chill	2015-12-09	None (SUTime, HeidelTime)
Hashtag	RT @arianatotally: Who listening to the *#SATURDAY* #Night w/ @AlexAngelo?I'm loving it.	2015-04-11	None (SUTime, HeidelTime)
Out of Range	RT @460km: In memory of Constable Christine Diotte @rcmpgrcpolice EOW: *March 12, 2002* #HeroesInLife #HerosEnVie	2002-03-12	2015-03-12 (TweeTime)
Over-Prediction	RT @tinatbh: January 2015: this will be my year December 2015: maybe not.	None	2015-12-08 (TweeTime)

Table 7: Representative Examples of System (SUTime, HeidelTime, TweeTIME) Errors

annotated data. One existing work used distant supervision (Angeli et al., 2012; Angeli and Uszkoreit, 2013), but for normalization only, assuming gold time mentions as input. They used an EM-style bootstrapping approach and a CKY parser.

Distant Supervision has recently become popular in natural language processing. Much of the work has focused on the task of relation extraction (Craven and Kumlien, 1999; Bunescu and Mooney, 2007; Mintz et al., 2009b; Riedel et al., 2010; Hoffmann et al., 2011b; Nguyen and Moschitti, 2011; Surdeanu et al., 2012; Xu et al., 2013; Ritter et al., 2013; Angeli et al., 2014). Recent work also shows exciting results on extracting named entities (Ritter et al., 2011; Plank et al., 2014), emotions (Purver and Battersby, 2012), sentiment (Marchetti-Bowick and Chambers, 2012), as well as finding evidence in medical publications (Wallace et al., 2016). Our work is closely related to the joint word-sentence model that exploits multiple-instance learning for paraphrase identification (Xu et al., 2014) in Twitter.

7 Conclusions

In this paper, we showed how to learn time resolvers from large amounts of unlabeled text, using a database of known events as distant supervision. We presented a method for learning a word-level temporal tagging models from tweets that are heuristically labeled with only sentence-level labels. This approach was further extended to account for the case of missing tags, or temporal properties that are not explicitly mentioned in the text of a tweet. These temporal tags were then combined with a variety of other features in a novel date-resolver that predicts normalized dates referenced in a Tweet. By learning from large quantities of in-domain data, we were able to achieve 0.68 F1 score on the end-to-end time normalization task for social media data, significantly outperforming SUTime, TempEx, HeidelTime and UWTime on this challenging dataset for time normalization.

Acknowledgments

We would like to thank the anonymous reviewers for helpful feedback on a previous draft. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1464128. Alan Ritter is supported by the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C-0114. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government.

References

Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. 2007. On the value of temporal information in information retrieval. In *ACM SIGIR Forum*, volume 41, pages 35–41. ACM.

Gabor Angeli and Jakob Uszkoreit. 2013. Language-independent discriminative parsing of temporal expressions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gabor Angeli, Christopher D Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Steven Bethard and Guergana Savova. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*.

Steven Bethard. 2013a. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Steven Bethard. 2013b. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the Web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers. 2013. NavyTime: Event and time ordering from raw text. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*.

Angel X Chang and Christopher D Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. *Proccedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Technologies (NAACL)*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Leon Derczynski and Robert J Gaizauskas. 2013. Temporal signals help label temporal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1).

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. ManTIME: Temporal expression identification and normalization in the TempEval-3 challenge. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011a. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011b. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of the Fourth Text Analysis Conference (TAC)*.

Nattiya Kanhabua, Sara Romano, Avaré Stewart, and Wolfgang Nejdl. 2012. Supporting temporal analytics for health-related events in microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*.

Oleksandr Kolomiyets and Marie-Francine Moens. 2010. KUL: Recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL)*.

Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for microblogs with distant supervision: Political forecasting with Twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009a. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the Association of Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009b. Distant supervision for relation extraction without labeled data. In *Proceedigns of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL)*.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Barbara Plank, Dirk Hovy, Ryan McDonald, and Anders Søgaard. 2014. Adapting taggers to twitter with not-so-distant supervision. pages 1783–1792.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedigns of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

Alan Ritter, Mausam, Sam Clark, and Oren Etzioni. 2011. Named entity recognition in Tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics (TACL)*, 1:367–378.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from Twitter. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*.

H Andrew Schwartz, Greg Park, Maarten Sap, Evan Weingarten, Johannes Eichstaedt, Margaret Kern, Jonah Berger, Martin Seligman, and Lyle Ungar. 2015. Extracting human temporal orientation in Facebook language. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Naushad UzZaman and James F Allen. 2010. TRIPS and TRIOS system for Tempeval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Brian Zhu, and Iain J Marshall. 2016. Extracting PICO sentences from clinical trial reports using supervised distant supervision. *Journal of Machine Learning Research (JMLR)*.

Wei Xu, Raphael Hoffmann, Zhao Le, and Ralph Grishman. 2013. Filling knowledge base gaps for distant

supervision of relation extraction. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics (TACL)*, 2(1).

Language as a Latent Variable: Discrete Generative Models for Sentence Compression

Yishu Miao[1], **Phil Blunsom**[1,2]
[1]University of Oxford, [2]Google Deepmind
{yishu.miao, phil.blunsom}@cs.ox.ac.uk

Abstract

In this work we explore deep generative models of text in which the latent representation of a document is itself drawn from a discrete language model distribution. We formulate a variational auto-encoder for inference in this model and apply it to the task of compressing sentences. In this application the generative model first draws a latent summary sentence from a background language model, and then subsequently draws the observed sentence conditioned on this latent summary. In our empirical evaluation we show that generative formulations of both abstractive and extractive compression yield state-of-the-art results when trained on a large amount of supervised data. Further, we explore semi-supervised compression scenarios where we show that it is possible to achieve performance competitive with previously proposed supervised models while training on a fraction of the supervised data.

1 Introduction

The recurrent sequence-to-sequence paradigm for natural language generation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014) has achieved remarkable recent success and is now the approach of choice for applications such as machine translation (Bahdanau et al., 2015), caption generation (Xu et al., 2015) and speech recognition (Chorowski et al., 2015). While these models have developed sophisticated conditioning mechanisms, e.g. attention, fundamentally they are discriminative models trained only to approximate the conditional output distribution of strings. In this paper we explore modelling the joint distribution of string pairs using a deep generative model and employing a discrete variational auto-encoder (VAE) for inference (Kingma and Welling, 2014; Rezende et al., 2014; Mnih and Gregor, 2014). We evaluate our generative approach on the task of sentence compression. This approach provides both alternative supervised objective functions and the opportunity to perform semi-supervised learning by exploiting the VAEs ability to marginalise the latent compressed text for unlabelled data.

Auto-encoders (Rumelhart et al., 1985) are a typical neural network architecture for learning compact data representations, with the general aim of performing dimensionality reduction on embeddings (Hinton and Salakhutdinov, 2006). In this paper, rather than seeking to embed inputs as points in a vector space, we describe them with explicit natural language sentences. This approach is a natural fit for summarisation tasks such as sentence compression. According to this, we propose a generative **auto-encoding sentence compression** (ASC) model, where we introduce a latent language model to provide the variable-length compact summary. The objective is to perform Bayesian inference for the posterior distribution of summaries conditioned on the observed utterances. Hence, in the framework of VAE, we construct an inference network as the variational approximation of the posterior, which generates compression samples to optimise the variational lower bound.

The most common family of variational auto-encoders relies on the reparameterisation trick, which is not applicable for our discrete latent language model. Instead, we employ the REINFORCE algorithm (Mnih et al., 2014; Mnih and Gregor, 2014)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 319–328,
Austin, Texas, November 1-5, 2016.

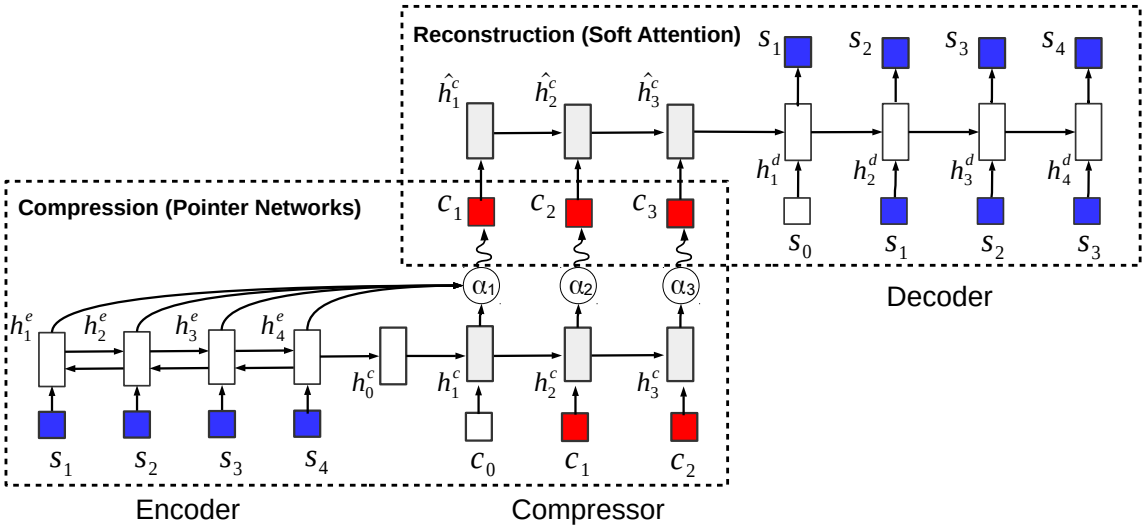


Figure 1: Auto-encoding Sentence Compression Model

to mitigate the problem of high variance during sampling-based variational inference. Nevertheless, when directly applying the RNN encoder-decoder to model the variational distribution it is very difficult to generate reasonable compression samples in the early stages of training, since each hidden state of the sequence would have $|V|$ possible words to be sampled from. To combat this we employ pointer networks (Vinyals et al., 2015) to construct the variational distribution. This biases the latent space to sequences composed of words only appearing in the source sentence (i.e. the size of softmax output for each state becomes the length of current source sentence), which amounts to applying an extractive compression model for the variational approximation.

In order to further boost the performance on sentence compression, we employ a supervised **forced-attention sentence compression** model (FSC) trained on labelled data to teach the ASC model to generate compression sentences. The FSC model shares the pointer network of the ASC model and combines a softmax output layer over the whole vocabulary. Therefore, while training on the sentence-compression pairs, it is able to balance copying a word from the source sentence with generating it from the background distribution. More importantly, by jointly training on the labelled and unlabelled datasets, this shared pointer network enables the model to work in a semi-supervised scenario. In this case, the FSC teaches the ASC to generate reasonable samples, while the pointer network trained on a large unlabelled data set helps the FSC model to perform better abstractive summarisation.

In Section 6, we evaluate the proposed model by jointly training the generative (ASC) and discriminative (FSC) models on the standard Gigaword sentence compression task with varying amounts of labelled and unlabelled data. The results demonstrate that by introducing a latent language variable we are able to match the previous benchmakers with small amount of the supervised data. When we employ our mixed discriminative and generative objective with all of the supervised data the model significantly outperforms all previously published results.

2 Auto-Encoding Sentence Compression

In this section, we introduce the auto-encoding sentence compression model (Figure 1)[1] in the framework of variational auto-encoders. The ASC model consists of four recurrent neural networks – an encoder, a compressor, a decoder and a language model.

Let $\boldsymbol{s}$ be the source sentence, and $\boldsymbol{c}$ be the compression sentence. The **compression** model (encoder-compressor) is the inference network $q_\phi(\boldsymbol{c}|\boldsymbol{s})$ that takes source sentences $\boldsymbol{s}$ as inputs and generates extractive compressions $\boldsymbol{c}$. The **reconstruction**

[1]The language model, layer connections and decoder soft attentions are omitted in Figure 1 for clarity.

model (compressor-decoder) is the generative network $p_\theta(\boldsymbol{s}|\boldsymbol{c})$ that reconstructs source sentences $\boldsymbol{s}$ based on the latent compressions $\boldsymbol{c}$. Hence, the forward pass starts from the encoder to the compressor and ends at the decoder. As the prior distribution, a language model $p(\boldsymbol{c})$ is pre-trained to regularise the latent compressions so that the samples drawn from the compression model are likely to be reasonable natural language sentences.

2.1 Compression

For the compression model (encoder-compressor), $q_\phi(\boldsymbol{c}|\boldsymbol{s})$, we employ a pointer network consisting of a bidirectional LSTM encoder that processes the source sentences, and an LSTM compressor that generates compressed sentences by attending to the encoded source words.

Let $\boldsymbol{s}_i$ be the words in the source sentences, $\boldsymbol{h}_i^e$ be the corresponding state outputs of the encoder. $\boldsymbol{h}_i^e$ are the concatenated hidden states from each direction:

$$\boldsymbol{h}_i^e = f_{\overrightarrow{\text{enc}}}(\vec{\boldsymbol{h}}_{i-1}^e, \boldsymbol{s}_i)||f_{\overleftarrow{\text{enc}}}(\overleftarrow{\boldsymbol{h}}_{i+1}^e, \boldsymbol{s}_i) \quad (1)$$

Further, let $\boldsymbol{c}_j$ be the words in the compressed sentences, $\boldsymbol{h}_j^c$ be the state outputs of the compressor. We construct the predictive distribution by attending to the words in the source sentences:

$$\boldsymbol{h}_j^c = f_{\text{com}}(\boldsymbol{h}_{j-1}^c, \boldsymbol{c}_{j-1}) \quad (2)$$

$$\boldsymbol{u}_j(i) = \boldsymbol{w}_3^T \tanh(\boldsymbol{W}_1\boldsymbol{h}_j^c + \boldsymbol{W}_2\boldsymbol{h}_i^e) \quad (3)$$

$$q_\phi(\boldsymbol{c}_j|\boldsymbol{c}_{1:j-1}, \boldsymbol{s}) = \text{softmax}(\boldsymbol{u}_j) \quad (4)$$

where $\boldsymbol{c}_0$ is the start symbol for each compressed sentence and $\boldsymbol{h}_0^c$ is initialised by the source sentence vector of $\boldsymbol{h}_{|\boldsymbol{s}|}^e$. In this case, all the words $\boldsymbol{c}_j$ sampled from $q_\phi(\boldsymbol{c}_j|\boldsymbol{c}_{1:j-1}, \boldsymbol{s})$ are the subset of the words appeared in the source sentence (i.e. $\boldsymbol{c}_j \in \boldsymbol{s}$).

2.2 Reconstruction

For the reconstruction model (compressor-decoder) $p_\theta(\boldsymbol{s}|\boldsymbol{c})$, we apply a soft attention sequence-to-sequence model to generate the source sentence $\boldsymbol{s}$ based on the compression samples $\boldsymbol{c} \sim q_\phi(\boldsymbol{c}|\boldsymbol{s})$.

Let $\boldsymbol{s}_k$ be the words in the reconstructed sentences and $\boldsymbol{h}_k^d$ be the corresponding state outputs of the decoder:

$$\boldsymbol{h}_k^d = f_{\text{dec}}(\boldsymbol{h}_{k-1}^d, \boldsymbol{s}_{k-1}) \quad (5)$$

In this model, we directly use the recurrent cell of the compressor to encode the compression samples[2]:

$$\hat{\boldsymbol{h}}_j^c = f_{\text{com}}(\hat{\boldsymbol{h}}_{j-1}^c, \boldsymbol{c}_j) \quad (6)$$

where the state outputs $\hat{\boldsymbol{h}}_j^c$ corresponding to the word inputs $\boldsymbol{c}_j$ are different from the outputs $\boldsymbol{h}_j^c$ in the compression model, since we block the information from the source sentences. We also introduce a start symbol $\boldsymbol{s}_0$ for the reconstructed sentence and $\boldsymbol{h}_0^d$ is initialised by the last state output $\hat{\boldsymbol{h}}_{|\boldsymbol{c}|}^c$. The soft attention model is defined as:

$$v_k(j) = \boldsymbol{w}_6^T \tanh(\boldsymbol{W}_4\boldsymbol{h}_k^d + \boldsymbol{W}_5\hat{\boldsymbol{h}}_j^c) \quad (7)$$

$$\gamma_k(j) = \text{softmax}(v_k(j)) \quad (8)$$

$$\boldsymbol{d}_k = \sum\nolimits_j^{|\boldsymbol{c}|} \gamma_k(j)\hat{\boldsymbol{h}}_j^c(v_k(j)) \quad (9)$$

We then construct the predictive probability distribution over reconstructed words using a softmax:

$$p_\theta(\boldsymbol{s}_k|\boldsymbol{s}_{1:k-1}, \boldsymbol{c}) = \text{softmax}(\boldsymbol{W}_7\boldsymbol{d}_k) \quad (10)$$

2.3 Inference

In the ASC model there are two sets of parameters, ϕ and θ, that need to be updated during inference. Due to the non-differentiability of the model, the reparameterisation trick of the VAE is not applicable in this case. Thus, we use the REINFORCE algorithm (Mnih et al., 2014; Mnih and Gregor, 2014) to reduce the variance of the gradient estimator.

The variational lower bound of the ASC model is:

$$\begin{aligned} L &= \mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[\log p_\theta(\boldsymbol{s}|\boldsymbol{c})] - D_{KL}[q_\phi(\boldsymbol{c}|\boldsymbol{s})||p(\boldsymbol{c})] \\ &\leqslant \log \int \frac{q_\phi(\boldsymbol{c}|\boldsymbol{s})}{q_\phi(\boldsymbol{c}|\boldsymbol{s})} p_\theta(\boldsymbol{s}|\boldsymbol{c})p(\boldsymbol{c})d\boldsymbol{c} = \log p(\boldsymbol{s}) \end{aligned} \quad (11)$$

Therefore, by optimising the lower bound (Eq. 11), the model balances the selection of keywords for the summaries and the efficacy of the composed compressions, corresponding to the reconstruction error and KL divergence respectively.

In practise, the pre-trained language model prior $p(\boldsymbol{c})$ prefers short sentences for compressions. As one of the drawbacks of VAEs, the KL divergence term in the lower bound pushes every sample drawn

[2]The recurrent parameters of the compressor are not updated by the gradients from the reconstruction model.

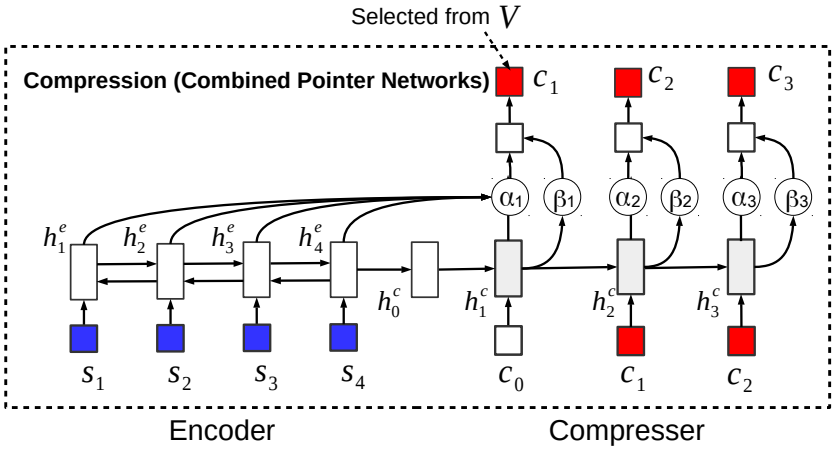


Figure 2: Forced Attention Sentence Compression Model

from the variational distribution towards the prior. Thus acting to regularise the posterior, but also to restrict the learning of the encoder. If the estimator keeps sampling short compressions during inference, the LSTM decoder would gradually rely on the contexts from the decoded words instead of the information provided by the compressions, which does not yield the best performance on sentence compression.

Here, we introduce a co-efficient λ to scale the learning signal of the KL divergence:

$$L=\mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[\log p_\theta(\boldsymbol{s}|\boldsymbol{c})]-\lambda D_{KL}[q_\phi(\boldsymbol{c}|\boldsymbol{s})||p(\boldsymbol{c})] \quad (12)$$

Although we are not optimising the exact variational lower bound, the ultimate goal of learning an effective compression model is mostly up to the reconstruction error. In Section 6, we empirically apply $\lambda = 0.1$ for all the experiments on ASC model. Interestingly, λ controls the compression rate of the sentences which can be a good point to be explored in future work.

During the inference, we have different strategies for updating the parameters of ϕ and θ. For the parameters θ in the reconstruction model, we directly update them by the gradients:

$$\frac{\partial L}{\partial \theta} = \mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[\frac{\partial \log p_\theta(\boldsymbol{s}|\boldsymbol{c})}{\partial \theta}] \approx \frac{1}{M}\sum_m \frac{\partial \log p_\theta(\boldsymbol{s}|\boldsymbol{c}^{(m)})}{\partial \theta} \quad (13)$$

where we draw M samples $\boldsymbol{c}^{(m)} \sim q_\phi(\boldsymbol{c}|\boldsymbol{s})$ independently for computing the stochastic gradients.

For the parameters ϕ in the compression model, we firstly define the learning signal,

$$l(\boldsymbol{s},\boldsymbol{c}) = \log p_\theta(\boldsymbol{s}|\boldsymbol{c}) - \lambda(\log q_\phi(\boldsymbol{c}|\boldsymbol{s}) - \log p(\boldsymbol{c})).$$

Then, we update the parameters ϕ by:

$$\frac{\partial L}{\partial \phi} = \mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[l(\boldsymbol{s},\boldsymbol{c})\frac{\partial \log q_\phi(\boldsymbol{c}|\boldsymbol{s})}{\partial \phi}] \approx \frac{1}{M}\sum_m [l(\boldsymbol{s},\boldsymbol{c}^{(m)})\frac{\partial \log q_\phi(\boldsymbol{c}^{(m)}|\boldsymbol{s})}{\partial \phi}] \quad (14)$$

However, this gradient estimator has a big variance because the learning signal $l(\boldsymbol{s},\boldsymbol{c}^{(m)})$ relies on the samples from $q_\phi(\boldsymbol{c}|\boldsymbol{s})$. Therefore, following the REINFORCE algorithm, we introduce two baselines b and $b(\boldsymbol{s})$, the centred learning signal and input-dependent baseline respectively, to help reduce the variance.

Here, we build an MLP to implement the input-dependent baseline $b(\boldsymbol{s})$. During training, we learn the two baselines by minimising the expectation:

$$\mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[(l(\boldsymbol{s},\boldsymbol{c}) - b - b(\boldsymbol{s}))^2]. \quad (15)$$

Hence, the gradients w.r.t. ϕ are derived as,

$$\frac{\partial L}{\partial \phi} \approx \frac{1}{M}\sum_m (l(\boldsymbol{s},\boldsymbol{c}^{(m)}) - b - b(\boldsymbol{s}))\frac{\partial \log q_\phi(\boldsymbol{c}^{(m)}|\boldsymbol{s})}{\partial \phi} \quad (16)$$

which is basically a likelihood-ratio estimator.

3 Forced-attention Sentence Compression

In neural variational inference, the effectiveness of training largely depends on the quality of the inference network gradient estimator. Although we introduce a biased estimator by using pointer networks, it is still very difficult for the compression model to generate reasonable natural language sentences at the early stage of learning, which results in

high-variance for the gradient estimator. Here, we introduce our supervised forced-attention sentence compression (FSC) model to teach the compression model to generate coherent compressed sentences.

Neither directly replicating the pointer network of ASC model, nor using a typical sequence-to-sequence model, the FSC model employs a force-attention strategy (Figure 2) that encourages the compressor to select words appearing in the source sentence but keeps the original full output vocabulary V. The force-attention strategy is basically a combined pointer network that chooses whether to select a word from the source sentence $\boldsymbol{s}$ or to predict a word from V at each recurrent state. Hence, the combined pointer network learns to copy the source words while predicting the word sequences of compressions. By sharing the pointer networks between the ASC and FSC model, the biased estimator obtains further positive biases by training on a small set of labelled source-compression pairs.

Here, the FSC model makes use of the compression model (Eq. 1 to 4) in the ASC model,

$$\alpha_j = \text{softmax}(\boldsymbol{u}_j), \tag{17}$$

where $\alpha_j(i), i \in (1, \ldots, |\boldsymbol{s}|)$ denotes the probability of selecting $\boldsymbol{s}_i$ as the prediction for $\boldsymbol{c}_j$.

On the basis of the pointer network, we further introduce the probability of predicting $\boldsymbol{c}_j$ that is selected from the full vocabulary,

$$\beta_j = \text{softmax}(\boldsymbol{W}\boldsymbol{h}_j^c), \tag{18}$$

where $\beta_j(w), w \in (1, \ldots, |V|)$ denotes the probability of selecting the wth from V as the prediction for $\boldsymbol{c}_j$. To combine these two probabilities in the RNN, we define a selection factor $\boldsymbol{t}$ for each state output, which computes the semantic similarities between the current state and the attention vector,

$$\boldsymbol{\eta}_j = \sum_i^{|\boldsymbol{s}|} \alpha_j(i)\boldsymbol{h}_i^e \tag{19}$$

$$\boldsymbol{t}_j = \sigma(\boldsymbol{\eta}_j^T \boldsymbol{M}\boldsymbol{h}_j^c). \tag{20}$$

Hence, the probability distribution over compressed words is defined as,

$$p(\boldsymbol{c}_j|\boldsymbol{c}_{1:j-1}, \boldsymbol{s}) = \begin{cases} \boldsymbol{t}_j\alpha_j(i) + (1-\boldsymbol{t}_j)\beta_j(\boldsymbol{c}_j), & \boldsymbol{c}_j = \boldsymbol{s}_i \\ (1-\boldsymbol{t}_j)\beta_j(\boldsymbol{c}_j), & \boldsymbol{c}_j \notin \boldsymbol{s} \end{cases} \tag{21}$$

Essentially, the FSC model is the extended compression model of ASC by incorporating the pointer network with a softmax output layer over the full vocabulary. So we employ ϕ to denote the parameters of the FSC model $p_\phi(\boldsymbol{c}|\boldsymbol{s})$, which covers the parameters of the variational distribution $q_\phi(\boldsymbol{c}|\boldsymbol{s})$.

4 Semi-supervised Training

As the auto-encoding sentence compression (ASC) model grants the ability to make use of an unlabelled dataset, we explore a semi-supervised training framework for the ASC and FSC models. In this scenario we have a labelled dataset that contains source-compression parallel sentences, $(\boldsymbol{s}, \boldsymbol{c}) \in \mathbb{L}$, and an unlabelled dataset that contains only source sentences $\boldsymbol{s} \in \mathbb{U}$. The FSC model is trained on $\mathbb{L}$ so that we are able to learn the compression model by maximising the log-probability,

$$F = \sum_{(\boldsymbol{c},\boldsymbol{s})\in\mathbb{L}} \log p_\phi(\boldsymbol{c}|\boldsymbol{s}). \tag{22}$$

While the ASC model is trained on $\mathbb{U}$, where we maximise the modified variational lower bound,

$$L = \sum_{\boldsymbol{s}\in\mathbb{U}} (\mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[\log p_\theta(\boldsymbol{s}|\boldsymbol{c})] - \lambda D_{KL}[q_\phi(\boldsymbol{c}|\boldsymbol{s})||p(\boldsymbol{c})]). \tag{23}$$

The joint objective function of the semi-supervised learning is,

$$J = \sum_{\boldsymbol{s}\in\mathbb{U}} (\mathbb{E}_{q_\phi(\boldsymbol{c}|\boldsymbol{s})}[\log p_\theta(\boldsymbol{s}|\boldsymbol{c})] - \lambda D_{KL}[q_\phi(\boldsymbol{c}|\boldsymbol{s})||p(\boldsymbol{c})]) + \sum_{(\boldsymbol{c},\boldsymbol{s})\in\mathbb{L}} \log p_\phi(\boldsymbol{c}|\boldsymbol{s}). \tag{24}$$

Hence, the pointer network is trained on both unlabelled data, $\mathbb{U}$, and labelled data, $\mathbb{L}$, by a mixed criterion of REINFORCE and cross-entropy.

5 Related Work

As one of the typical sequence-to-sequence tasks, sentence-level summarisation has been explored by a series of discriminative encoder-decoder neural models. Filippova et al. (2015) carries out extractive summarisation via deletion with LSTMs, while Rush et al. (2015) applies a convolutional encoder and an

attentional feed-forward decoder to generate abstractive summarises, which provides the benchmark for the *Gigaword* dataset. Nallapati et al. (2016) further improves the performance by exploring multiple variants of RNN encoder-decoder models. The recent works Gulcehre et al. (2016), Nallapati et al. (2016) and Gu et al. (2016) also apply the similar idea of combining pointer networks and softmax output. However, different from all these discriminative models above, we explore generative models for sentence compression. Instead of training the discriminative model on a big labelled dataset, our original intuition of introducing a combined pointer networks is to bridge the unsupervised generative model (ASC) and supervised model (FSC) so that we could utilise a large additional dataset, either labelled or unlabelled, to boost the compression performance. Dai and Le (2015) also explored semi-supervised sequence learning, but in a pure deterministic model focused on learning better vector representations.

Recently variational auto-encoders have been applied in a variety of fields as deep generative models. In computer vision Kingma and Welling (2014), Rezende et al. (2014), and Gregor et al. (2015) have demonstrated strong performance on the task of image generation and Eslami et al. (2016) proposed variable-sized variational auto-encoders to identify multiple objects in images. While in natural language processing, there are variants of VAEs on modelling documents (Miao et al., 2016), sentences (Bowman et al., 2015) and discovery of relations (Marcheggiani and Titov, 2016). Apart from the typical initiations of VAEs, there are also a series of works that employs generative models for supervised learning tasks. For instance, Ba et al. (2015) learns visual attention for multiple objects by optimising a variational lower bound, Kingma et al. (2014) implements a semi-supervised framework for image classification and Miao et al. (2016) applies a conditional variational approximation in the task of factoid question answering. Dyer et al. (2016) proposes a generative model that explicitly extracts syntactic relationships among words and phrases which further supports the argument that generative models can be a statistically efficient method for learning neural networks from small data.

6 Experiments

6.1 Dataset & Setup

We evaluate the proposed models on the standard *Gigaword*[3] sentence compression dataset. This dataset was generated by pairing the headline of each article with its first sentence to create a source-compression pair. Rush et al. (2015) provided scripts[4] to filter out outliers, resulting in roughly 3.8M training pairs, a 400K validation set, and a 400K test set. In the following experiments all models are trained on the training set with different data sizes[5] and tested on a 2K subset, which is identical to the test set used by Rush et al. (2015) and Nallapati et al. (2016). We decode the sentences by $k = 5$ Beam search and test with full-length Rouge score.

For the ASC and FSC models, we use 256 for the dimension of both hidden units and lookup tables. In the ASC model, we apply a 3-layer bidirectional RNN with skip connections as the encoder, a 3-layer RNN pointer network with skip connections as the compressor, and a 1-layer vanilla RNN with soft attention as the decoder. The language model prior is trained on the article sentences of the full training set using a 3-layer vanilla RNN with 0.5 dropout. To lower the computational cost, we apply different vocabulary sizes for encoder and compressor (119,506 and 68,897) which corresponds to the settings of Rush et al. (2015). Specifically, the vocabulary of the decoder is filtered by taking the most frequent 10,000 words from the vocabulary of the encoder, where the rest of the words are tagged as '*<unk>*'. In further consideration of efficiency, we use only one sample for the gradient estimator. We optimise the model by Adam (Kingma and Ba, 2015) with a 0.0002 learning rate and 64 sentences per batch. The model converges in 5 epochs. Except for the pretrained language model, we do not use dropout or embedding initialisation for ASC and FSC models.

6.2 Extractive Summarisation

The first set of experiments evaluate the models on extractive summarisation. Here, we denote the joint

[3] https://catalog.ldc.upenn.edu/LDC2012T21

[4] https://github.com/facebook/NAMAS

[5] The hyperparameters where tuned on the validation set to maximise the perplexity of the summaries rather than the reconstructed source sentences.

Model	Training Data		Recall			Precision			F-1		
	Labelled	Unlabelled	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
FSC	500K	-	**30.817**	**10.861**	**28.263**	22.357	7.998	20.520	23.415	8.156	21.468
ASC+FSC$_1$	500K	500K	29.117	10.643	26.811	28.558	10.575	26.344	26.987	9.741	24.874
ASC+FSC$_2$	500K	3.8M	28.236	10.359	26.218	**30.112**	**11.131**	**27.896**	**27.453**	**9.902**	**25.452**
FSC	1M	-	**30.889**	**11.645**	**28.257**	27.169	10.266	24.916	26.984	10.028	24.711
ASC+FSC$_1$	1M	1M	30.490	11.443	28.097	28.109	10.799	25.943	27.258	10.189	25.148
ASC+FSC$_2$	1M	3.8M	29.034	10.780	26.801	**31.037**	**11.521**	**28.658**	**28.336**	**10.313**	**26.145**
FSC	3.8M	-	**30.112**	12.436	**27.889**	34.135	13.813	31.704	30.225	12.258	28.035
ASC+FSC$_1$	3.8M	3.8M	29.946	**12.558**	27.805	**35.538**	**14.699**	**32.972**	**30.568**	**12.553**	**28.366**

Table 1: Extractive Summarisation Performance. (1) The extractive summaries of these models are decoded by the pointer network (i.e the shared component of the ASC and FSC models). (2) R-1, R-2 and R-L represent the Rouge-1, Rouge-2 and Rouge-L score respectively.

models by ASC+FSC$_1$ and ASC+FSC$_2$ where ASC is trained on unlabelled data and FSC is trained on labelled data. The ASC+FSC$_1$ model employs equivalent sized labelled and unlabelled datasets, where the article sentences of the unlabelled data are the same article sentences in the labelled data, so there is no additional unlabelled data applied in this case. The ASC+FSC$_2$ model employs the full unlabelled dataset in addition to the existing labelled dataset, which is the true semi-supervised setting.

Table 1 presents the test Rouge score on extractive compression. We can see that the ASC+FSC$_1$ model achieves significant improvements on F-1 scores when compared to the supervised FSC model only trained on labelled data. Moreover, fixing the labelled data size, the ASC+FSC$_2$ model achieves better performance by using additional unlabelled data than the ASC+FSC$_1$ model, which means the semi-supervised learning works in this scenario. Interestingly, learning on the unlabelled data largely increases the precisions (though the recalls do not benefit from it) which leads to significant improvements on the F-1 Rouge scores. And surprisingly, the extractive ASC+FSC$_1$ model trained on full labelled data outperforms the abstractive NABS (Rush et al., 2015) baseline model (in Table 4).

6.3 Abstractive Summarisation

The second set of experiments evaluate performance on abstractive summarisation (Table 2). Consistently, we see that adding the generative objective to the discriminative model (ASC+FSC$_1$) results in a significant boost on all the Rouge scores, while employing extra unlabelled data increase performance further (ASC+FSC$_2$). This validates the effectiveness of transferring the knowledge learned on unlabelled data to the supervised abstractive summarisation.

In Figure 3, we present the validation perplexity to compare the abilities of the three models to learn the compression languages. The ASC+FSC$_1$(red) employs the same dataset for unlabelled and labelled training, while the ASC+FSC$_2$(black) employs the full unlabelled dataset. Here, the joint ASC+FSC$_1$ model obtains better perplexities than the single discriminative FSC model, but there is not much difference between ASC+FSC$_1$ and ASC+FSC$_2$ when the size of the labelled dataset grows. From the perspective of language modelling, the generative ASC model indeed helps the discriminative model learn to generate good summary sentences. Table 3 displays the validation perplexities of the benchmark models, where the joint ASC+FSC$_1$ model trained on the full labelled and unlabelled datasets performs the best on modelling compression languages.

Table 4 compares the test Rouge score on abstractive summarisation. Encouragingly, the semi-supervised model ASC+FSC$_2$ outperforms the baseline model NABS when trained on 500K supervised pairs, which is only about an eighth of the supervised data. In Nallapati et al. (2016), the authors exploit the full limits of discriminative RNN encoder-decoder models by incorporating a sampled softmax, expanded vocabulary, additional lexical features, and combined pointer networks[6], which yields the best performance listed in Table 4. However, when all the data is employed with the mixed ob-

[6]The idea of the combined pointer networks is similar to the FSC model, but the implementations are slightly different.

Model	Training Data		Recall			Precision			F-1		
	Labelled	Unlabelled	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
FSC	500K	-	27.147	10.039	25.197	33.781	13.019	31.288	29.074	10.842	26.955
ASC+FSC$_1$	500K	500K	27.067	10.717	25.239	33.893	13.678	31.585	29.027	11.461	27.072
ASC+FSC$_2$	500K	3.8M	**27.662**	**11.102**	**25.703**	**35.756**	**14.537**	**33.212**	**30.140**	**12.051**	**27.99**
FSC	1M	-	28.521	11.308	26.478	33.132	13.422	30.741	29.580	11.807	27.439
ASC+FSC$_1$	1M	1M	28.333	11.814	26.367	35.860	**15.243**	33.306	30.569	12.743	28.431
ASC+FSC$_2$	1M	3.8M	**29.017**	**12.007**	**27.067**	**36.128**	14.988	**33.626**	**31.089**	**12.785**	**28.967**
FSC	3.8M	-	31.148	13.553	28.954	36.917	16.127	34.405	32.327	14.000	30.087
ASC+FSC$_1$	3.8M	3.8M	**32.385**	**15.155**	**30.246**	**39.224**	**18.382**	**36.662**	**34.156**	**15.935**	**31.915**

Table 2: Abstractive Summarisation Performance. The abstractive summaries of these models are decoded by the combined pointer network (i.e. the shared pointer network together with the softmax output layer over the full vocabulary).

Model	Labelled Data	Perplexity
Bag-of-Word (BoW)	3.8M	43.6
Convolutional (TDNN)	3.8M	35.9
Attention-Based (NABS) (Rush et al., 2015)	3.8M	27.1
Forced-Attention (FSC)	3.8M	18.6
Auto-encoding (ASC+FSC$_1$)	3.8M	16.6

Table 3: Comparison on validation perplexity. BoW, TDNN and NABS are the baseline neural compression models with different encoders in Rush et al. (2015)

Model	Labelled Data	R-1	R-2	R-L
(Rush et al., 2015)	3.8M	29.78	11.89	26.97
(Nallapati et al., 2016)	3.8M	33.17	**16.02**	30.98
ASC + FSC$_2$	500K	30.14	12.05	27.99
ASC + FSC$_2$	1M	31.09	12.79	28.97
ASC + FSC$_1$	3.8M	**34.17**	15.94	**31.92**

Table 4: Comparison on test Rouge scores

jective ASC+FSC$_1$ model, the result is significantly better than this previous state-of-the-art. As the semi-supervised ASC+FSC$_2$ model can be trained on unlimited unlabelled data, there is still significant space left for further performance improvements.

Table 5 presents the examples of the compression sentences decoded by the joint model ASC+FSC$_1$ and the FSC model trained on the full dataset.

7 Discussion

From the perspective of generative models, a significant contribution of our work is a process for reducing variance for discrete sampling-based variational inference. The first step is to introduce two baselines in the control variates method due to the fact that the reparameterisation trick is not applicable for discrete latent variables. However it is the second step of using a pointer network as the biased estimator that makes the key contribution. This results in a much smaller state space, bounded by the length of the source sentence (mostly between 20 and 50 tokens), compared to the full vocabulary. The final step is to apply the FSC model to transfer the knowledge learned from the supervised data to the pointer network. This further reduces the sampling variance by acting as a sort of bootstrap or constraint on the unsupervised latent space which could encode almost anything but which thus becomes biased towards matching the supervised distribution. By using these variance reduction methods, the ASC model is able to carry out effective variational inference for the latent language model so that it learns to summarise the sentences from the large unlabelled training data.

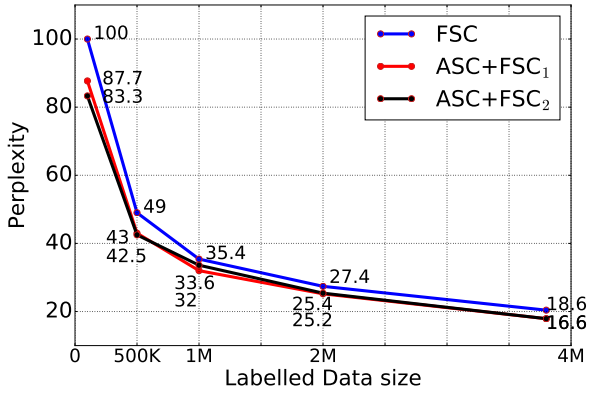


Figure 3: Perplexity on validation dataset.

In a different vein, according to the reinforcement learning interpretation of sequence level training (Ranzato et al., 2016), the compression model of the ASC model acts as an agent which iteratively generates words (takes actions) to compose the com-

pression sentence and the reconstruction model acts as the reward function evaluating the quality of the compressed sentence which is provided as a reward signal. Ranzato et al. (2016) presents a thorough empirical evaluation on three different NLP tasks by using additional sequence-level reward (BLEU and Rouge-2) to train the models. In the context of this paper, we apply a variational lower bound (mixed reconstruction error and KL divergence regularisation) instead of the explicit Rouge score. Thus the ASC model is granted the ability to explore unlimited unlabelled data resources. In addition we introduce a supervised FSC model to teach the compression model to generate stable sequences instead of starting with a random policy. In this case, the pointer network that bridges the supervised and unsupervised model is trained by a mixed criterion of REINFORCE and cross-entropy in an incremental learning framework. Eventually, according to the experimental results, the joint ASC and FSC model is able to learn a robust compression model by exploring both labelled and unlabelled data, which outperforms the other single discriminative compression models that are only trained by cross-entropy reward signal.

8 Conclusion

In this paper we have introduced a generative model for jointly modelling pairs of sequences and evaluated its efficacy on the task of sentence compression. The variational auto-encoding framework provided an effective inference algorithm for this approach and also allowed us to explore combinations of discriminative (FSC) and generative (ASC) compression models. The evaluation results show that supervised training of the combination of these models improves upon the state-of-the-art performance for the Gigaword compression dataset. When we train the supervised FSC model on a small amount of labelled data and the unsupervised ASC model on a large set of unlabelled data the combined model is able to outperform previously reported benchmarks trained on a great deal more supervised data. These results demonstrate that we are able to model language as a discrete latent variable in a variational auto-encoding framework and that the resultant generative model is able to effectively exploit both supervised and unsupervised data in sequence-to-sequence tasks.

src	the sri lankan government on wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country .
ref	sri lanka closes schools as war escalates
asc$_a$	sri lanka closes government schools
asc$_e$	sri lankan government closure schools escalated
fsc$_a$	sri lankan government closure with tamil rebels closure
src	factory orders for manufactured goods rose #.# percent in september , the commerce department said here thursday .
ref	us september factory orders up #.# percent
asc$_a$	us factory orders up #.# percent in september
asc$_e$	factory orders rose #.# percent in september
fsc$_a$	factory orders #.# percent in september
src	hong kong signed a breakthrough air services agreement with the united states on friday that will allow us airlines to carry freight to asian destinations via the territory .
ref	hong kong us sign breakthrough aviation pact
asc$_a$	us hong kong sign air services agreement
asc$_e$	hong kong signed air services agreement with united states
fsc$_a$	hong kong signed air services pact with united states
src	a swedish un soldier in bosnia was shot and killed by a stray bullet on tuesday in an incident authorities are calling an accident , military officials in stockholm said tuesday .
ref	swedish un soldier in bosnia killed by stray bullet
asc$_a$	swedish un soldier killed in bosnia
asc$_e$	swedish un soldier shot and killed
fsc$_a$	swedish soldier shot and killed in bosnia
src	tea scores on the fourth day of the second test between australia and pakistan here monday .
ref	australia vs pakistan tea scorecard
asc$_a$	australia v pakistan tea scores
asc$_e$	australia tea scores
fsc$_a$	tea scores on #th day of #nd test
src	india won the toss and chose to bat on the opening day in the opening test against west indies at the antigua recreation ground on friday .
ref	india win toss and elect to bat in first test
asc$_a$	india win toss and bat against west indies
asc$_e$	india won toss on opening day against west indies
fsc$_a$	india chose to bat on opening day against west indies
src	a powerful bomb exploded outside a navy base near the sri lankan capital colombo tuesday , seriously wounding at least one person , military officials said .
ref	bomb attack outside srilanka navy base
asc$_a$	bomb explodes outside sri lanka navy base
asc$_e$	bomb outside sri lankan navy base wounding one
fsc$_a$	bomb exploded outside sri lankan navy base
src	press freedom in algeria remains at risk despite the release on wednesday of prominent newspaper editor mohamed <unk> after a two-year prison sentence , human rights organizations said .
ref	algerian press freedom at risk despite editor 's release <unk> picture
asc$_a$	algeria press freedom remains at risk
asc$_e$	algeria press freedom remains at risk
fsc$_a$	press freedom in algeria at risk

Table 5: Examples of the compression sentences. **src** and **ref** are the source and reference sentences provided in the test set. **asc**$_a$ and **asc**$_e$ are the abstractive and extractive compression sentences decoded by the joint model ASC+FSC$_1$, and **fsc**$_a$ denotes the abstractive compression obtained by the FSC model.

References

[Ba et al.2015] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of ICLR*.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

[Bowman et al.2015] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

[Chorowski et al.2015] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of NIPS*, pages 577–585.

[Dai and Le2015] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proceedings of NIPS*, pages 3061–3069.

[Dyer et al.2016] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*.

[Eslami et al.2016] SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, Koray Kavukcuoglu, and Geoffrey E Hinton. 2016. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*.

[Filippova et al.2015] Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of EMNLP*, pages 360–368.

[Gregor et al.2015] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In *Proceedings of ICML*.

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

[Gulcehre et al.2016] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

[Hinton and Salakhutdinov2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*.

[Kingma and Ba2015] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

[Kingma and Welling2014] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of ICLR*.

[Kingma et al.2014] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proceedings of NIPS*.

[Marcheggiani and Titov2016] Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4.

[Miao et al.2016] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML*.

[Mnih and Gregor2014] Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of ICML*.

[Mnih et al.2014] Volodymyr Mnih, Nicolas Heess, and Alex Graves. 2014. Recurrent models of visual attention. In *Proceedings of NIPS*.

[Nallapati et al.2016] Ramesh Nallapati, Bowen Zhou, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

[Ranzato et al.2016] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks.

[Rezende et al.2014] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML*.

[Rumelhart et al.1985] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

[Rush et al.2015] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

[Vinyals et al.2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*, pages 2674–2682.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention.

Globally Coherent Text Generation with Neural Checklist Models

Chloé Kiddon Luke Zettlemoyer Yejin Choi
Computer Science & Engineering
University of Washington
{chloe, lsz, yejin}@cs.washington.edu

Abstract

Recurrent neural networks can generate locally coherent text but often have difficulties representing what has already been generated and what still needs to be said – especially when constructing long texts. We present the *neural checklist model*, a recurrent neural network that models global coherence by storing and updating an agenda of text strings which should be mentioned somewhere in the output. The model generates output by dynamically adjusting the interpolation among a language model and a pair of attention models that encourage references to agenda items. Evaluations on cooking recipes and dialogue system responses demonstrate high coherence with greatly improved semantic coverage of the agenda.

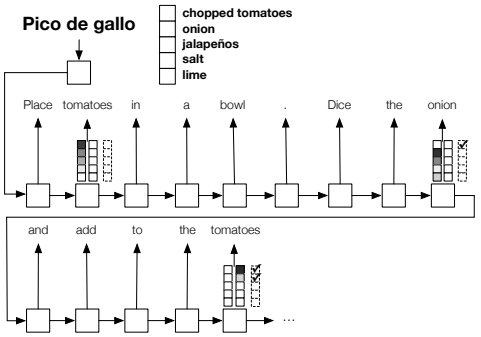


Figure 1: Example checklist recipe generation. A checklist (right dashed column) tracks which agenda items (top boxes; "salt," "lime," etc.) have already been used (checked boxes). The model is trained to interpolate an RNN (e.g., encode "pico de gallo" and decode a recipe) with attention models over new (left column) and used (middle column) items that identify likely items for each time step (shaded boxes; "tomatoes," etc.).

1 Introduction

Recurrent neural network (RNN) architectures have proven to be well suited for many natural language generation tasks (Mikolov et al., 2010; Mikolov et al., 2011; Sordoni et al., 2015; Xu et al., 2015; Wen et al., 2015; Mei et al., 2016). Previous neural generation models typically generate locally coherent language that is on topic; however, overall they can miss information that should have been introduced or introduce duplicated or superfluous content. These errors are particularly common in situations where there are multiple distinct sources of input or the length of the output text is sufficiently long. In this paper, we present a new recurrent neural model that maintains coherence while improving coverage by globally tracking what has been said and what is still left to be said in complete texts.

For example, consider the challenge of generating a cooking recipe, where the title and ingredient list are provided as inputs and the system must generate a complete text that describes how to produce the desired dish. Existing RNN models may lose track of which ingredients have already been mentioned, especially during the generation of a long recipe with many ingredients. Recent work has focused on adapting neural network architectures to improve coverage (Wen et al., 2015) with application to generating customer service responses, such as hotel information, where a single sentence is generated to describe a few key ideas. Our focus is instead on developing a model that maintains coherence while producing longer texts or covering longer

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 329–339,
Austin, Texas, November 1-5, 2016.

input specifications (e.g., a long ingredient list).

More specifically, our *neural checklist model* generates a natural language description for achieving a goal, such as generating a recipe for a particular dish, while using a new checklist mechanism to keep track of an agenda of items that should be mentioned, such as a list of ingredients (see Fig. 1). The checklist model learns to interpolate among three components at each time step: (1) an encoder-decoder language model that generates goal-oriented text, (2) an attention model that tracks remaining agenda items that need to be introduced, and (3) an attention model that tracks the used, or checked, agenda items. Together, these components allow the model to learn representations that best predict which words should be included in the text and when references to agenda items should be checked off the list (see check marks in Fig. 1).

We evaluate our approach on a new cooking recipe generation task and the dialogue act generation from Wen et al. (2015). In both cases, the model must correctly describe a list of agenda items: an ingredient list or a set of facts, respectively. Generating recipes additionally tests the ability to maintain coherence in long procedural texts. Experiments in dialogue generation demonstrate that our approach outperforms previous work with up to a 4 point BLEU improvement. Our model also scales to cooking recipes, where both automated and manual evaluations demonstrate that it maintains the strong local coherence of baseline RNN techniques while significantly improving the global coverage by effectively integrating the agenda items.

2 Task

Given a goal $\mathbf{g}$ and an agenda $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_{|E|}\}$, our task is to generate a goal-oriented text $\mathbf{x}$ by making use of items on the agenda. For example, in the cooking recipe domain, the goal is the recipe title ("pico de gallo" in Fig. 1), and the agenda is the ingredient list (e.g., "lime," "salt"). For dialogue systems, the goal is the dialogue type (e.g., inform or query) and the agenda contains information to be mentioned (e.g., a hotel name and address). For example, if $\mathbf{g}$ ="inform" and $E = \{$name(Hotel Stratford), has_internet(no)$\}$, an output text might be $\mathbf{x}$ ="Hotel Stratford does not have internet."

3 Related Work

Attention models have been used for many NLP tasks such as machine translation (Balasubramanian et al., 2013; Bahdanau et al., 2014), abstractive sentence summarization (Rush et al., 2015), machine reading (Cheng et al., 2016), and image caption generation (Xu et al., 2015). Our model uses new types of attention to record what has been said and to select new agenda items to be referenced.

Recently, other researchers have developed new ways to use attention mechanisms for related generation challenges. Most closely related, Wen et al. (2015) and Wen et al. (2016) present neural network models for generating dialogue system responses given a set of agenda items. They focus on generating short texts (1-2 sentences) in a relatively small vocabulary setting and assume a fixed set of possible agenda items. Our model composes substantially longer texts, such as recipes, with a more varied and open ended set of possible agenda items. We also compare performance for our model on their data.

Maintaining coherence and avoiding duplication have been recurring challenges when generating text using RNNs for other applications, including image captioning (Jia et al., 2015; Xu et al., 2015) and machine translation (Tu et al., 2016b; Tu et al., 2016a). A variety of solutions have been developed to address infrequent or out-of-vocabulary words in particular (Gülçehre et al., 2016; Jia and Liang, 2016). Instead of directly copying input words or deterministically selecting output, our model can learn how to generate them (e.g., it might prefer to produce the word "steaks" when the original recipe ingredient was "ribeyes"). Finally, recent work in machine translation models has introduced new training objectives to encourage attention to all input words (Luong et al., 2015), but these models do not accumulate attention while decoding.

Generating recipes was an early task in planning (Hammond, 1986) and generating referring expression research (Dale, 1988). These can be seen as key steps in classic approaches to generating natural language text: a formal meaning representation is provided as input and the model first does content selection to determine the non-linguistic concepts to be conveyed by the output text (i.e., *what to say*) and then does realization to describe those concepts

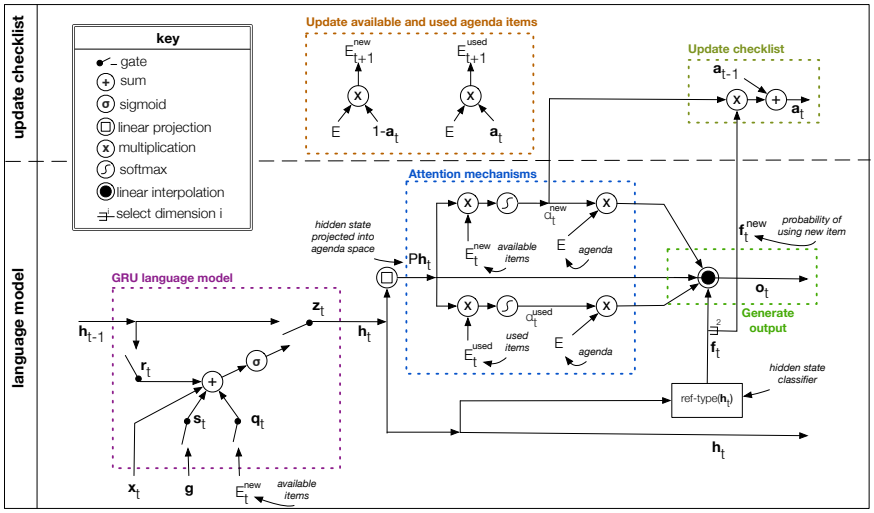


Figure 2: A diagram of the neural checklist model. The bottom portion depicts how the model generates the output embedding $\mathbf{o}_t$. The top portion shows how the checklist and available/used agenda item matrices are updated.

in natural language text (i.e., *how to say it*) (Thompson, 1977; Reiter and Dale, 2000). More recently, machine learning methods have focused on parts of this approach (Barzilay and Lapata, 2005; Liang et al., 2009) or the full two-stage approach (Angeli et al., 2010; Konstas and Lapata, 2013). Most of these models shorter texts, although Mori et al. (2014) did consider longer cooking recipes. Our approach is a joint model that instead operates with textual input and tries to cover all of the content it is given.

4 Model

Fig. 2 shows a graphical representation of the neural checklist model. At a high level, our model uses a recurrent neural network (RNN) language model that encodes the goal as a bag-of-words and then generates output text token by token. It additionally stores a vector that acts as a soft checklist of what agenda items have been used so far during generation. This checklist is updated every time an agenda item reference is generated and is used to compute the available agenda items at each time step. The available items are used as an input to the language model and to constrain which agenda items can still be referenced during generation. Agenda embeddings are also used when generating item references.

4.1 Input variable definitions

We assume the goal $\mathbf{g}$ and agenda items E (see Sec. 2) are each defined by a set of tokens. Goal tokens come from a fixed vocabulary $\mathcal{V}_{goal}$, the item tokens come from a fixed vocabulary $\mathcal{V}_{agenda}$, and the tokens of the text $\mathbf{x}_t$ come from a fixed vocabulary $\mathcal{V}_{text}$. In an abuse of notation, we represent each goal $\mathbf{g}$, agenda item $\mathbf{e}_i$, and text token $\mathbf{x}_t$ as a k-dimensional word embedding vector. We compute these embeddings by creating indicator vectors of the vocabulary token (or set of tokens for goals and agenda items) and embed those vectors using a trained $k \times |\mathcal{V}_z|$ projection matrix, where $z \in \{goal, agenda, text\}$ depending whether we are generating a goal, agenda item, or text token.

Given a goal embedding $\mathbf{g} \in \mathbb{R}^k$, a matrix of L agenda items $E \in \mathbb{R}^{L \times k}$, a checklist soft record of what items have been used $\mathbf{a}_{t-1} \in \mathbb{R}^L$, a previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^k$, and the current input word embedding $\mathbf{x}_t \in \mathbb{R}^k$, our architecture computes the next hidden state $\mathbf{h}_t$, an embedding used to generate the output word $\mathbf{o}_t$, and the updated checklist $\mathbf{a}_t$.

4.2 Generating output token probabilities

To generate the output token probability distribution (see "Generate output" box in Fig. 2), $\mathbf{w}_t \in \mathbb{R}^{|V_{text}|}$, we project the *output hidden state* $\mathbf{o}_t$ into the vocabulary space and apply a softmax:

$$\mathbf{w}_t = \text{softmax}(W_o \mathbf{o}_t),$$

where $W_o \in \mathbb{R}^{|V| \times k}$ is a trained projection matrix. The output hidden state is the linear interpolation of (1) content $\mathbf{c}_t^{gru}$ from a Gated Recurrent Unit

(GRU) language model, (2) an encoding $\mathbf{c}_t^{new}$ generated from the new agenda item reference model (Sec. 4.3), and (3) and an encoding $\mathbf{c}_t^{used}$ generated from a previously used item model (Sec. 4.4):

$$\mathbf{o}_t = f_t^{gru}\mathbf{c}_t^{gru} + f_t^{new}\mathbf{c}_t^{new} + f_t^{used}\mathbf{c}_t^{used}.$$

The interpolation weights, f_t^{gru}, f_t^{new}, and f_t^{used}, are probabilities representing how much the output token should reflect the current state of the language model or a chosen agenda item. f_t^{gru} is the probability of a non-agenda-item token, f_t^{new} is the probability of an new item reference token, and f_t^{used} is the probability of a used item reference. In the Fig. 1 example, f_t^{new} is high in the first row when new ingredient references "tomatoes" and "onion" are generated; f_t^{used} is high when the reference back to "tomatoes" is made in the second row, and f_t^{gru} is high the rest of the time.

To generate these weights, our model uses a three-way probabilistic classifier, $ref\text{-}type(\mathbf{h}_t)$, to determine whether the hidden state of the GRU $\mathbf{h}_t$ will generate non-agenda tokens, new agenda item references, or used item references. $ref\text{-}type(\mathbf{h}_t)$ generates a probability distribution $\mathbf{f}_t \in \mathbb{R}^3$ as

$$\mathbf{f}_t = ref\text{-}type(\mathbf{h}_t) = softmax(\beta S\mathbf{h}_t),$$

where $S \in \mathbb{R}^{3\times k}$ is a trained projection matrix and β is a temperature hyper-parameter. $f_t^{gru} = \mathbf{f}_t^1$, $f_t^{new} = \mathbf{f}_t^2$, and $f_t^{used} = \mathbf{f}_t^3$. $ref\text{-}type()$ does not use the agenda, only the hidden state $\mathbf{h}_t$: $\mathbf{h}_t$ must encode when to use the agenda, and $ref\text{-}type()$ is trained to identify that in $\mathbf{h}_t$.

4.3 New agenda item reference model

The two key features of our model are that it (1) predicts which agenda item is being referred to, if any, at each time step and (2) stores those predictions for use during generation. These components allow for improved output texts that are more likely to mention agenda items while avoiding repetition and references to irrelevant items not in the agenda.

These features are enabled by a *checklist vector* $\mathbf{a}_t \in \mathbb{R}^L$ that represents the probability each agenda item has been introduced into the text. The checklist vector is initialized to all zeros at $t = 1$, representing that all items have yet to be introduced. The checklist vector is a soft record with each $\mathbf{a}_{t,i} \in [0, 1]$.[1]

We introduce the remaining items as a matrix $E_t^{new} \in \mathbb{R}^{L\times k}$, where each row is an agenda item embedding weighted by how likely it is to still need to be referenced. For example, in Fig. 1, after the first "tomatoes" is generated, the row representing "chopped tomatoes" in the agenda will be weighted close to 0. We calculate E_t^{new} using the checklist vector (see "Update [...] items" box in Fig. 2):

$$E_t^{new} = ((\mathbf{1}_L - \mathbf{a}_{t-1}) \otimes \mathbf{1}_k) \circ E,$$

where $\mathbf{1}_L = \{1\}^L$, $\mathbf{1}_k = \{1\}^k$, and the outer product $\otimes$ replicates $\mathbf{1}_L - \mathbf{a}_{t-1}$ for each dimension of the embedding space. $\circ$ is the Hadamard product (i.e., element-wise multiplication) of two matrices with the same dimensions.

The model predicts when an agenda item will be generated using $ref\text{-}type()$ (see Sec. 4.2 for details). When it does, the encoding $\mathbf{c}_t^{new}$ approximates which agenda item is most likely. $\mathbf{c}_t^{new}$ is computed using an attention model that generates a learned soft alignment $\boldsymbol{\alpha}_t^{new} \in \mathbb{R}^L$ between the hidden state $\mathbf{h_t}$ and the rows of E_t^{new} (i.e., available items). The alignment is a probability distribution representing how close $\mathbf{h}_t$ is to each item:

$$\boldsymbol{\alpha}_t^{new} \propto \exp(\gamma E_t^{new} P\mathbf{h}_t),$$

where $P \in \mathbb{R}^{k\times k}$ is a learned projection matrix and γ is a temperature hyper-parameter. In Fig. 1, the shaded squares in the top line (i.e., the first "tomatoes" and the onion references) represent this alignment. The attention encoding $\mathbf{c}_t^{new}$ is then the attention-weighted sum of the agenda items:

$$\mathbf{c}_t^{new} = E^T\boldsymbol{\alpha}_t^{new}.$$

At each step, the model updates the checklist vector based on the probability of generating a new agenda item reference, $\mathbf{f}_t^{new}$, and the attention alignment $\boldsymbol{\alpha}_t^{new}$. We calculate the update to checklist, $\mathbf{a}_t^{new}$, as $\mathbf{a}_t^{new} = \mathbf{f}_t^{new} \cdot \boldsymbol{\alpha}_t^{new}$. Then, the new checklist $\mathbf{a}_t$ is $\mathbf{a}_t = \mathbf{a}_{t-1} + \mathbf{a}_t^{new}$.

[1] By definition, $\mathbf{a}_t$ is non-negative. We truncate any values greater than 1 using a hard tanh function.

4.4 Previously used item reference model

We also allow references to be generated for previously used agenda items through the previously used item encoding $\mathbf{c}_t^{used}$. This is useful in longer texts – when agenda items can be referred to more than once – so that the agenda is always responsible for generating its own referring expressions. The example in Fig. 1 refers back to tomatoes when generating to what to add the diced onion.

At each time step t, we use a second attention model to compare $\mathbf{h}_t$ to a used items matrix $E_t^{used} \in \mathbb{R}^{L \times k}$. Like the remaining agenda item matrix E_t^{new}, E_t^{used} is calculated using the checklist vector generated at the previous time step:

$$E_t^{used} = (\mathbf{a}_{t-1} \otimes \mathbf{1}_k) \circ E.$$

The attention over the used items, $\boldsymbol{\alpha}_t^{used} \in \mathbb{R}^L$, and the used attention encoding $\mathbf{c}_t^{used}$ are calculated in the same way as those over the available items (see Sec. 4.3 for comparison):

$$\boldsymbol{\alpha}_t^{used} \propto \exp(\gamma E_t^{used} P \mathbf{h}_t),$$
$$\mathbf{c}_t^{used} = E^T \boldsymbol{\alpha}_t^{used}.$$

4.5 GRU language model

Our decoder RNN adapts a Gated Recurrent Unit (GRU) (Cho et al., 2014). Given an input $\mathbf{x}_t \in \mathbb{R}^k$ at time step t and the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^k$, a GRU computes the next hidden state $\mathbf{h}_t$ as

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t.$$

The *update gate*, $\mathbf{z}_t$, interpolates between $\mathbf{h}_{t-1}$ and new content, $\tilde{\mathbf{h}}_t$, defined respectively as

$$\mathbf{z}_t = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1}),$$
$$\tilde{\mathbf{h}}_t = \tanh(W \mathbf{x}_t + \mathbf{r}_t \odot U \mathbf{h}_{t-1}).$$

$\odot$ is an element-wise multiplication, and the *reset gate*, $\mathbf{r}_t$, is calculated as

$$\mathbf{r}_t = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1}).$$

W_z, U_z, W, U, W_r, $U_r \in \mathbb{R}^{k \times k}$ are trained projection matrices.

We adapted a GRU to allow extra inputs, namely the goal $\mathbf{g}$ and the available agenda items E_t^{new} (see "GRU language model" box in Fig. 2). These extra inputs help guide the language model stay on topic. Our adapted GRU has a change to the computation of the new content $\tilde{\mathbf{h}}_t$ as follows:

$$\begin{aligned}\tilde{\mathbf{h}}_t \;=\; & \tanh(W_h x_t + \mathbf{r}_t \odot U_h \mathbf{h}_{t-1} \\ & + \mathbf{s}_t \odot Y \mathbf{g} + \mathbf{q}_t \odot (\mathbf{1}_L^T Z E_t^{new})^T,\end{aligned}$$

where $\mathbf{s}_t$ is a *goal select* gate and $\mathbf{q}_t$ is a *item select* gate, respectively defined as

$$\mathbf{s}_t = \sigma(W_s \mathbf{x}_t + U_s \mathbf{h}_{t-1}),$$
$$\mathbf{q}_t = \sigma(W_q \mathbf{x}_t + U_q \mathbf{h}_{t-1}).$$

$\mathbf{1}_L$ sums the rows of the available item matrix E_t^{new}. Y, Z, W_s, U_s, W_q, $U_q \in \mathbb{R}^{k \times k}$ are trained projection matrices. The goal select gate controls when the goal should be taken into account during generation: for example, the recipe title may be used to decide what the imperative verb for a new step should be. The item select gate controls when the available agenda items should be taken into account (e.g., when generating a list of ingredients to combine). The GRU hidden state is initialized with a projection of the goal: $\mathbf{h}_0 = U_g \mathbf{g}$, where $U_g \in \mathbb{R}^{k \times k}$.

The content vector $\mathbf{c}_t^{gru}$ that is used to compute the output hidden state $\mathbf{o}_t$ is a linear projection of the GRU hidden state, $\mathbf{c}_t^{gru} = P\mathbf{h}_t$, where P is the same learned projection matrix used in the computation of the attention weights (see Sections 4.3 and 4.4).

4.6 Training

Given a training set of (goal, agenda, output text) triples $\{(\mathbf{g}^{(1)}, E^{(1)}, \mathbf{x}^{(1)}), \ldots, (\mathbf{g}^{(J)}, E^{(J)}, \mathbf{x}^{(J)})\}$, we train model parameters by minimizing negative log-likelihood: $NLL(\theta) =$

$$-\sum_{j=1}^{J} \sum_{i=2}^{N_j} \log p(\mathbf{x}_i^{(j)} | \mathbf{x}_1^{(j)}, \ldots, \mathbf{x}_{i-1}^{(j)}, \mathbf{g}^{(j)}, E^{(j)}; \theta),$$

where $\mathbf{x}_1^{(j)}$ is the start symbol. We use mini-batch stochastic gradient descent, and back-propagate through the goal, agenda, and text embeddings.

It is sometimes the case that weak heuristic supervision on latent variables can be easily gathered to improve training. For example, for recipe generation, we can approximate the linear interpolation weights $\mathbf{f}_t$ and the attention updates $\mathbf{a}_t^{new}$ and $\mathbf{a}_t^{used}$ using string match heuristics comparing tokens in

the text to tokens in the ingredient list.[2] When this extra signal is available, we add mean squared loss terms to $NLL(\theta)$ to encourage the latent variables to take those values; for example, if $\mathbf{f}_t^*$ is the true value and $\mathbf{f}_t$ is the predicted value, a loss term $-(\mathbf{f}_t^* - \mathbf{f}_t)^2$ is added. When this signal is not available, as is the case with our dialogue generation task, we instead introduce a mean squared loss term that encourages the final checklist $\mathbf{a}_{N_j}^{(j)}$ to be a vector of 1s (i.e., every agenda item is accounted for).

4.7 Generation

We generate text using beam search, which has been shown to be fast and accurate for RNN decoding (Graves, 2012; Sutskever et al., 2014). When the beam search completes, we select the highest probability sequence that uses the most agenda items. This is the count of how many times the three-way classifier, $ref\text{-}type(\mathbf{h}_t)$, chose to generate an new item reference with high probability (i.e., $> 50\%$).

5 Experimental setup

Our model was implemented and trained using the Torch scientific computing framework for Lua.[3]

Experiments We evaluated neural checklist models on two natural language generation tasks. The first task is cooking recipe generation. Given a recipe title (i.e., the name of the dish) as the goal and the list of ingredients as the agenda, the system must generate the correct recipe text. Our second evaluation is based on the task from Wen et al. (2015) for generating dialogue responses for hotel and restaurant information systems. The task is to generate a natural language response given a query type (e.g., informing or querying) and a list of facts to convey (e.g., a hotel's name and address).

Parameters We constrain the gradient norm to 5.0 and initialize parameters uniformly on $[-0.35, 0.35]$. We used a beam of size 10 for generation. Based on dev set performance, a learning rate of 0.1 was chosen, and the temperature hyperparameters (β, γ) were $(5, 2)$ for the recipe task and $(1, 10)$ for the dialogue task. The models for the recipe task had a hidden state size of $k = 256$; the models for the dialogue task had $k = 80$ to compare to previous models. We use a batch size 30 for the recipe task and 10 for the dialogue task.

Recipe data and pre-processing We use the Now You're Cooking! recipe library: the data set contains over 150,000 recipes in the Meal-Master™ format.[4] We heuristically removed sentences that were not recipe steps (e.g., author notes, nutritional information, publication information). 82,590 recipes were used for training, and 1,000 each for development and testing. We filtered out recipes to avoid exact duplicates between training and dev (test) sets.

We collapsed multi-word ingredient names into single tokens using word2phrase[5] ran on the training data ingredient lists. Titles and ingredients were cleaned of non-word tokens. Ingredients additionally were stripped of amounts (e.g., "1 tsp"). As mentioned in Sec. 4.6, we approximate true values for the interpolation weights and attention updates for recipes based on string match between the recipe text and the ingredient list. The first ingredient reference in a sentence cannot be the first token or after a comma (e.g., the bold tokens cannot be ingredients in "**oil** the pan" and "in a large bowl, **mix** [...]").

Recipe data statistics Automatic recipe generation is difficult due to the length of recipes, the size of the vocabulary, and the variety of possible dishes. In our training data, the average recipe length is 102 tokens, and the longest recipe has 814 tokens. The vocabulary of the recipe text from the training data (i.e., the text of the recipe not including the title or ingredient list) has 14,103 unique tokens. About 31% of tokens in the recipe vocabulary occur at least 100 times in the training data; 8.6% of the tokens occur at least 1000 times. The training data also represents a wide variety of recipe types, defined by the recipe titles. Of 3793 title tokens, only 18.9% of the title tokens in the title vocabulary occur at least 100 times in the training data, which demonstrates the large variability in the titles.

Dialogue system data and processing We used the hotel and restaurant dialogue system corpus and the same train-development-test split from Wen et al. (2015). We used the same pre-processing, sets

[2] Similar to $\mathbf{a}_t^{new}$, $\mathbf{a}_t^{used} = \mathbf{f}_t^{used} \cdot \boldsymbol{\alpha}_t^{used}$.

[3] http://torch.ch/

[4] Recipes and format at http://www.ffts.com/recipes.htm

[5] See https://code.google.com/p/word2vec/

of reference samples, and baseline output, and we were given model output to compare against.[6] For training, slot values (e.g., "Red Door Cafe") were replaced by generic tokens (e.g., "NAME_TOKEN"). After generation, generic tokens were swapped back to specific slot values. Minor post-processing included removing duplicate determiners from the relexicalization and merging plural "-s" tokens onto their respective words. After replacing specific slot values with generic tokens, the training data vocabulary size of the hotel corpus is 445 tokens, and that of the restaurant corpus is 365 tokens. The task has eight goals (e.g., *inform*, *confirm*).

Models Our main baseline *EncDec* is a model using the RNN Encoder-Decoder framework proposed by Cho et al. (2014) and Sutskever et al. (2014). The model encodes the goal and then each agenda item in sequence and then decodes the text using GRUs. The encoder has two sets of parameters: one for the goal and the other for the agenda items. For the dialogue task, we also compare against the *SC-LSTM* system from Wen et al. (2015) and the handcrafted rule-based generator described in that paper.

For the recipe task, we also compare against three other baselines. The first is a basic attention model, *Attention*, that generates an attention encoding by comparing the hidden state $\mathbf{h}_t$ to the agenda. That encoding is added to the hidden state, and a nonlinear transformation is applied to the result before projecting into the output space. We also present a nearest neighbor baseline (*NN*) that simply copies over an existing recipe text based on the input similarity computed using cosine similarity over the title and the ingredient list. Finally, we present a hybrid approach (*NN-Swap*) that revises a nearest neighbor recipe using the neural checklist model. The neural checklist model is forced to generate the returned recipe nearly verbatim, except that it can generate new strings to replace any extraneous ingredients.

Our neural checklist model is labeled *Checklist*. We also present the *Checklist+* model, which interactively re-writes a recipe to better cover the input agenda: if the generated text does not use every agenda item, embeddings corresponding to missing items are multiplied by increasing weights and a new recipe is generated. This process repeats until the new recipe does not contain new items.

[6] We thank the authors for sharing their system outputs.

Model	BLEU-4	METEOR	Avg. % given items	Avg. extra items
Attention	2.8	8.6	22.8%	3.0
EncDec	3.1	9.4	26.9%	2.0
NN	**7.1**	12.1	40.0%	4.2
NN-Swap	**7.1**	**12.8**	58.2%	2.1
Checklist	3.0	10.3	67.9%	**0.6**
- $\mathbf{o}_t = \mathbf{h}_t$	2.1	8.3	29.1%	2.4
- no used	3.0	10.4	62.2%	1.9
- no supervision	3.7	10.1	38.9%	1.8
Checklist+	3.8	11.5	**83.4%**	0.8

Table 1: Quantitative results on the recipe task. The line with $\mathbf{o}_t = \mathbf{h}_t$ has the results for the non-interpolation ablation.

We also report the performance of our checklist model without the additional weak supervision of heuristic ingredient references (- *no supervision*) (see Sec. 4.6).[7] we also evaluate two ablations of our checklist model on the recipe task. First, we remove the linear interpolation and instead use $\mathbf{h}_t$ as the output (see Sec. 4.2). Second, we remove the previously used item reference model by changing $ref\text{-}type()$ to a 2-way classifier between new ingredient references and all other tokens (see Sec. 4.4).

Metrics We include commonly used metrics like BLEU-4,[8] and METEOR (Denkowski and Lavie, 2014). Because neither of these metrics can measure how well the generated recipe follows the input goal and the agenda, we also define two additional metrics. The first measures the percentage of the agenda items corrected used, while the second measures the number of extraneous items incorrectly introduced. Both these metrics are computed based on simple string match and can miss certain referring expressions (e.g., "meat" to refer to "pork"). Because of the approximate nature of these automated metrics, we also report a human evaluation.

6 Recipe generation results

Fig. 1 results for recipe generation. All BLEU and METEOR scores are low, which is expected for long texts. Our checklist model performs better than both neural network baselines (Attention and EncDec) in all metrics. Nearest neighbor baselines (NN and NN-Swap) perform the best in terms of BLEU and

[7] For this model, parameters were initialized on [-0.2, 0.2] to maximize development accuracy.

[8] See Moses system (http://www.statmt.org/moses/)

Model	Syntax	Ingredient use	Follows goal
Attention	4.47	3.02	3.47
EncDec	**4.58**	3.29	3.61
NN	4.22	3.02	3.36
NN-Swap	4.11	3.51	3.78
Checklist	**4.58**	3.80	3.94
Checklist+	4.39	**3.95**	**4.10**
Truth	4.39	4.03	4.34

Table 2: Human evaluation results on the generated and true recipes. Scores range in $[1, 5]$.

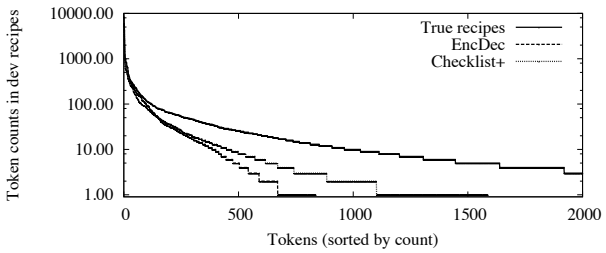


Figure 3: Counts of the most used vocabulary tokens (sorted by count) in the true dev set recipes and in generated recipes.

METEOR; this is due to a number of recipes that have very similar text but make different dishes.

However, NN baselines are not successful in generating a goal-oriented text that follows the given agenda: compared to Checklist+ (83.4%), they use substantially less % of the given ingredients (40% - 58.2%) while also introducing extra ingredients not provided. EncDec and Attention baselines similarly generate recipes that are not relevant to the given input, using only 22.8% - 26.9% of the agenda items. Checklist models rarely introduce extraneous ingredients not provided (0.6 - 0.8), while other baselines make a few mistakes on average (2.0 - 4.2).

The ablation study demonstrates the empirical contribution of different model components. ($\mathbf{o}_t = \mathbf{h}_t$) shows the usefulness of the attention encodings when generating the agenda references, while (*-no used*) shows the need for separate attention mechanisms between new and used ingredient references for more accurate use of the agenda items. Similarly, (*-no supervision*) demonstrates that the weak supervision encourages the model to learn more accurate management of the agenda items.

Human evaluation Because neither BLEU nor METEOR is suitable for evaluating generated text in terms of their adherence to the provided goal and the agenda, we also report human evaluation using Amazon Mechanical Turk. We evaluate the generated recipes on (1) grammaticality, (2) how well the recipe adheres to the provided ingredient list, and (3) how well the generated recipe accomplishes the desired dish. We selected 100 random test recipes. For each question we used a Likert scale ($\in [1, 5]$) and report averaged ratings among five turkers.

Table 2 shows the averaged scores over the responses. The checklist models outperform all baselines in generating recipes that follow the provided agenda closely and accomplish the desired goal, where NN in particular often generates the wrong dish. Perhaps surprisingly, both the Attention and EncDec baselines and the Checklist model beat the true recipes in terms of having better grammar. This can partly be attributed to noise in the parsing of the true recipes, and partly because the neural models tend to generate shorter, simpler texts.

Fig. 3 shows the counts of the most used vocabulary tokens in the true dev set recipes compared to the recipes generated by EncDec and Checklist+. Using the vocabulary from the training data, the true dev recipes use 5206 different tokens. The EncDec's vocabulary is only $\sim$16% of that size, while the Checklist+ model is a third of the size.

An error analysis on the dev set shows that the EncDec baseline over-generates catch-all phrases like "all ingredients" or "the ingredients," used in 21% of the generated recipes, whereas only 7.8% of true recipes use that construction. This phrase type simplifies the recipe, but using all ingredients in one step reduces the chance of accomplishing the desired dish correctly. The Checklist model only generates those phrases 13% of the time.

Qualitative analysis Fig. 4 shows two dev set recipes with generations from the EncDec and Checklist+ models. The EncDec model is much more likely to both use incorrect ingredients and to introduce ingredients more than once (e.g., "baking power" and "salt" in the bottom example are not in the ingredient list, and "milk" in the top example is duplicated). In the top example, the Checklist+ model refers to both Parmesean and Swiss cheese as "cheese"; generating more precise referring expressions is an important area for future work. The Checklist+ recipes generate the correct dishes to an extent: for example, the top recipe makes a casserole but does not cook the ingredients together before baking and mixes in biscuits instead of putting

Title: Casino royale casserole
Ingredients: 10 oz chopped broccoli, 2 tbsp butter, 2 tbsp flour, 1/2 tsp salt, 1/4 tsp black pepper, 1/4 tsp ground nutmeg, 1 cup milk, 1 1/2 cup shredded swiss cheese, 2 tsp lemon juice, 2 cup cooked cubed turkey, 4 oz mushrooms, 1/4 cup grated Parmesan cheese, 1 can refrigerated biscuits

Truth	EncDec	Checklist+
Preheat oven to 375. Melt butter in saucepan. Blend in flour, salt, pepper, and nutmeg; cook 1-2 minutes. Gradually add milk; cook, stirring, until slightly thickened. Stir in frozen vegetables. Remove from heat; stir in cheese until melted. Add lemon juice, turkey, mushrooms, and broccoli. Pour mixture into a lightly greased 8"x8" baking dish; sprinkle with parmesean and top with biscuits. Bake 20 minutes, or until biscuits are golden brown.	Cook broccoli according to package directions; drain. In medium saucepan, melt butter. Stir in flour, salt and pepper. Gradually add milk, stirring constantly. Cook over medium heat, stirring constantly, until mixture comes to a boil. Reduce heat to low. Cook, stirring constantly, until mixture boils and thickens, about 5 minutes. Stir in **milk**. Gradually heat to boiling; reduce heat. Cover; simmer 10 minutes or until heated through. Serve immediately.	Mix the turkey with the flour, salt, pepper, milk and lemon juice. Stir in the cheese. Add broccoli, mushrooms, nutmeg, and cheese. Mix well. Add the butter and mix well. Add the biscuits and mix well. Spread in a greased 13x9 inch baking pan. Bake in a 350f oven for 35 to 40 minutes, or until pick inserted in the center comes out clean. Serve hot.

Title: Kathie Frieman's cheese cake
Ingredients: 2 lb cream cheese, 1 3/4 cups sugar, 5 eggs, 15 oreo cookies, 1/2 tsp vanilla, 1/2 tsp almond extract

Truth	EncDec	Checklist+
Butter pan well. Whip cheese until soft and add sugar, combine until fluffy. Add extracts and eggs. Chop oreos into pieces and fold into mixture. Pour into pan and place in a pan of 1 or more inches of water. Bake at 350 1 1/2 hours (check after 1 1/4 hours) until center is set. Cool in pan of water. When cool turn upside-down on plate. Sprinkle with cookie crumbs.	Beat cream cheese and sugar together until light and fluffy. Add eggs, one at a time, beating well after each addition. Blend in vanilla. Sift flour, measure, and sift with **baking power** and **salt**. Add to creamed mixture alternately with **milk**. Fold in stiffly beaten **egg whites**. Pour into 9x13 pan. Bake at 350 degrees for 25 to 30 minutes.	Cream sugar and cream cheese until smooth. Add eggs, one at a time, beating well after each addition. Add vanilla, almond extract, and cookies. Mix well. Pour into greased and floured tube pan. Bake at 350 degrees for 30 minutes.

Figure 4: Example dev set generated recipes. Tokenization, newlines, and capitalization changed for space and readability. Bolded ingredient references are either ingredients not in the list and/or duplicated initial ingredient references.

Model	Hotel		Restaurant	
	BLEU	**METEOR**	**BLEU**	**METEOR**
HDC	55.52	48.10	44.39	43.42
SC-LSTM	86.53	60.84	74.49	54.31
Checklist	**90.61**	**62.10**	**77.82**	**54.42**

Table 3: Quantitative evaluation of the top generations in the hotel and restaurant domains

them on top. Future work could better model the full set of steps needed to achieve the overall goal.

7 Dialogue system results

Figure 3 shows our results on the hotel and restaurant dialogue system generation tasks. HDC is the rule-based baseline from Wen et al. (2015). For both domains, the checklist model achieved the highest BLEU-4 and METEOR scores, but both neural systems performed very well. The power of our model is in generating long texts, but this experiment shows that our model can generalize well to other tasks with different kinds of agenda items and goals.

8 Future work and conclusions

We present the neural checklist model that generates globally coherent text by keeping track of what has been said and still needs to be said from a provided agenda. Future work includes incorporating referring expressions for sets or compositions of agenda items (e.g., "vegetables"). The neural checklist model is sensitive to hyperparameter initialization, which should be investigated in future work. The neural checklist model can also be adapted to handle multiple checklists, such as checklists over composite entities created over the course of a recipe (see Kiddon (2016) for an initial proposal).

Acknowledgements

This research was supported in part by the Intel Science and Technology Center for Pervasive Computing (ISTC-PC), NSF (IIS-1252835 and IIS-1524371), DARPA under the CwC program through the ARO (W911NF-15-1-0543), and gifts by Google and Facebook. We thank our anonymous reviewers for their comments and suggestions, as well as Yannis Konstas, Mike Lewis, Mark Yatskar, Antoine Bosselut, Luheng He, Eunsol Choi, Victoria Lin, Kenton Lee, and Nicholas FitzGerald for helping us read and edit. We also thank Mirella Lapata and Annie Louis for their suggestions for baselines.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods on Natural Language Processing*, pages 1721–1731.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, pages 331–338.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Robert Dale. 1988. *Generating Referring Expressions in a Domain of Objects and Processes*. Ph.D. thesis, Centre for Cognitive Science, University of Edinburgh.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, pages 376–380.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *Representation Learning Worksop, ICML*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

Kristian J. Hammond. 1986. CHEF: A model of case-based planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 267–271.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 12–22.

Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding long-short term memory for image caption generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2407–2415.

Chloé Kiddon. 2016. *Learning to Interpret and Generate Instructional Recipes*. Ph.D. thesis, Computer Science & Engineering, University of Washington.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research (JAIR)*, 48:305–346.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 91–99.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, September.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective generation using lstms with coarse-to-fine alignment. In *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH 2010, the 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP 2011)*, pages 5528–5531.

Shinsuke Mori, Hirokuni Maeta, Tetsuro Sasada, Koichiro Yoshino, Atsushi Hashimoto, Takuya Funatomi, and Yoko Yamakata. 2014. FlowGraph2Text: Automatic sentence skeleton compilation for procedural text generation. In *Proceedings of the 8th International Natural Language Generation Conference*, pages 118–122.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Henry S. Thompson. 1977. Strategy and tactics: a model for language production. In *Papers from the Thirteenth Regional Meeting of the Chicago Linguistics Society*, pages 89–95. Chicago Linguistics Society.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016a. Context gates for neural machine translation. *CoRR*, abs/1608.06043.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-hao Su, David Vandyke, and Steve J. Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057.

A Dataset and Evaluation Metrics for Abstractive Compression of Sentences and Short Paragraphs

Kristina Toutanova
Microsoft Research
Redmond, WA, USA

Chris Brockett
Microsoft Research
Redmond, WA, USA

Ke M. Tran*
University of Amsterdam
Amsterdam, The Netherlands

Saleema Amershi
Microsoft Research
Redmond, WA, USA

Abstract

We introduce a manually-created, multi-reference dataset for abstractive sentence and short paragraph compression. First, we examine the impact of single- and multi-sentence level editing operations on human compression quality as found in this corpus. We observe that substitution and rephrasing operations are more meaning preserving than other operations, and that compressing in context improves quality. Second, we systematically explore the correlations between automatic evaluation metrics and human judgments of meaning preservation and grammaticality in the compression task, and analyze the impact of the linguistic units used and precision versus recall measures on the quality of the metrics. Multi-reference evaluation metrics are shown to offer significant advantage over single reference-based metrics.

1 Introduction

Automated sentence compression condenses a sentence or paragraph to its most important content in order to enhance writing quality, meet document length constraints, and build more accurate document summarization systems (Berg-Kirkpatrick et al., 2011; Vanderwende et al., 2007). Though word deletion is extensively used (e.g., (Clarke and Lapata, 2008)), state-of-the-art compression models (Cohn and Lapata, 2008; Rush et al., 2015) benefit crucially from data that can represent complex abstractive compression operations, including substitution of words and phrases and reordering.

This paper has two parts. In the first half, we introduce a manually-created *multi-reference* dataset for *abstractive* compression of sentences and short paragraphs, with the following features:

- It contains approximately 6,000 source texts with multiple compressions (about 26,000 pairs of source and compressed texts), representing business letters, newswire, journals, and technical documents sampled from the Open American National Corpus (OANC[1]).
- Each source text is accompanied by up to five crowd-sourced rewrites constrained to a preset compression ratio and annotated with quality judgments. Multiple rewrites permit study of the impact of operations on human compression quality and facilitate automatic evaluation.
- This dataset is the first to provide compressions at the multi-sentence (two-sentence paragraph) level, which may present a stepping stone to whole document summarization. Many of these two-sentence paragraphs are compressed both as paragraphs and separately sentence-by-sentence, offering data that may yield insights into the impact of multi-sentence operations on human compression quality.
- A detailed edit history is provided that may allow fine-grained alignment of original and compressed texts and measurement of the cognitive load of different rewrite operations.

Our analysis of this dataset reveals that abstraction has a significant positive impact on meaning preservation, and that application of trans-sentential

*This research was conducted during the author's internship at Microsoft Research.

[1] http://www.anc.org/data/oanc

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 340–350,
Austin, Texas, November 1-5, 2016.

context has a significant positive impact on both meaning preservation and grammaticality.

In the second part, we provide a systematic empirical study of eighty automatic evaluation metrics for text compression using this dataset, correlating them with human judgments of meaning and grammar. Our study shows strong correlation of the best metrics with human judgments of meaning, but weaker correlations with judgments of grammar. We demonstrate significant gains from multiple references. We also provide analyses of the impact of the linguistics units used (surface n-grams of different sizes versus parse-based triples), and the use of precision versus recall-based measures.

2 Related Work

Prior studies of human compression: Clarke (2008) studied the properties of manually-collected deletion-based compressions in the news genre, comparing them with automatically-mined data from the Ziff-Davis corpus in terms of compression rate, length of deleted spans, and deletion probability by syntactic constituent type. Jing and McKeown (1999) identified abstractive operations (other than word deletion) employed by professional writers, including paraphrasing and re-ordering of phrases, and merging and reordering sentences, but did not quantify their impact on compression quality.

Deletion-based compression corpora: Currently available automatically-mined deletion corpora are single-reference and have varying (uncontrolled) compression rates. Knight and Marcu (2002) automatically mined a small parallel corpus (1,035 training and 32 test sentences) by aligning abstracts to sentences in articles. Filippova and Altun (2013) extracted deletion-based compressions by aligning news headlines to first sentences, yielding a corpus of 250,000 parallel sentences. The same approach was used by Filippova et al. (2015) to create a set of 2M sentence pairs. Only a subset of 10,000 parallel sentences from the latter has been publicly released. Clarke and Lapata (2006) and Clarke and Lapata (2008) provide two manually-created two-reference corpora for deletion-based compression:[2] their sizes are 1,370 and 1,433 sentences, respectively.

[2] http://jamesclarke.net/research/resources

Abstractive compression corpora: Rush et al. (2015) have mined 4 million compression pairs from news articles and released their code to extract data from the Annotated Gigaword (Napoles et al., 2012). A news-domain parallel sentence corpus containing 1,496 parallel examples has been culled from multi-reference Chinese-English translations by Ganitkevitch et al. (2011). The only publicly-available manually-created abstractive compression corpus is that described by Cohn and Lapata (2008), which comprises 575 single-reference sentence pairs.

Automatic metrics: Early automatic metrics for evaluation of compressions include success rate (Jing, 2000), defined as accuracy of individual word or constituent deletion decisions; Simple String Accuracy (string edit distance), introduced by Bangalore et al. (2000) for natural language generation tasks; and Word Accuracy (Chiori and Furui, 2004), which generalizes Bangalore et al. (2000) to multiple references. Riezler et al. (2003) introduced the use of F-measure over grammatical relations. Word unigram and word-bigram F-measure have also been used (Unno et al., 2006; Filippova et al., 2015). Variants of ROUGE (Lin, 2004), used for summarization evaluation, have also been applied to sentence compressions (Rush et al., 2015).

Riezler et al. (2003) show that F-measure over grammatical relations agrees with human ratings on the relative ranking of three systems at the corpus level. Clarke and Lapata (2006) evaluate two deletion-based automatic compression systems against a deletion-based gold-standard on sets of 20 sentences. Parse-based F-1 was shown to have high sentence-level Pearson's ρ correlation with human judgments of overall quality, and to have higher ρ than Simple String Accuracy.

Napoles et al. (2011) have pointed to the need of multiple references and studies of evaluation metrics. For the related tasks of document and multi-document summarization, Graham (2015) provides a fine-grained comparison of automated evaluation methods. However, to the best of our knowledge, no studies of automatic evaluation metrics exist for abstractive compression of shorter texts.

Length		Text	Operations
1-Sent	Source	Think of all the ways everyone in your household will benefit from your membership in Audubon.	N/A
	Ref-1	Imagine how your household will benefit from your Audubon membership.	paraphrase + deletion + transformation
	Ref-2	Everyone in your household will benefit from membership in Audubon.	deletion
2-Sent	Source	Will the administration live up to its environmental promises? Can we save the last of our ancient forests from the chainsaw?	N/A
	Ref-1	Can the administration keep its promises? Can we save the last of our forests from loss?	two-sentences + deletion + paraphrase
	Ref-2	Will the administration live up to its environmental promises to save our ancient forests?	merge + deletion

Table 1: Examples of 1- and 2-sentence crowd-sourced compressions, illustrating different rewrite types.

	Newswire	Letters	Journal	Non-fiction
#texts	695	1,591	1,871	2,012

Table 2: Overview of the dataset by genre.

3 Dataset: Annotation and Properties

We sampled single sentences and two-sentence paragraphs from several genres in the written text section of the Manually Annotated Sub-Corpus (MASC) (Ide et al., 2008; Ide et al., 2010) of the Open American National Corpus (OANC), supplemented by additional data from the written section of OANC. Two-sentence paragraphs account for approximately 23% of multi-sentence paragraphs in the OANC. The two-sentence paragraphs we sampled contain at least 25 words. Table 2 breaks the sampled texts down by genre. Non-news genres are better represented in our sample than the newswire typically used in compression tasks. The *Letters* examples are expected to be useful for learning to compress emails. The *Journal* texts are likely to be challenging as their purpose is often more than to convey information. The *Non-Fiction* collection includes material from technical academic publications, such as PLoS Medicine, an open access journal.[3]

3.1 Annotation

Compressions were created using UHRS, an in-house crowd-sourcing system similar to Amazon's Mechanical Turk, in two annotation rounds, one for shortening and a second to rate compression quality.

Generating compressions: In the first round, we asked five workers (*editors*) to abridge each source text by at least 25%, while remaining grammatical and fluent, and retaining the meaning of the original. This requirement was enforced programmatically on the basis of character count. The 25% rate is intended to reflect practical editing scenarios (e.g., shrink 8 pages to 6). To facilitate meeting this requirement, the minimum source text length presented to editors was 15 words. For a subset of paragraphs, we collected compressions both as independent rewrites of their component sentences, and of the paragraph as a whole. Table 1 show compression examples and strategies.

Evaluating compression quality: In the second round, we asked 3-5 judges (*raters*) to evaluate the grammaticality of each compression on a scale from 1 (major errors, disfluent) through 3 (fluent), and again analogously for meaning preservation on a scale from 1 (orthogonal) through 3 (most important meaning-preserving).[4] We later used the same process to evaluate compressions produced by automatic systems. The full guidelines for the editors and raters are available with the data release.

Quality controls: All editors and raters were based in the US, and the raters were required to pass a qualification test which asked them to rate the meaning and grammaticality for a set of examples with known answers. To further improve the quality of the data, we removed low-quality compressions. We computed the quality of each compression as the average of the grammar and meaning quality as judged by the raters. We then computed the mean quality for each editor, and removed compressions authored by the bottom 10% of editors. We did the same for the bottom 10% of the raters.[5]

[3] http://journals.plos.org/plosmedicine/

[4] Pilot studies suggested that a scale of 1-3 offered better inter-annotator agreement than the standard 5-point Likert-type scale, at the cost of granularity.

[5] This was motivated by the observation that the quality of work produced by judges is relatively constant (Gao et al., 2015).

Description	Texts			Quality	
	Source	Target	Avg CPS	Meaning	Grammar
All	6,169	26,423	4.28	2.78	2.82
Per Source Length					
1-Sent	3,764	15,523	4.12	2.78	2.81
2-Sent	2,405	10,900	4.53	2.78	2.83

Table 3: Overview of the dataset, presenting the overall number of source and target texts, the average quality of the compressed texts, and breakdown by length of source (number of sentences).

Table 3 shows the number of compressions in the cleaned dataset, as well as the average number of compressions per source text (CPS) and the average meaning and grammar scores. Meaning quality and grammaticality scores are relatively good, averaging 2.78 and 2.82 respectively. The filtered crowdsourced compressions were most frequently judged to retain the most important meaning (80% of the time), or much of the meaning (17% of the time), with the lowest rating of 1 appearing only 3% of the time. This distribution is quite different from that of automatic compression systems in Section 4.

We provide a standard split of the data into training, development and test sets.[6] There are 4,936 source texts in the training, 448 in the development, and 785 in the test set.

3.2 Inter-Annotator Agreement

Crowd Workers: Since a different set of judges performs each task, large sets of inputs judged by the same two raters are unavailable. To simulate two raters, we follow Pavlick and Tetrault (2016): for each sentence, we randomly choose one annotator's output as the category for annotator A, and select the rounded average ranking for the remaining annotators as the category for annotator B. We then compute quadratic weighted κ (Cohen, 1968) for this pair over the whole corpus. We repeat the process 1000 times to compute the mean and variance of κ. The first row of the Table 4 reports the absolute agreement and κ, where the absolute agreement measures the fraction of times that A is equal to B. The 95% confidence intervals for κ are narrow, with width at most .01.

[6]The dataset can be downloaded from the project's website `https://www.microsoft.com/en-us/research/project/intelligent-editing/`.

Description	**Meaning**		**Grammar**	
	Agreement	κ	**Agreement**	κ
worker versus worker	.721	.306	.784	.381
expert versus expert	.888	.518	.890	.514
expert versus worker	.946	.549	.930	.344

Table 4: Agreement on meaning preservation and grammatically between crowd workers and experts.

Expert Raters: A small sample of 116 sentence pairs was rated by two expert judges. We used quadratic weighted κ directly, without sampling. To assess agreement between experts and non-experts, we computed weighted κ between the (rounded) average of the expert judgments and the (rounded) average of the crowd judgments, using 25,000 bootstrap replications each. The results are shown in the last two rows of Table 4. The confidence intervals for κ are wide due to the small sample size, and span values up to .17 away from the mean. Overall, agreement of experts with the average crowdsourced ratings is moderate (approaching substantial) for meaning, and fair for grammar.

3.3 Analysis of Editing Operations

Frequency analysis: To analyze the editing operations used, we applied the state-of-the-art monolingual aligner Jacana (Yao et al., 2013) to align input to compressed texts. Out of the 26,423 compressions collected, 25.2% contained only token deletions. Those containing deletion and reordering amounted to a mere 9.1%, while those that also contain substitution or rephrasing (abstractive compressions) is 65.6%. Although abstraction is present in the large majority of compressions, these statistics do not indicate that paraphrasing is more prevalent than copying at the token level. The word alignments for target compression words indicate that 7.1% of target tokens were inserted, 75.4% were copied and 17.3% were paraphrased. From the alignments for source text words, we see that 31% of source words were deleted. The fraction of inserted and deleted words is probably overestimated by this approach, as it is likely that sequences of source words were abstracted as shorter sequences of target words in many-to-one or many-to-many alignment patterns that are difficult to detect automatically.

For the subset of examples where the input text

Operation	Meaning		Grammar	
	Present	Absent	Present	Absent
Substitute	2.81**	2.70	2.79	2.85**
Reorder	2.80	2.82	2.80	2.82**
Merge	2.63	2.82**	2.84**	2.82
Sentence Delete	2.57	2.82*	2.84	2.75

Table 5: Meaning and grammaticality judgments by compression operation. *p = 0.002. **p $<$ 0.0001.

Source Type	Meaning	Grammar
2-Sentence	2.86**	2.87**
1-Sentence	2.78	2.82

Table 6: Meaning and grammaticality judgments for compressing two sentences jointly versus individually. **p $<$ 0.0001.

contained more than one sentence, we computed the frequency of sentence-merging and sentence deletion when compressing. Of the compressions for two-sentence paragraphs, 72.4% had two sentences in the output, 0.4% had one sentence deleted, and 27.3% had the two source sentences merged.

Impact of operations: Because the dataset contains multiple compressions of the same sources, we are able to estimate the impact of different editing operations. These were classified using the Jacana word alignment tool. Table 5 presents the average judgment scores for meaning preservation and grammaticality for four operations. The upper two rows apply to all texts, the lower two to two-sentence paragraphs only. The statistical significance of their impact was tested using the Wilcoxon signed-rank test on paired observations. It appears that raters view compressions that involve substitutions as significantly more meaning-preserving than those that do not ($p < 0.0001$), but judge their grammaticality to be lower than that of deletion-based compressions. Note that the reduced grammaticality may be due to typographical errors that have been introduced during rephrasing, which could have been avoided had a more powerful word processor been used as an editing platform. Reordering has no significant impact on meaning, but leads to substantial degradation in grammatically. Conversely, abridgments that merge or delete sentences are rated as significantly less meaning preserving, but score higher for grammaticality, possibly reflecting greater skill on the part of those editors..

Impact of sentence context: Table 6 shows that the context provided by 2-sentence sources yields significantly improved scores for both meaning and grammaticality. Here we used the matched pairs design to compare the average quality of two-sentence paragraph compressions with the average quality of the compressions of the same paragraphs produced by separately compressing the two sentences.

4 Evaluating Evaluation Metrics

Progress in automated text compression is standardly measured by comparing model outputs at the *corpus* level. To train models discriminatively and to perform fine-grained system comparisons, however, it is also necessary to have evaluation of system outputs at the *individual input* level. Below, we examine automated metric correlation with human judgments at both levels of granularity.

4.1 Automatic Metrics

The goal of this analysis is to develop an understanding of the performance of automatic evaluation metrics for text compression, and the factors contributing to their performance. To this end, we group automatic metrics according to three criteria. The first is the *linguistic units* used to compare system and reference compressions. Prior work on compression evaluation has indicated that a parse-based metric is superior to one based on surface substrings (Clarke and Lapata, 2006), but the contribution of the linguistic units has not been isolated, and surface n-gram units have otherwise been successfully used for evaluation in related tasks (Graham, 2015). Accordingly, we empirically compare metrics based on surface uni-grams (LR-1), bi-grams (LR-2), tri-grams (LR-3), and four-grams (LR-4), as well skip bi-grams (with a maximum of four intervening words as in ROUGE-S4) (SKIP-2), and dependency tree triples obtained from collapsed dependencies output from the Stanford parser (PARSE-2).[7] The second criterion is the *scoring measure* used to evaluate the match between two sets of linguistic units corresponding to a system output and a reference compression. We compare Precision, Recall, F-measure, and Precision+Brevity penalty (as

[7] Clarke and Lapata (2006) used the RASP parser (Briscoe and Carroll, 2002), but we expect that the Stanford parser is similarly robust and would lead to similar correlations.

in BLEU). The third criterion is whether *multiple references* or a *single reference* is used, and in the case of multiple references, the method used to aggregate information from multiple references. We investigate two previously applied methods and introduce a novel approach that often outperforms the standard methods.

To illustrate, we introduce some notation and use a simple example. Consider a sub-phrase of one of the sentences in Table 1, *think about your household*, as an input text to compress. Let us assume that we have two reference compressions, R1: *imagine your household*, and R2: *your household*. Each metric m is a function from a pair of a system output o and a list of references $r_1, r_2, \ldots, r_k$ to the reals. To compute most metrics, we first compute a linguistic unit feature vector for each reference $\Phi(r_j)$, as well as for the set of references $\Phi(r_1, r_2, \ldots, r_k)$. Similarly, we compute a linguistic unit vector for the output $\Phi(o)$ and measure the overlap between the system and reference vectors. The vectors of the example references, if we use surface bigram units, would be, for R1, {`imagine_your:1`, `your_household:1`}, and for R2, {`your_household:1`}. The weights of all n-grams in individual references and system outputs are equal to 1.[8] If we use dependency-parse triples instead, the vector of R2 would be {`nmod:poss(household, your):1`}.

The precision of a system output against a reference is defined as the match $\Phi(r)^T\Phi(o)$ divided by the number of units in the vector of o; the latter can be expressed as the L_1 norm of $\Phi(o)$ because all weights are positive: $\text{Precision}(o, r) = \frac{\Phi(r)^T\Phi(o)}{|\Phi(o)|_1}$. The recall against a single reference can be similarly defined as the match divided by the number of units in the reference: $\text{Recall}(o, r) = \frac{\Phi(r)^T\Phi(o)}{|\Phi(r)|_1}$.

We distinguish three methods for aggregating information from multiple references: MULT-MAX which uses the single reference out of a set that results in the highest single-reference score, and two further methods, MULT-ALL and MULT-PROB, that construct an aggregate linguistic unit vector $\Phi(r_1, \ldots, r_k)$ before matching. MULT-ALL is the standard method used in multi-reference BLEU, where the vector for a set of references is defined as the union of the features of the set. For our example, the combined vector of R1 and R2 is equal to the vector of R1, because R2 adds no new bigrams. MULT-PROB, a new method that we propose here, is motivated by the observation that although judgments of importance of content are subjective, the more annotators assert some information is important, the more this information should contribute to the matching score.[9] In MULT-PROB we define the weight of a linguistic unit in the combined reference vector as the proportion of references that include the unit. For our example, $\Phi_{\text{MULT-PROB}}(R1, R2)$ is {`imagine_your`:.5, `your_household`:1}.

4.2 Models for Text Compression

For the purpose of analysis, we trained and evaluated four compression systems. These include both deletion-based and abstractive models: (1) ILP, an integer linear programing approach for deletion-based compression (Clarke and Lapata, 2008), (2) T3, a tree transducer-based model for abstractive compression (Cohn and Lapata, 2008), (3) Seq2seq, a neural network model for deletion-based compression (Filippova et al., 2015), and (4) NAMAS, a neural model for abstractive compression and summarization (Rush et al., 2015). We are not concerned with the relative performance of these models so much as we are concerned with evaluating the automatic evaluation metrics themselves. We have sought to make the models competitive, but have not required that all systems use identical training data.

All of the models are evaluated on the test set portion of our dataset. All models use the training portion of the data for training, and two models (Seq2Seq and NAMAS[10]) additionally use external training data. The external data is summarized in Table 7. The Gigaword set was extracted from the Annotated Gigaword (Napoles et al., 2012), using the implementation provided by Rush et al. (2015). The Headline data was extracted in similar fashion using an in-house news collection.

[8] We handle repeating n-grams by assigning each subsequent n-gram of the same type a distinct type, so that the i-th *the* of a system output can match the i-th *the* of a reference.

[9] A similar insight was used in one of the component metrics of the SARI evaluation metric used for text simplification evaluation (Xu et al., 2016).

[10] The original works introducing these models employed much larger training corpora, believed to be key to improving the accuracy of neutral network models with large parameter spaces.

Data		#src tokens	#trg tokens	#sents
Abstractive	Gigaword	114.1M	30.0M	3.6M
	Headline	6.0M	1.4M	0.2M
Deletion-based	Gigaword	1,353K	329K	47K
	Headline	59K	11K	2K

Table 7: External data statistics.

ILP: We use an open-source implementation[11] of the semi-supervised ILP model described in (Clarke and Lapata, 2008). The model uses a trigram language model trained on a 9 million token subset of the OANC corpus. The ILP model requires parsed sentences coupled with deletion-based compressions for training, so we filtered and preprocessed our dataset to satisfy these constraints. We used all single sentence inputs with their corresponding deletion-based compressions, and additionally used two-sentence paragraph input/output pairs split into sentences by heuristically aligning source to target sentences in the paragraphs.

T3: We use the authors' implementation of the tree transducer system described in Cohn and Lapata (2008). T3 similarly requires sentence-level input/output pairs, but can also learn from abstractive compressions. We thus used a larger set of approximately 28,000 examples (single sentences with abstractive compressions taken directly from the data or as a result of heuristic sentence-level alignment of two-sentence paragraphs). We obtained parse trees using the Stanford parser (Klein and Manning, 2003), and used Jacana (Yao et al., 2013) for word alignment. The performance obtained by T3 in our experiments is substantially weaker (relative to ILP) than that reported in prior work (Cohn and Lapata, 2008). We therefore interpret this system output solely as data for evaluating automatic metrics.

NAMAS: We run the publicly available implementation of NAMAS[12] with the settings described by Rush et al. (2015). We modified the beam search algorithm to produce output with a compression ratio similar to that of the human references, since this ratio is a large factor in compression quality (Napoles et al., 2011), and systems generally perform better if allowed to produce longer output, up to the maximum length limit. We enforced output length between 50% and 75% of input length, which resulted in improved performance.

Seq2seq: We implemented the sequence-to-sequence model[13] described in Filippova et al. (2015). A deletion-based model, it uses the deletion-based subset of our training dataset and the deletion-based subset from the external data in Table 7. The encoder and decoder have three stacked LSTM layers, the hidden dimension size is 512, and the vocabulary size is 30,000. The compression rate was controlled in the same range as for the NAMAS model.

All models produce output on all inputs in the test set. For all models, we generated outputs for multi-sentence inputs by concatenating outputs for each individual sentence.[14]

4.3 Results

Overall, we consider 80 metric variants, consisting of combinations of six types of linguistic units, combined with three scoring methods (Precision, Recall, and F-measure) and four settings of single reference SINGLE-REF or three ways of scoring against multiple references MULT-ALL,MULT-MAX,MULT-PROB. Additionally, we include the standard single and multi-reference versions of BLEU-2,BLEU-3,BLEU-4, and ROUGE-L.

We compare automatic metrics to human judgements at the level of individual outputs or groups of outputs (the whole corpus). For a single output o, the human quality judgment is defined as the average assigned by up to five human raters. We denote the meaning, grammar, and combined quality values by $M(o)$, $G(o)$, and $C(o) = .5M(o) + .5G(o)$, respectively. We define the quality for a group of outputs as the arithmetic mean of judgments over the outputs in the group. We use the arithmetic mean of automating metrics at the individual output level to define automatic corpus quality metrics as well.[15] To compare different metrics and establish statistical significance of the difference between two metrics, we use Williams test of the significance of the difference

[11] https://github.com/cnap/sentence-compression

[12] https://github.com/facebook/NAMAS

[13] https://github.com/ketranm/tardis

[14] In small scale preliminary manual evaluation, we found that, although some models are theoretically able to make use of context beyond the sentence boundary, they performed better if they compressed each sentence in a sequence independently.

[15] This method has been standard for ROUGE, but has not for BLEU. We find that averaging sentence-level metrics is also advantageous for BLEU .

System	Meaning	Grammar	Combined
T3	1.14	1.40	1.26
NAMAS	1.56	1.30	1.43
Seq2Seq	1.64	1.51	1.57
ILP	2.28	2.22	2.25

Table 8: Average human ratings of system outputs for meaning and grammar separately and in combination.

between dependent Pearson correlations with human judgments (Williams, 1959) as recommended for summarization evaluation (Graham, 2015) and other NLP tasks (e.g. (Yannakoudakis et al., 2011)).

4.3.1 Corpus-level metrics

Table 8 shows the average human ratings of the four systems, separately in meaning and grammar, as well as the combined measure (an arithmetic mean of meaning and grammar judgments). Even though the performance of some systems is similar, the differences between all pairs of systems in meaning and grammar are significant $p < 0.0001$ according to a paired t-test. It is interesting to note that ILP outperforms the more recently developed neural network systems Seq2Seq and NAMAS. This might seem to contradict recent results showing that the new models are superior to traditional baselines, such as ILP. We note however that performance on the test corpus in our study might not substantially improve through the use of large automatically mined data-sets of headlines and corresponding news article sentences, due to differences in genre and domain. Using such data-sets for effective training of neural network models for non-newswire domains remains an open problem.

For each of the 80 metrics, we compared the ranking of the four systems with the ranking according to average human quality. Fifty three of the metrics achieved perfect Spearman ρ and Kendall τ_B correlation with human judgments of combined meaning and grammar quality. Due to the small sample size (four systems), we are unable to find statistically significant differences among metrics at the corpus level. We only note that precision-based metrics involving large linguistic units (four-grams) had negative correlations with human judgments. We can conclude, however, that evaluation at the corpus level is robust for a wide variety of standard metrics using linguistic units of size three or smaller.

4.3.2 Single input-level pairwise system comparisons

We can garner greater insight into the difference of metric performance when we compare metrics at the single input level. To gauge the ability of metrics to comparatively evaluate the quality of two systems, we compute single input-level correlations of automatic metrics with human judgments following the protocol of Galley et al. (2015). Each system A produces a sequence of outputs ${o_1}^A, \ldots, {o_n}^A$, corresponding to inputs $x_1, \ldots, x_n$. For each system output, we use $Q(a)$ to denote a generic human quality metric, varying over meaning, grammar, and their combination. For each pair of systems A and B, and each metric m, we compute the difference in quality for corresponding system outputs for each input x_i: $m({o_i}^A) - m({o_i}^B)$ and the difference in quality according to human judgments: $Q({o_i}^A) - Q({o_i}^B)$, and compute the correlation between these two sequences. We can thus compute the single input-level correlation between m and Q for each pair of systems A and B, resulting in a total of six correlation values (for the six pairs of systems) for each metric. For each pair of metrics m_1 and m_2, and for each pair of systems A and B, we compute the statistical significance of the difference between the Pearson correlations of these metrics with human judgements. We say that m_1 is significantly better than m_2 on the A vs. B comparison if its Pearson correlation with human quality Q is significantly better (according to the Williams test of the difference in dependent correlations) than that of m_2 with a p-value less than .05. We say that m_1 dominates m_2 overall if it is significantly better than m_2 on at least 80% of the pair-wise system comparisons.

Table 9 shows the main correlation results at the level of individual inputs. We report correlations with meaning, grammar, and combined quality separately. For each human quality metric, we see the top automatic metrics in the first group of rows. The top metrics are ones that, for at least 80% of the system comparisons, are not significantly dominated by any other metric. In addition, we show the impact of each of the three criteria: linguistic units, scoring measure, and multiple references, in corresponding groups of rows. For each linguistic unit type, we show the best-performing metric that uses units of

Top metrics					
SKIP-2+Recall+MULT-PROB	.59	PARSE-2+Recall+MULT-MAX	.35	PARSE-2+Recall+MULT-PROB	.52
PARSE-2+Recall+MULT-PROB	.57	LR-3+F-1+MULT-ALL	.35	PARSE-2+Recall+MULT-MAX	.52
SKIP-2+Recall+MULT-MAX	.58	PARSE-2+F-1+MULT-ALL	.35	SKIP-2+Recall+MULT-MAX	.51
		PARSE-2+Recall+MULT-PROB	.35	LR-2+Recall+MULT-PROB	.51
		LR-2+F-1+MULT-ALL	.34	LR-2+F-1+MULT-ALL	.50
		LR-3+Recall+MULT-ALL	.34		
Best per linguistic unit					
LR-1+Recall+MULT-PROB	.54	**LR-1**+Recall+MULT-MAX	.25^{-}	**LR-1**+Recall+MULT-PROB	.44^{-}
LR-2+Recall+MULT-PROB	.56*	**LR-2**+F-1+MULT-ALL	.34*	**LR-2**+Recall+MULT-PROB	.51*
LR-3+Recall+MULT-ALL	.55*	**LR-3**+F-1+MULT-ALL	.35*	**LR-3**+F-1+MULT-ALL	.50*
LR-4+Recall+MULT-ALL	.52^{-}	**LR-4**+F-1+MULT-ALL	.34*	**LR-4**+Recall+MULT-ALL	.47
SKIP-2+Recall+MULT-PROB	.59*	**SKIP-2**+F-1+MULT-PROB	.33	**SKIP-2**+Recall+MULT-MAX	.51*
PARSE-2+Recall+MULT-PROB	.57*	**PARSE-2**+Recall+MULT-MAX	.35*	**PARSE-2**+Recall+MULT-PROB	.52*
Best per scoring type					
SKIP-2+**Recall**+MULT-PROB	.59*	PARSE-2+**Recall**+MULT-MAX	.35	PARSE-2+**Recall**+MULT-PROB	.52
SKIP-2+**Precision**+MULT-ALL	.36^{-}	LR-2+**Precision**+MULT-ALL	.31	SKIP-2+**Precision**+MULT-ALL	.37^{-}
SKIP-2+**F-1**+MULT-ALL	.58*	LR-3+**F-1**+MULT-ALL	.35	LR-2+**F-1**+MULT-ALL	.50
Best per reference aggregation					
SKIP-2+Recall+**SINGLE-REF**	.49^{-}	PARSE-2+F-1+**SINGLE-REF**	.29	SKIP-2+Recall+**SINGLE-REF**	.44^{-}
SKIP-2+Recall+**MULT-MAX**	.58*	PARSE-2+Recall+**MULT-MAX**	.35	PARSE-2+Recall+**MULT-MAX**	.52
SKIP-2+Recall+**MULT-PROB**	.59*	PARSE-2+Recall+**MULT-PROB**	.35	PARSE-2+Recall+**MULT-PROB**	.52
SKIP-2+Recall+**MULT-ALL**	.58*	LR-3+F-1+**MULT-ALL**	.35	LR-2+F-1+**MULT-ALL**	.50
Other standard setting combinations					
BLEU-3+PrecBrev+MULT-ALL	.50^{-}	BLEU-4+PrecBrev+MULT-ALL	.30	BLEU-3+PrecBrev+MULT-ALL	.45^{-}
ROUGE-L+Recall+MULT-MAX	.49^{-}	ROUGE-L+Recall+MULT-MAX	.27	ROUGE-L+Recall+MULT-MAX	.43

Table 9: Left to right: Pearson correlation of automatic metrics with human ratings for meaning, grammar, and combined quality.

this type. Similarly, for the other criteria, we show the best performing metric for each value of the criterion. Metrics with a * suffix in each group significantly dominate metrics with a $^{-}$ suffix. Metrics with a $^{-}$ suffix in a group are dominated by at least one other metric, possibly outside of the group. The lowest group of rows in each main column presents the performance of other metrics that cannot be classified directly based on the three criteria.

A high-level observation that can be made is that the correlations with meaning are much higher than the correlations with grammar. The best correlations in meaning can be classified as "strong", whereas the best correlations in grammar are in the "medium" range. Unigrams are heavily dominated by higher order n-grams in all settings. Four-grams are also weaker that other units in measuring meaning preservation. Dependency triple (parse-based) metrics are strong, in particular in measuring grammaticality, but do not significantly dominate skip bi-grams or contiguous bi-grams. The scoring measure used has a strong impact. We see that precision-based metrics are substantially dominated by metrics that incorporate recall, except for grammar evaluation. Importantly, we see that multiple references contribute substantially to metric quality, as all methods that use multiple references outperform single-reference metrics. In both meaning and combined evaluation, this difference was statistically significant. Finally, we observe that standard BLEU metrics and ROUGE-L were not competitive.

5 Conclusion

We have introduced a large manually collected multi-reference abstractive dataset and quantified the impact of editing operations and context on human compression quality, showing that substitution and rephrasing operations are more meaning preserving than other operations, and that compression in context improves quality. Further, in the first systematic study of automatic evaluation metrics for text compression, we have demonstrated the importance of utilizing multiple references and suitable linguistic units, and incorporating recall.

Acknowledgments

We are grateful to Jaime Teevan, Shamsi Iqbal, Dan Liebling, Bill Dolan, Michel Galley, and Wei Xu, together with the three anonymous reviewers for their helpful advice and suggestions.

References

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of INLG*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL-HLT*.

Ted Briscoe and John A Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of LREC*.

Hori Chiori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, 87(1):15–25.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of ACL-COLING*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, pages 399–429.

James Clarke. 2008. *Global Inference for Sentence Compression: An Integer Linear Programming Approach*. Ph.D. thesis, Univeristy of Edinburgh.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of EMNLP*.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of ACL-IJCNLP (Volume 2: Short Papers)*.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of EMNLP*.

Mingkun Gao, Wei Xu, and Chris Callison-Burch. 2015. Cost optimization in crowdsourcing translation: Low cost translations made even cheaper. In *Proceedings of NAACL-HLT*.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of EMNLP*.

Nancy Ide, Collin F. Baker, Christiane Fellbaum, Charles J. Fillmore, and Rebecca J. Passonneau. 2008. MASC: the manually annotated sub-corpus of american english. In *Proceedings of LREC*.

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of ACL (Volume 2: Short Papers)*.

Hongyan Jing and Kathleen R McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of SIGIR*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of ANLP*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Ellie Pavlick and Joel Tetrault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, pages 61–74.

Stefan Riezler, Tracy H King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of NAACL-HLT*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

Yuya Unno, Takashi Ninomiya, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Trimming CFG parse trees for sentence compression using machine learning approaches. In *Proceedings of COLING-ACL*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

Evan James Williams. 1959. *Regression analysis*. Wiley, New York.

Wei Xu, Courtney Napoles, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of ACL-HLT*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. A lightweight and high performance monolingual word aligner. In *Proceedings of ACL (Volume 2: Short Papers)*.

PaCCSS–IT: A Parallel Corpus of Complex–Simple Sentences for Automatic Text Simplification

Dominique Brunato, Andrea Cimino, Felice Dell'Orletta, Giulia Venturi
Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC–CNR)
ItaliaNLP Lab - *www.italianlp.it*
{name.surname}@ilc.cnr.it

Abstract

In this paper we present PaCCSS–IT, a Parallel Corpus of Complex–Simple Sentences for ITalian. To build the resource we develop a new method for automatically acquiring a corpus of complex–simple paired sentences able to intercept structural transformations and particularly suitable for text simplification. The method requires a wide amount of texts that can be easily extracted from the web making it suitable also for less–resourced languages. We test it on the Italian language making available the biggest Italian corpus for automatic text simplification.

1 Introduction

The availability of monolingual parallel corpora is a prerequisite for research on automatic text simplification (ATS), i.e. the task of reducing sentence complexity by preserving the original meaning. This has been recently shown for different languages, e.g. English (Zhu et al., 2010; Woodsend and Lapata, 2011; Wubben et al., 2012; Siddharthan and Angrosh, 2014), Spanish (Bott and Saggion, 2011; Bott and Saggion, 2014), French (Brouwers et al., 2014), Portuguese (Caseli et al., 2009), Danish (Klerke and Søgaard, 2012), Italian (Brunato et al., 2015). While English can rely on large datasets like the well-known Parallel Wikipedia Simplification corpus (Coster and Kauchak, 2011; Zhu et al., 2010) and, more recently, the Newsela corpus (Xu et al., 2015), for other languages similar resources are difficult to acquire and tend to be very small, thus preventing the application of data–driven techniques to automatically induce simplification operations. This is true for the language we are considering, i.e. Italian, where the only documented corpus for text simplification contains approximately 1,000 aligned original and manually simplified sentences (Brunato et al., 2015).

In this paper we present *PaCCSS–IT*, a Parallel Corpus of Complex–Simple Aligned Sentences for ITalian. To build the resource we developed a new approach for automatically acquiring a large corpus of paired sentences containing structural transformations which can be used as a developmental resource for text simplification systems. The proposed approach relies on monolingual sentence alignment techniques which have been exploited in different scenarios such as e.g. paraphrase detection (Ganitkevitch et al., 2013; Barzilay and Lee, 2003; Dolan et al., 2004) and evaluation (Chen and Dolan, 2011), question answering (Fader et al., 2013), textual entailment (Bosma and Callison-Burch, 2007), machine translation (Marton et al., 2009), short answer scoring (Koleva et al., 2014), domain adaptation of dependency parsing (Choe and McClosky, 2015). Specifically in ATS, these techniques are typically applied to already existing parallel corpora; in this case the task of aligning the original sentence to its corresponding simple version can be tackled by applying similarity metrics that consider the TF/IDF score of the words in the sentence (Barzilay and Elhadad, 2003; Nelken and Shieber, 2006; Coster and Kauchak, 2011) or methods taking into account also the order in which information is presented (Bott and Saggion, 2011).

Differently from these methods, our approach

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 351–361,
Austin, Texas, November 1-5, 2016.

contains two important novelties: the typology of the starting data and consequently the methodology developed to build the complex–simple aligned corpus. To overcome the scarcity of large parallel corpora of complex and simple texts in less–resourced languages like Italian, we started from a wide amount of texts that can be easily extracted from the web for all languages. This makes our method less expensive since it does not need a manually created corpus of aligned documents.

The proposed alignment method has been strongly shaped by the perspective from which we investigate text simplification, i.e. syntactic rather than lexical simplification. While lexical simplification aims at the substitution of complex words by simpler synonyms, syntactic simplification attempts to reduce complexity at grammatical level (Bott and Saggion, 2014). As shown by comparative analyses of monolingual parallel corpora in many languages, syntactic simplification concerns transformations affecting e.g. verbal features, the order of phrases or the deletion of redundant or unnecessary words (Brunato et al., 2015; Bott and Saggion, 2014; Coster and Kauchak, 2011; Caseli et al., 2009). Following this second perspective we define a method for bootstrapping and pairing sentences that intercepts simplification operations at morpho–syntactic and syntactic level typically used by human experts when simplify real texts.

Section 2 illustrates the approach to automatically acquire the corpus of complex–simple aligned sentences. In Section 3, the approach is tested and tuned on a development corpus. In Section 4, our approach is applied on a large corpus thus obtaining the final corpus of paired sentences, named PaCCSS–IT. In this last section, we also provide a global evaluation of the whole process and a qualitative analysis of the linguistic phenomena related to sentence complexity that we intercepted.

2 The Approach

Our approach for automatically acquiring the collection of paired sentences combines three steps. In a first step, we devised an unsupervised methodology *i)* to collect pools of sentences from a large corpus with overlapping lexicon but possible different structures; *ii)* to rank the resulting candidate sentences according to a similarity metric intended to bootstrap lexical–equivalent pairs undergoing structural transformations. In the second step, the top–list of the ranked pairs was manually revised and used to develop a classifier based on lexical, morpho–syntactic and syntactic features to detect the sentences correctly paired. In the third step, the individual sentences of each pair were ordered with respect to linguistic complexity computed by using an automatic readability assessment tool.

This approach has been tuned on PAISÀ[1] (Lyding et al., 2014) and tested on ItWaC (Baroni et al., 2009). The two analysed corpora were automatically POS tagged by the Part–Of–Speech tagger described in Dell'Orletta (2009) and dependency–parsed by the DeSR parser (Attardi et al., 2009).

PAISÀ is a freely distributed corpus of texts with Creative Commons license automatically harvested from the web. This corpus includes approximately 388,000 documents for a total of 250 millions of tokens and it is a large existing corpus of authentic contemporary texts in Italian which is free of copyright restrictions. ItWaC is the largest existing corpus of authentic contemporary texts in Italian. It is a 2 billion word corpus constructed from the Web limiting the crawl to the .it domain and using medium-frequency words from *La Repubblica* journalistic corpus and Basic Italian Vocabulary lists as seeds.

2.1 Unsupervised Step

The first step is aimed at clustering all sentences contained in a large corpus. To be included in the same cluster, the sentences have to share all lemmas tagged with the Part–Of–Speech (POS) "noun", "verb", "numeral", "personal pronoun" and "negative adverb". Nouns and verbs were selected because they capture the informational content of a sentence. The other functional categories have also to be shared, otherwise the meaning of the sentence would be altered. For example, the deletion of the negative adverb *non* (not) in one of the two following sentences would convey the opposite meaning: *Non farei mai una cosa del genere!* (I would never do something like that) *Non potevo fare una cosa del genere.* (I could not do something like that). In

[1] http://www.corpusitaliano.it/

the overlapping process we did not take into account the linear order of the considered lemma POS. This was meant to capture lexically–equivalent sentences undergoing potential structural transformations (e.g. passivization, topicalization).

All sentences within the same cluster were paired and the pairs were ranked for similarity by calculating the cosine distance between the sentence vectors. Each vector is constituted by the frequencies in the cluster of all lemma of the sentence. The cosine similarity served to discard different and equal or quasi–equal sentences.

The whole unsupervised step was used to select the set of candidate pairs reducing the number of pairs on which the following supervised step had been applied.

2.2 Supervised Step

The supervised step is meant to classify whether candidate pairs were correctly or incorrectly aligned. To this end, we built a classifier based on Support Vector Machines with a quadratic kernel using LIBSVM (Chang and Lin, 2001) that was trained on a corpus of paired sentences correctly aligned. The classifier used different types of linguistic features, i.e. lexical, morpho–syntactic and syntactic, meant to mainly capture structural transformations occurring in the paired sentences.

These features were extracted both calculating their distribution in each sentence and considering their overlap between the two paired sentences. They can be classified into the following types:

cosine similarity feature: it refers to the cosine value calculated for each pair of sentences;

raw text feature: it refers to the sentence length calculated in terms of i) tokens of each of the two paired sentences and ii) the different number of tokens between the two sentences;

lexical features: they refer to i) the lemma unigrams contained in the two sentences excluding the PoS already considered in the pairing process (i.e. nouns, verbs, numerals, personal pronouns, negative adverbs); ii) the distribution of word unigrams overlapping between the two paired sentences considering all PoS.

morpho–syntactic feature: it refers to the distribution of up to 4–grams of coarse grained Parts–Of–Speech;

syntactic features: they refer to i) the distribution of up to 4–grams of dependency types calculated with respect to the hierarchical parse tree structure and the surface linear ordering of words; ii) the distribution of up to 4–grams of coarse grained Parts–Of–Speech of a dependent (d) involved in a dependency relation and the dependency relation type (t) with respect to the hierarchical parse tree structure.

2.3 Readability Assessment Step

In the third step, the individual sentences of each classified pair were ordered with respect to linguistic complexity computed by using an automatic readability assessment tool. In text simplification research it is widely accepted the use of readability assessment metrics for evaluating the transformations that reduce sentence complexity (Zhu et al., 2010; Woodsend and Lapata, 2011; Vajjala and Meurers, 2016). Since our approach is devoted to building resources for developing ATS systems, we relied on readability assessment techniques to rank the individual sentences of the pair. To this aim, we used READ–IT (Dell'Orletta et al., 2011), the only existing NLP–based readability assessment tool devised for Italian. It operates on syntactically parsed texts and assigns to each sentence a score quantifying its readability. The assigned readability score ranges between 0 (easy–to–read) and 1 (difficult–to–read) referring to the percentage probability for the documents or sentences to belong to the class of difficult–to–read documents. The two poles were defined on two typologies of texts belonging to the same textual genre (i.e. newswire texts) but intended for different users: adults with a rudimentary literacy level or with mild intellectual disabilities for the easy–to–read pole and readers of a national daily newspaper considered of medium difficulty for $\sim$70% of Italian laymen for the difficult–to–read pole.

3 Tuning Process and Evaluation

In order to tune and evaluate each step of the proposed approach, we tested it on PAISÀ. We first pruned from the corpus the sentences with a number of tokens <5 and >40. The resulting sentences were then grouped with respect to their shared POS (i.e. nouns, verbs, numerals, personal pronouns and negative adverbs) and paired using cosine similarity.

Cosine	nº *correct* pairs	% *correct* pairs
0.92	54	12.03
0.91	151	29.49
0.90	91	26.76
0.89	157	23.36
0.88	331	48.04
0.87	336	68.15
0.86	674	57.36
0.85	107	57.22
0.84	176	61.75
0.83	1096	40.35
0.71	1092	38.97
0.70	62	27.80
	Total: 4,327	

Table 1: Absolute number and % distribution of *correct* extracted pairs for each manually reviewed cosine threshold in PAISÁ.

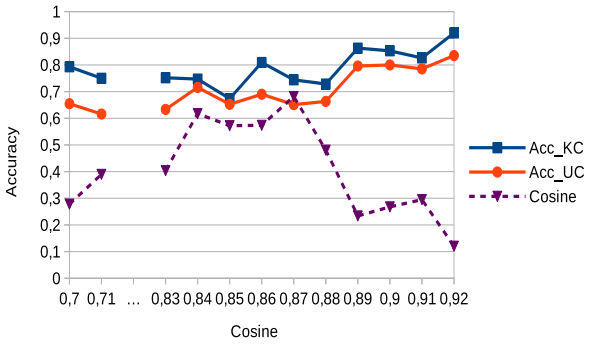


Figure 1: Accuracy of the *KC* and *UC* experiments compared with the distribution of correctly paired sentences at different cosine similarity scores.

We obtained 256,383 clusters containing at least two sentences. In order to discard different and equal or quasi–equal sentences we empirically set two cosine pruning thresholds: we discarded pairs with cosine below 0.4 since they were too lexically different and above 0.93 since they were too identical.

To build the training set for the supervised step we selected a subset of pairs resulting from the unsupervised step at different cosine similarity scores. This subset was manually reviewed by two native–speaker linguists with a background in text simplification. Specifically, they reviewed a subset of 10,543 pairs at different cosine similarity scores, i.e. those comprised between 0.92 and 0.83. In order to evaluate sentence similarity at lower values we also selected cosine scores 0.71 and 0.70. In the end, we obtained 4,327 correct pairs (i.e. about 41% of the whole set of candidate sentence pairs) distributed as in Table 1.

This manually revised set of pairs was then used to test the classifier in two different experimental scenarios. In the first one, named *Known Cosine, KC*, we tested the classifier in a five–fold cross validation process where pairs of sentences belonging to all the cosine scores were contained in each training and test set. In the second experiment, named *Unknown Cosine, UC*, the manually–revised corpus was differently split. In this case, the test set was composed by pairs of sentences with a cosine similarity score not contained in the training set and consequently twelve classification runs were performed.

In order to assess the discriminating power of the linguistic features used in the classification, we carried out an Information Gain analysis. This analysis showed the effectiveness of all the selected features in both experiments (i.e. KC, UC). In particular, we observed that the best ranked features are the morpho–syntactic and syntactic ones. This might suggest that our classification approach is intercepting pairs of sentences undergoing different typologies of structural transformations involving e.g. the use of verbal features or the order of phrases. Sentence length and lexical features play a lower discriminative role with respect to the grammatical features; this follows from the constraints we put on the unsupervised sentence pairing process. As it can be expected, the best ranked features are those providing information about the overlapping characteristics of the paired sentences.

Figure 1 and 2 report the results for each cosine threshold considered in the manual revision of the two experiments respectively in terms of *i)* Accuracy in the classification of the *correct* and *incorrect* alignments, and of *ii)* Precision, Recall of the classification of the *correct* alignments. As it can be noted in Figure 1, in both experiments the classifier is able to outperform the process of sentence pairing based only on the cosine (i.e. line *Cosine*, that represents the unsupervised step of the pairing process). As we can expect, Precision, Recall and Accuracy of the *KC* experiment are higher than the classification results obtained in the *UC*. The latter represents a more

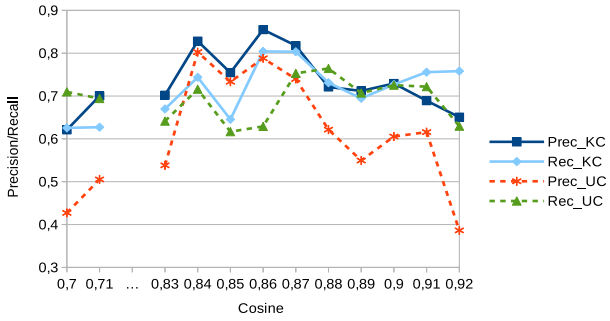


Figure 2: Precision and Recall of the two experiments (*KC* and *UC*) in the classification of the *correct alignments*.

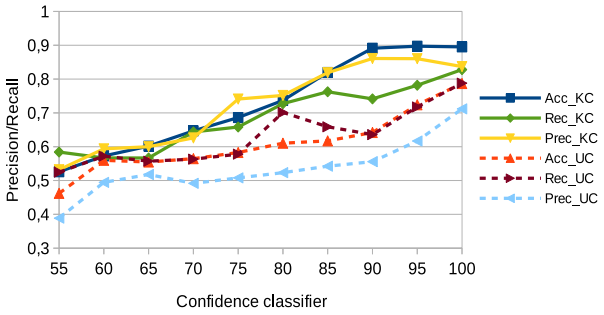


Figure 3: Classifier performance in the *KC* and *UC* experiments with probability intervals reported along the *x* axis.

challenging experimental scenario where the classifier is tested on a cosine threshold unseen in training. The overall results for the *KC* and the *UC* experiments are respectively 73.95% and 58.71% in terms of Precision, 70.3% and 68.1% in terms of Recall; and respectively 77.64% and 67.2% in terms of Accuracy. These results are significantly higher when compared with the accuracy of 41% reported for the unsupervised alignment (i.e. line *Cosine*). Interestingly, in the *KC* experiment, Precision and Recall lines are close and they remain stable with respect to all cosines even if the distribution of correct pairs varies in the different cosine values.

Figure 3 shows the accuracy of our classifier at different confidence thresholds (i.e. the probability assigned by the classifier for the *correct* alignments) for both the *KC* and *UC* experiments. Note that for each confidence intervals we have a different number of total pairs, and of gold-correct alignments and gold-incorrect alignments. As expected, the performance grows as the confidence grows. Interestingly, in the *KC* scenario, the classifier reached up to 90% of accuracy in discriminating the *correct* from the *incorrect* alignments when the classifier has a confidence score $\geq$0.90. These results look very promising if we consider that 30% of the pairs of the whole test set classified as correct alignments is comprised in the subset for which the classifier is more confident. This is also the case of the *UC* experiment, where, even if with lower accuracies, more than 56% of the *correct* alignments occurs when the classifier has a confidence score $\geq$0.90.

We carried out a last evaluation to estimate the classifier performance in the *UC* scenario for low cosine ranges not comprised in the manually revised portion of the corpus (from 0.45 to 0.75, excluding cosines 0.70 and 0.71). We considered only the pairs classified as *correct* with a confidence score of $\geq$85%. As expected, the system performance grows as the cosine grows: only few correct pairs occur at cosine $<$0.60, at cosine 0.60–0.65 the classifier assigns the *correct* class 237 times with an accuracy of 62.97%, at 0.65–0.69 330 times with 71.82% and at 0.72–0.75 256 times with an accuracy of 87.89%. According to these evaluations, we extracted from PAISÁ those pairs with a confidence score $\geq$ 85% and cosine similarity between 0.6 and 0.93, resulting in a collection of about 20,000 pairs.

In the last step, the sentences in each pair were ranked according to the readability score automatically assigned by READ–IT making a collection of complex–simple aligned sentences. However, the average difference of the readability score between the complex and simple sentences is only 0.13, making this collection not so useful for ATS. For this reason, we selected only pairs with a difference of readability score higher than a significant threshold set at 0.2. We defined this threshold on the basis of previous empirical experiments carried out using READ-IT on different typologies of texts. Lower variations of READ-IT score are scarcely perceived by human subjects. Since the construction of this resource has been specifically designed to develop ATS systems for human target, this READ-IT variation is a fundamental parameter of PaCCSS-IT. We thus obtained about 4,450 pairs.

4 PaCCSS–IT

The unsupervised step applied to ItWaC resulted in ∼28 million of clusters with overlapping lexicon for a total of ∼35 million of single sentences. From this initial set we pruned sentences with a number of tokens <5 and >40 and clusters containing less than two sentences. We obtained 419,252 clusters for a total of ∼8,5 million of single sentences and an average number of pairs in each cluster of 1,613. Filtering the pairs according to the cosine similarity range defined in the development step, we obtained a subset of 73,142 clusters with an average of ∼112 pairs for each. The classifier with the same model tested on PAISÁ recognised about 1 million of *correct* aligned pairs. Excluding pairs below the confidence score $\geq$ 85% we obtained ∼284,000 pairs. This collection was further pruned selecting only those pairs with at least 0.2 points in terms of variation in readability score. PaCCSS–IT is the resulting resource. It is a freely available resource [2] composed of ∼63,000 pairs of sentences (∼126k sentences) ranked with respect to the readability score of the two sentences. For each pair the cosine similarity, the probability score of the classifier and the readability level of the sentences are provided.

The following sections report the evaluation and the qualitative analysis we carried out on PaCCSS–IT. The evaluation was performed to assess the reliability of sentence alignment and of the sentence ranking with respect to readability level. The qualitative analysis was focused on studying which linguistic phenomena typically related to text simplification are successfully intercepted by our approach in order to show the applicability of the resource in a ATS scenario.

4.1 Evaluation

The evaluation process was intended to calculate the accuracy of i) the automatic classification process in predicting correct sentence alignments and ii) the automatic readability ranking of each pair.

The alignment evaluation was carried out by two trained linguists who manually revised 40 pairs of randomly selected sentences for each cosine score (1,088 paired sentences). It resulted that 85% of pairs were correctly classified (i.e. 921 pairs) and precision increases as cosine grows (from 73.2% at cosine 0.65-0.69 to 90.8% at cosine 0.90-0.92).

The subset of 921 pairs correctly classified was further investigated with respect to the readability level automatically assigned. To this aim we elicited human judgements through the crowdsourcing platform CrowdFlower[3]. We collected judgements from 7 workers that were asked to rate for each pair which of the two individual sentences was simpler. We considered the majority label to be true label for each pair. Comparing the score obtained by our system with the human judgements we obtained an accuracy of 74%. Restricting the evaluation only to pairs with the same label assigned by at least five out seven annotators (i.e. 79% of the whole pairs), the system achieved an accuracy of 78%.

4.2 Qualitative Analysis

Two qualitative analyses were carried out on PaCCSS–IT. The first analysis took into account the subset of 921 revised pairs with the aim of manually investigating what kinds of sentence transformations previously observed in the literature on text simplification were intercepted by our approach. In the second one, the whole resource was automatically investigated to study how the alignment process impacts on the distribution of multi–level linguistic features correlated to sentence complexity.

4.2.1 Analysis of Simplification Operations

Following the classification of simplification operations proposed in the literature (Brunato et al., 2015; Bott and Saggion, 2014; Coster and Kauchak, 2011; Caseli et al., 2009), we identified the major types of operations occurring in the subset of revised pairs [4], namely:

Deletion: the second sentence (S) does not contain one or more than two words occurring in the first one (C):

- C: *Ma c'è un altro problema, ancora più grave*. [Lit: But there is another problem, even more serious.]

[2] http://www.italianlp.it/software-data/

[3] www.crowdflower.com

[4] In each of the following examples the first sentence (C) is the complex sentence and the second (S) the simple one. We underlined the text span affected by the operation.

- S: *Poi c'è un altro problema*. [Lit: Then there is another problem.]

Verbal Features: the two sentences differ with respect to verbal mood and tense:

- C: *I suoi libri sono stati tradotti in molte lingue*. [Lit: His books have been translated in many languages.]

- S: *I suoi libri sono tradotti in diverse lingue*. [Lit: His books have been translated in different languages.]

Lexical Substitution: the two sentences contain synonyms of words tagged with POS which were not considered in the clustering step based on POS overlapping, e.g. adjectives, adverbs:

- C: *Il colore è un rosso rubino fittissimo, quasi impenetrabile, limpido*. [Lit: The color is a rubyred very dense, almost impenetrable, clear.]

- S: *Il colore è un rosso rubino vivo quasi impenetrabile*. [Lit: The color is a bright red ruby almost impenetrable.]

Reordering: the two sentences contain a different word order both at single word (e.g. subject in pre- vs. post-verbal position) and phrase level (e.g a subordinate clause proceeds vs. follows the main clause):

- C: *Ringraziandola per la sua cortese attenzione, resto in attesa di risposta*. [Lit: Thanking you for your kind attention, I look forward to your answer.]

- S: *Resto in attesa di una risposta e ringrazio vivamente per l'attenzione*. [Lit: I look forward to your answer and I thank you greatly for your attention.]

Insertion: the second sentence contains one or *n*-words more than the first one:

- C: *In attesa di un sollecito riscontro, distinti saluti*. [Lit: Waiting for an early reply, yours faithfully.]

- S: *In attesa di un riscontro porgiamo distinti saluti*. [Lit: Waiting for a reply, we offer our regards.]

Sentence Type: the two sentences differ with respect to their form (i.e. affirmative vs. interrogative):

- C: *Quale consiglio darebbe ai genitori?* [Lit: Which advice would you give to parents?]

- S: *Diamo un consiglio ai genitori*. [Lit: Let's give an advice to parents.]

For each operation there can be different degrees of sentence transformation. For example, focusing on *Verbal Feature*, the example reported above represents a "light" transformation while a "stronger" transformation can occur when the verb changes from the conditional to the indicative mood (or vice versa), as in the following pair:

- C: *Sarebbe un grave un errore*. [Lit: It would be a serious error.]

- S: *Ma è un grave errore*. [Lit: But it is a serious error.]

Figure 4 reports the distribution of these sentence operations in the manually revised portion of the corpus. The distribution in *All cosines* shows that the two most frequent operations are deletion (30.74%) and changes affecting verbal features (26.30%). According to the literature, the deletion of redundant information (e.g. adjectives, adverbs) is one of the main phenomena typically related to reduction of complexity. Also transformations of verbal features are likely to intercept simplification operations in a language like Italian with a rich inflectional paradigm. The third most frequent operation is lexical substitution (15.52%). According to the POS filter used in the unsupervised step of sentence alignment, this operation affects morpho–syntactic categories such as e.g. adverbs, adjectives, conjunctions or prepositions which are substituted with a simpler synonym. Reordering and insertion of words or phrases are respectively the fourth and the fifth types of transformation. Reordering can be expected as a simplification strategy especially when it yields a more canonical word order. The distribution of reordering here reported, i.e. 14.24%,

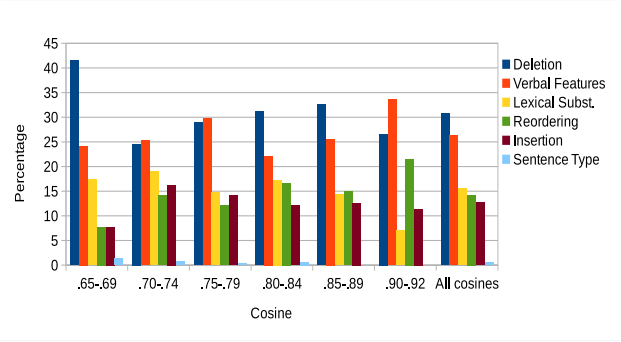


Figure 4: Distribution of sentence operations at different cosine ranges.

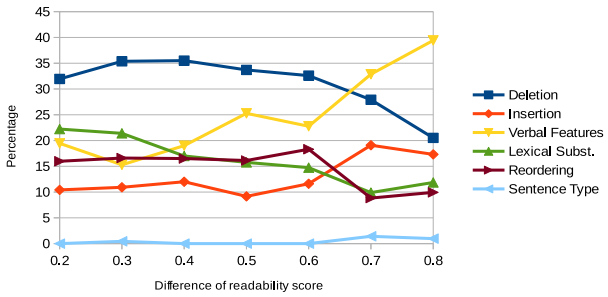


Figure 5: Distribution of sentence operations at different readability scores.

is quite high if compared to the distribution of the same operation found in hand-crafted simplified corpora where it represents about 8% of sentence operations (Brunato et al., 2015). This result gives evidence that our approach succeeds in automatically intercepting this kind of syntactic transformation. In the manually revised portion of PaCCSS–IT insertion represents 12.72% of the whole operations. Despite inserting words or phrases could make more complex a sentence, this operation is used in the simplification process e.g. when it makes explicit missing arguments in elliptical clauses more frequently used in non–standard language varieties or sublanguages such as legal language. This is the case of the heterogeneous nature of the corpus from which PaCCSS–IT derives, where documents characterized by non–canonical languages (e.g. blogs, e–mails) or domain–specific documents (e.g. administrative acts) are mixed to texts representative of more standard varieties, e.g. newspapers, novels.

Let us consider the relation between simplification operations, cosine values and readability levels. For what concerns the distribution of the operations at different cosines (Figure 4), we observe that deletion is the most frequent operation at all cosines, in particular at lower cosines i.e. $<.70$. At high cosines, i.e. $>.90$, operations affecting word order and verbal features increase. The relation between readability score and sentence operations is shown in Figure 5. Specifically, we calculated how the distribution of operations changes with increasing differences between the readability score assigned to the complex and the simple sentence of each pair. Although it is difficult to study the effect of each single operation on the readability score variation since these operations are usually applied in combination, we observe some clear tendencies. In particular, operations concerning deletion and verbal features are the most frequent ones both at lower and higher readability scores differences. However, they have an opposite distribution: transformations of verbal features increase at higher readability differences (>0.6) while deletions decrease. For what concerns the other operations, the trend is quite homogeneous along with the different readability scores. In particular, this is the case of reordering thus showing the proposed approach is able to intercept syntactic transformations which impact at different readability variations.

4.2.2 Analysis of Linguistic Phenomena

The second qualitative analysis focused on the whole resource which was searched for linguistic phenomena correlating with the process of sentence alignment. To this end, we compared the distribution of a set of different linguistic features, i.e. raw text, lexical, morpho–syntactic and syntactic, automatically extracted from the set of complex and simple sentences of PaCCSS–IT, which was previously tagged and dependency–parsed. In Table 2 we report a selection of the features with a statistically significant variation[5] between the complex and the simple sentences. As expected, the average sentence length (feature [1]) of the *Simple* sentence is lower than the *Complex* one. The higher distribution of adjectives [2], adverbs [3] and determiners [4] might be

[5]Wilcoxon's signed rank test was used to evaluate statistical significance.

related to the insertion of simple lexicon belonging to the Basic Italian Vocabulary (De Mauro, 2000). The distribution of verbal moods is also significantly correlated to a higher readability level: simple sentences have a higher percentage of indicatives [6] (a simple mood indicating a state of being or reality) and less participles [7] and gerundives [8] which are non finite moods and thus can be more ambiguous with respect to the reference. In addition, sentences classified as complex have higher parse trees [13], longer dependency links [14] and longer embedded complement chains modifying a noun [15], all features correlated with syntactic complexity (Gibson, 1998; Lin, 1986; Frazier, 1985). On the contrary, sentences classified as simple are characterised by a more canonical word order (Subject–Verb–Object in Italian) i.e. a lower distribution of post-verbal subjects [16] and of pre-verbal objects [17]. These sentences also contain a higher distribution of subordinate clauses following the main clause [18], an order easier to process.

Since syntactic features intercepting the structure of the sentence (e.g. parse tree depth and dependency length) heavily depend on the overall sentence length, we carried out an analysis only on pairs of sentences where the complex and the simple sentence have the same number of tokens (i.e. 15,958 pairs in PaCCSS–IT) and we compared how linguistic features vary between the complex and the simple sentences of these pairs. We observed that simple sentences have a more canonical position of the subject (i.e. a lower percentage distribution of post-verbal subjects: *C*: 18.14%, *S*: 15.72%) and of the object (i.e. a lower percentage distribution of pre-verbal objects: *C*: 1.52%, *S*: 1.18%). Simple sentences have also lower parsed trees (*C*: 2.42, *S*: 2.37) and shorter embedded complement chains modifying a noun (*C*: 0.27, *S*: 0.26). Since these variations cannot be due to sentence shortening, they rather follow from reordering phenomena e.g. changing from active to passive voice.

The distribution of linguistic features here reported has already been observed in hand–crafted corpora of complex and simple sentences for Italian (Brunato et al., 2015). This is a further evidence of the reliability of our method for automatically creating corpora of complex–simple sentences.

Feature	*Complex*	*Simple*	*Variation*
[1]	8.98	7.80	0.97
[2]	4.10	7.90	-3.80
[3]	9.10	10.0	-0.85
[4]	0.34	1.43	-1.10
[5]	10.70	20.30	-9.61
[6]	5.20	2.72	2.49
[7]	0.47	0.04	0.42
[8]	2.89	4.29	-1.40
[9]	79.18	80.91	-1.73
[10]	1.33	1.57	-0.24
[11]	2.03	2.14	-0.10
[12]	8.35	7.33	0.5
[13]	2.88	2.70	0.18
[14]	1.76	1.63	0.12
[15]	0.44	0.41	0.02
[16]	15.37	14.37	1.00
[17]	2.03	1.38	0.65
[18]	3.29	4.17	-0.90

Table 2: Distribution of a subset of linguistic features with statistically significant variation between the complex and simple sentences. Features [1],[13],[14],[15] are absolute values, the others are percentage distributions. All differences are significant at $p < 0.001$.

5 Conclusion

In this paper we have presented PaCCSS–IT, a corpus of complex–simple aligned sentences for Italian containing ∼63,000 paired sentences. To our knowledge, PaCCSS–IT is the biggest corpus of complex–simple aligned sentences, with the exception of English. It resulted from a new method for automatically acquiring corpora of parallel sentences able to capture structural transformations and particularly suitable for text simplification systems. A comparative analysis of the multi–level linguistic features in the complex and simple sentences showed that this method intercepts linguistic phenomena characterising simplification operations previously observed in manually–created complex–simple corpora. A main novelty of the proposed approach is that it does not rely on a large pre-existing corpus of aligned complex–simple documents like e.g. the English and Simple English Wikipedia. This makes it very appropriate for less–resourced languages. In addition, since the method does not need parallel corpora, the dimension of the web is the only limitation to the size of the corpus that could be created.

References

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. *Proceedings of the 2nd Workshop of Evalita 2009 - Evaluation of NLP and Speech Tools for Italian*, Reggio Emilia, Italy.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Regina Barzilay and Noemi Elhadad. 2003. Sentence alignment for monolingual comparable corpora. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wauter Bosma and Chris Callison-Burch. 2007. Paraphrase substitution for recognizing textual entailment. *Proceedings of the 7th International Conference on Cross-Language Evaluation Forum (CLEF)*.

Stefan Bott and Horacio Saggion. 2011. An unsupervised alignment algorithm for text simplification corpus construction. *Proceedings of the Workshop on Monolingual Text-To-Text Generation, co-located with ACL 2011*, Porland, Oregon.

Stefan Bott and Horacio Saggion. 2014. Text simplification resources for Spanish. *Language Resources and Evaluation*, 48(1):93–120.

Laetitia Brouwers, Delphine Bernhard, Anne-Laure Ligozat, and Thomas François. 2014. Syntactic sentence simplification for French. *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*.

Dominique Brunato, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2015. Design and annotation of the first Italian corpus for text simplification. *Proceedings of the 9th Linguistic Annotation Workshop (LAW'15)*, Denver, Colorado, USA.

Helena M. Caseli, Tiago F. P. Pereira, Lucia Specia, Thiago A. S. Pardo, Caroline Gasperin, and Sandra Aluísio. 2009. Building a Brazilian Portuguese parallel corpus of original and simplified texts. *Proceedings of the 10th Conference on Intelligent Text Processing and Computational Linguistics*.

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM: a library for support vector machines. *Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*.

Do Kook Choe and David McClosky. 2015. Parsing paraphrases with joint inference. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tullio De Mauro. 2000. *Il dizionario della lingua italiana*.

Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. 2011. READ–IT: assessing readability of italian texts with a view to text simplification. *Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, Edinburgh, UK.

Felice Dell'Orletta. 2009. Ensemble system for Part-of-Speech tagging. *Proceedings of Evalita'09, Evaluation of NLP and Speech Tools for Italian*, Reggio Emilia, December.

William Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*.

Lyn Frazier. 1985. Syntactic complexity. *Natural Language Parsing*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Sigrid Klerke and Anders Søgaard. 2012. DSim, a Danish parallel corpus for text simplification. *Proceedings of Language Resources and Evaluation Conference (LREC)*.

Nikolina Koleva, Andrea Horbach, Alexis Palmer, Simon Ostermann, and Manfred Pinkal. 2014. Paraphrase detection for short answer scoring. *Proceedings of the third workshop on NLP for computer-assisted language learning*.

Dekan Lin. 1986. On the structural complexity of natural language sentences. *Proceedings of COLING 1996*.

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Dittmann Henrik, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISA corpus of Italian web texts. *Proceedings of the 9th Web as Corpus Workshop (WAC-9) EACL*.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Rani Nelken and Stuart M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, 3–7 April.

Advaith Siddharthan and Mandya Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*.

Sowmya Vajjala and Detmar Meurers. 2016. Readability-based sentence ranking for evaluating text simplification. *arXiv*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. *Proceedings of the 23rd international conference on computational linguistics*.

Discourse Parsing with Attention-based Hierarchical Neural Networks

Qi Li **Tianshi Li** **Baobao Chang**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, China
Collaborative Innovation Center for Language Ability, Xuzhou, 221009, China
qi.li@pku.edu.cn lts_417@hotmail.com chbb@pku.edu.cn

Abstract

RST-style document-level discourse parsing remains a difficult task and efficient deep learning models on this task have rarely been presented. In this paper, we propose an attention-based hierarchical neural network model for discourse parsing. We also incorporate tensor-based transformation function to model complicated feature interactions. Experimental results show that our approach obtains comparable performance to the contemporary state-of-the-art systems with little manual feature engineering.

1 Introduction

A document is formed by a series of coherent text units. Document-level discourse parsing is a task to identify the relations between the text units and to determine the structure of the whole document the text units form. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the most influential discourse theories. According to RST, the discourse structure of a document can be represented by a Discourse Tree (DT). Each leaf of a DT denotes a text unit referred to as an Elementary Discourse Unit (EDU) and an inner node of a DT represents a text span which is constituted by several adjacent EDUs. DTs can be utilized by many NLP tasks including automatic document summarization (Louis et al., 2010; Marcu, 2000), question-answering (Verberne et al., 2007) and sentiment analysis (Somasundaran, 2010) etc.

Much work has been devoted to the task of RST-style discourse parsing and most state-of-the-art approaches heavily rely on manual feature engineering (Joty et al., 2013; Feng and Hirst, 2014; Ji and Eisenstein, 2014). While neural network models have been increasingly focused on for their ability to automatically extract efficient features which reduces the burden of feature engineering, there is little neural network based work for RST-style discourse parsing except the work of Li et al. (2014a). Li et al. (2014a) propose a recursive neural network model to compute the representation for each text span based on the representations of its subtrees. However, vanilla recursive neural networks suffer from gradient vanishing for long sequences and the normal transformation function they use is weak at modeling complicated interactions which has been stated by Socher et al. (2013). As many documents contain more than a hundred EDUs which form quite a long sequence, those weaknesses may lead to inferior results on this task.

In this paper, we propose to use a hierarchical bidirectional Long Short-Term Memory (bi-LSTM) network to learn representations of text spans. Comparing with vanilla recursive/recurrent neural networks, LSTM-based networks can store information for a long period of time and don't suffer from gradient vanishing problem. We apply a hierarchical bi-LSTM network because the way words form an EDU and EDUs form a text span is different and thus they should be modeled separately and hierarchically. On top of that, we apply attention mechanism to attend over all EDUs to pick up prominent semantic information of a text span. Besides, we use tensor-based transformation function to model complicated feature interactions and thus it can produce

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 362–371,
Austin, Texas, November 1-5, 2016.

combinatorial features.

We summarize contributions of our work as follows:

- We propose to use a hierarchical bidirectional LSTM network to learn the compositional semantic representations of text spans, which naturally matches and models the intrinsic hierarchical structure of text spans.
- We extend our hierarchical bi-LSTM network with attention mechanism to allow the network to focus on the parts of input containing prominent semantic information for the compositional representations of text spans and thus alleviate the problem caused by the limited memory of LSTM for long text spans.
- We adopt a tensor-based transformation function to allow explicit feature interactions and apply tensor factorization to reduce the parameters and computations.
- We use two level caches to intensively accelerate our probabilistic CKY-like parsing process.

The rest of this paper is organized as follows: Section 2 gives the details of our parsing model. Section 3 describes our parsing algorithm. Section 4 gives our training criterion. Section 5 reports the experimental results of our approach. Section 6 introduces the related work. Conclusions are given in section 7.

2 Parsing Model

Given two successive text spans, our parsing model evaluates the probability to combine them into a larger span, identifies which one is the nucleus and determines what is the relation between them. As with the work of Ji and Eisenstein (2014), we set three classifiers which share the same features as input to deal with those problems. The whole parsing model is shown in Figure 1. Three classifiers are on the top. The semantic representations of the two given text spans which come from the output of attention-based hierarchical bi-LSTM network with tensor-based transformation function is the main part of input to the classifiers. Additionally, following the previous practice of Li et al. (2014a), a small set of handcrafted features is introduced to enhance the model.

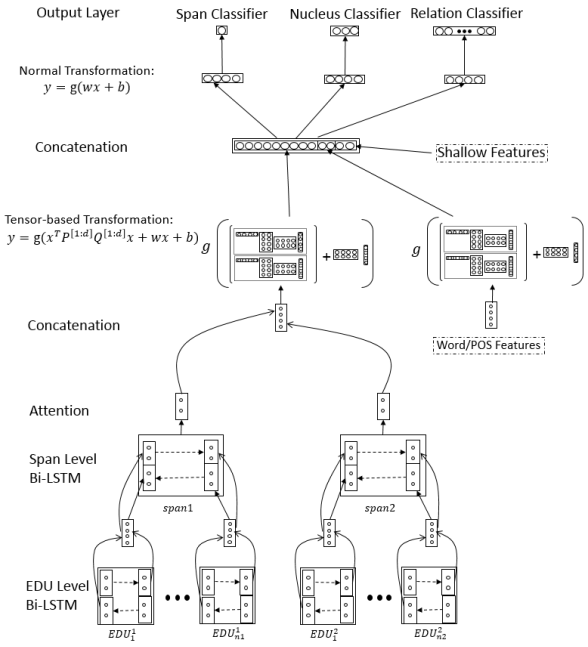


Figure 1: Schematic structure of our parsing model.

2.1 Hierarchical Bi-LSTM Network for Text Span Representations

Long Short-Term Memory (LSTM) networks have been successfully applied to a wide range of NLP tasks for the ability to handle long-term dependencies and to mitigate the curse of gradient vanishing (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014; Rocktäschel et al., 2015; Hermann et al., 2015). A basic LSTM can be described as follows. A sequence $\{x_1, x_2, ..., x_n\}$ is given as input. At each time-step, the LSTM computation unit takes in one token x_t as input and it keeps some information in a cell state C_t and gives an output h_t. They are calculated in this way:

$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i) \quad (1)$$

$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \quad (2)$$

$$\tilde{C}_t = tanh(W_C[h_{t-1}; x_t] + b_C) \quad (3)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o) \quad (5)$$

$$h_t = o_t \odot tanh(C_t) \quad (6)$$

where $W_i, b_i, W_f, b_f, W_c, b_C, W_o, b_o$ are LSTM parameters, $\odot$ denotes element-wise product and σ denotes sigmoid function. The output at the last token, i.e., h_n is taken as the representation of the whole sequence.

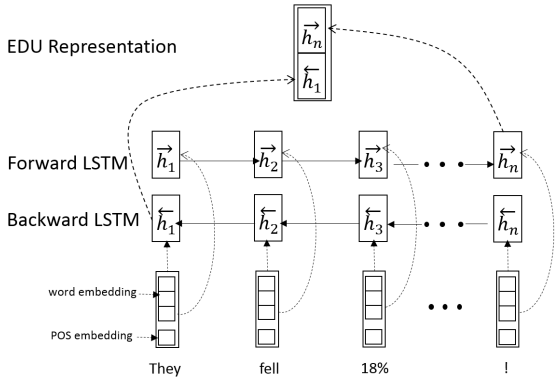


Figure 2: Bi-LSTM for computing the compositional semantic representation of an EDU.

Since an EDU is a sequence of words, we derive the representation of an EDU from the sequence constituted by concatenation of word embeddings and the POS tag embeddings of the words as Figure 2 shows. Previous work on discourse parsing tends to extract some features from the beginning and end of text units partly because discourse clues such as discourse markers(e.g., because, though) are often situated at the beginning or end of text units(Feng and Hirst, 2014; Ji and Eisenstein, 2014; Li et al., 2014a; Li et al., 2014b; Heilman and Sagae, 2015). Considering the last few tokens of a sequence normally have more influence on the representation of the whole sequence learnt with LSTM because they get through less times of forget gate from the LSTM computation unit, to effectively capture the information from both beginning and end of an EDU, we use bidirectional LSTM to learn the representation of an EDU. In other words, one LSTM takes the word sequence in forward order as input, the other takes the word sequence in reversed order as input. The representation of a sequence is the concatenation of the two vector representations calculated by the two LSTMs.

Since a text span is a sequence of EDUs, its meaning can be computed from the meanings of the EDUs. So we use another bi-LSTM to derive the compositional semantic representation of a text span from the EDUs it contains. The two bi-LSTM networks form a hierarchical structure as Figure 1 shows.

2.2 Attention

The representation of a sequence computed by bi-LSTMs is always a vector with fixed dimension despite the length of the sequence. Thus when dealing with a text span with hundreds of EDUs, bi-LSTM may not be enough to capture the whole semantic information with its limited output vector dimension. Attention mechanism can attend over the output at every EDU with global context and pick up prominent semantic information and drop the subordinate information for the compositional representation of the span, so we employ attention mechanism to alleviate the problem caused by the limited memory of LSTM networks. The attention mechanism is inspired by the work of Rocktäschel et al. (2015). Our attention-based bi-LSTM network is shown in Figure 3.

We combine the last outputs of the span level bi-LSTM to be $h_s = [\overrightarrow{h}_{e_n}, \overleftarrow{h}_{e_1}]$. We also combine the outputs of the two LSTM at every EDU of the span: $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$ and thus get a matrix $H = [h_1; h_2; ...; h_n]^T$. Taking $H \in \mathbb{R}^{d \times n}$ and $h_s \in \mathbb{R}^d$ as inputs, we get a vector $\alpha \in \mathbb{R}^n$ standing for weights of EDUs to the text span and use it to get a weighted representation of the span $r \in \mathbb{R}^d$:

$$M = tanh(W_y H + W_l h_s \otimes e_n) \quad (7)$$

$$\alpha = softmax(w_\alpha^T M) \quad (8)$$

$$r = H\alpha \quad (9)$$

where $\otimes$ denotes Cartesian product , $M \in \mathbb{R}^{k \times n}$, e_n is a n dimensional vector of all 1s and we use the Cartesian product $W_l h_s \otimes e_n$ to repeat the result of $W_l h_s$ n times in column to form a matrix and $W_y \in \mathbb{R}^{k \times d}, W_l, \in \mathbb{R}^{k \times d}, w_\alpha \in \mathbb{R}^k$ are parameters.

We synthesize the information of r and h_s to get the final representation of the span:

$$w_h = \sigma(W_{hr} r + W_{hh} h_s) \quad (10)$$

$$h = w_h \odot h_s + (1 - w_h) \odot r \quad (11)$$

where $W_{hr}, W_{hh} \in \mathbb{R}^{d \times d}$ are parameters, $w_h \in \mathbb{R}^d$ is a computed vector representing the element-wise weight of h_s and the element-wise weighted summation $h \in \mathbb{R}^d$ is the final representation of the text span computed by the attention-based bidirectional LSTM network.

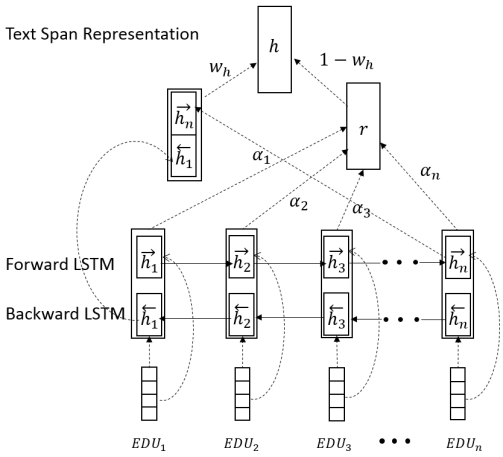


Figure 3: Attention-based bi-LSTM for computing the compositional semantic representation of a text span.

2.3 Classifiers

We concatenate the representations of the two given spans: $h = [h_{s1}, h_{s2}]$ and feed h into a full connection hidden layer to obtain a higher level representation v which is the input to the three classifiers:

$$v = Relu(W_h[h_{s1}, h_{s2}] + b_h) \quad (12)$$

For each classifier, we firstly transform $v \in \mathbb{R}^l$ into a hidden layer:

$$v_{sp} = Relu(W_{hs}v + b_{hs}) \quad (13)$$
$$v_{nu} = Relu(W_{hn}v + b_{hn}) \quad (14)$$
$$v_{rel} = Relu(W_{hr}v + b_{hr}) \quad (15)$$

where $W_{hs}, W_{hn}, W_{hr} \in \mathbb{R}^{h \times l}$ are transformation matrices and $b_{hs}, b_{hn}, b_{hr} \in \mathbb{R}^h$ are bias vectors.

Then we feed these vectors into the respective output layer:

$$y_{sp} = \sigma(w_s v_{sp} + b_s) \quad (16)$$
$$y_{nu} = softmax(W_n v_{nu} + b_n) \quad (17)$$
$$y_{rel} = softmax(W_r v_{rel} + b_r) \quad (18)$$

where $w_s \in \mathbb{R}^h, b_s \in \mathbb{R}, W_n \in \mathbb{R}^{3 \times h}, W_n \in \mathbb{R}^{3 \times h}, b_n \in \mathbb{R}^3, W_r \in \mathbb{R}^{n_r \times h}, b_n \in \mathbb{R}^{n_r}$ are parameters and n_r is the number of different discourse relations.

The first classifier is a binary classifier which outputs the probability the two spans should be combined. The second classifier is a multiclass classifier which identifies the nucleus to be span 1, span 2 or both. The third classifier is also a multiclass classifier which determines the relation between the two spans.

2.4 Tensor-based Transformation

Tensor-based transformation function has been successfully utilized in many tasks to allow complicated interaction between features (Sutskever et al., 2009; Socher et al., 2013; Pei et al., 2014). Based on the intuition that allowing complicated interaction between the features of the two spans may help to identify how they are related, we adopt tensor-based transformation function to strengthen our model.

A tensor-based transformation function on $x \in \mathbb{R}^{d_1}$ is as follows:

$$y = Wx + x^T T^{[1:d_2]} x + b \quad (19)$$
$$y_i = \sum_j W_{ij} x_j + \sum_{j,k} T^{[i]}_{j,k} x_j x_k + b_i \quad (20)$$

where $y \in \mathbb{R}^{d_2}$ is the output vector, $y_i \in \mathbb{R}$ is the ith element of y, $W \in \mathbb{R}^{d_2 \times d_1}$ is the transformation matrix, $T^{[1:d_2]} \in \mathbb{R}^{d_1 \times d_1 \times d_2}$ is a 3rd-order transformation tensor. A normal transformation function in neural network models only has the first term Wx with the bias term. It means for normal transformation function each unit of the output vector is the weighted summation of the input vector and this only allows additive interaction between the units of the input vector. With the tensor multiplication term, each unit of the output vector is augmented with the weighted summation of the multiplication of the input vector units and thus we incorporate multiplicative interaction between the units of the input vector.

Inevitably, the incorporation of tensor leads to side effects which include the increase in parameter number and computational complexity. To remedy this, we adopt tensor factorization in the same way as Pei et al. (2014): we use two low rank matrices to approximate each tensor slice $T^{[i]} \in \mathbb{R}^{d_1 \times d_1}$:

$$T^{[i]} \Rightarrow P^{[i]} Q^{[i]} \quad (21)$$

where $P^{[i]} \in \mathbb{R}^{d_1 \times r}$, $Q^{[i]} \in \mathbb{R}^{r \times d_1}$ and $r \ll d_1$. In this way, we drastically reduce parameter number and computational complexity.

We apply the factorized tensor-based transformation function to the combined text span representation $h = [h_{s1}, h_{s2}]$ to make the features of the two spans explicitly interact with each other:

$$v = Relu(W_h[h_{s1}, h_{s2}] + [h_{s1}, h_{s2}]^T P_h^{[1:d]} Q_h^{[1:d]} [h_{s1}, h_{s2}] + b_h) \quad (22)$$

Comparing with Eq. 12, the transformation function is added with a tensor term.

2.5 Handcrafted Features

Most previously proposed state-of-the-art systems heavily rely on handcrafted features (Hernault et al., 2010; Feng and Hirst, 2014; Joty et al., 2013; Ji and Eisenstein, 2014; Heilman and Sagae, 2015). Li et al. (2014a) show that some basic features are still necessary to get a satisfactory result for their recursive deep model. Following their practice, we adopt minimal basic features which are utilized by most systems to further strengthen our model. We list these features in Table 1. We apply the factorized tensor-based transformation function to Word/POS features to allow more complicated interaction between them.

3 Parsing Algorithm

In this section, we describe our parsing algorithm which utilizes the parsing model to produce the global optimal DT for a segmented document.

3.1 Probabilistic CKY-like Algorithm

We adopt a probabilistic CKY-like bottom-up algorithm which is also adopted in (Joty et al., 2013; Li et al., 2014a) to produce a DT for a document. This parsing algorithm is a dynamic programming algorithm and produces the global optimal DT with our parsing model. Given a text span which is constituted by $[e_i, e_{i+1}, ..., e_j]$ and the possible subtrees of $[e_i, e_{i+1}, ..., e_k]$ and $[e_{k+1}, e_{k+2}, ..., e_j]$ for all $k \in \{i, i+1, ..., j-1\}$ with their probabilities, we choose k and combine the corresponding subtrees to form a combined DT with the following recurrence formula:

$$k = \arg\max_k \{P_{sp}(i, k, j) P_{i,k} P_{k+1,j}\} \quad (23)$$

where $P_{i,k}$ and $P_{k+1,j}$ are the probabilities of the most probable subtrees of $[e_i, e_{i+1}, ..., e_k]$ and $[e_{k+1}, e_{k+2}, ..., e_j]$ respectively, $P_{sp}(i, k, j)$ is the probability which is predicted by our parsing model to combine those two subtrees to form a DT.

The probability of the most probable DT of $[e_i, e_{i+1}, ..., e_j]$ is:

$$P_{i,j} = \max_k \{P_{sp}(i, k, j) P_{i,k} P_{k+1,j}\} \quad (24)$$

3.2 Parsing Acceleration

Computational complexity of the original probabilistic CKY-like algorithm is $O(n^3)$ where n is the number of EDUs of the document. But in this work, given each pair of text spans, we compute the representations of them with hierarchical bi-LSTM network at the expense of an additional $O(n)$ computations. So the computational complexity of our parser becomes $O(n^4)$ and it is unacceptable for long documents. However, most computations are duplicated, so we use two level caches to drastically accelerate parsing.

Firstly, we cache the outputs of the EDU level bi-LSTM which are the semantic representations of EDUs. As for the forward span level LSTM, after we get the semantic representation of a span, we cache it too and use it to compute the representation of an extended span. For example, after we get the representation of span constituted by $[e_1, e_2, e_3]$, we take it with semantic representation of e_4 to compute the representation of the span constituted by $[e_1, e_2, e_3, e_4]$ in one LSTM computation step. For the backward span level LSTM, we do it the same way just in reversed order. Thus we decrease the computational complexity of computing the semantic representations for all possible span pairs which is the most time-consuming part of the original parsing process from $O(n^4)$ to $O(n^2)$.

Secondly, it can be seen that before we apply $Relu$ to the tensor-based transformation function, many calculations from the two spans which include a large part of tensor multiplication are independent. The multiplication between the elements of the representations of the two spans caused by the tensors and the element-wise non-linear activation function $Relu$ terminate the independence between them. So we can further cache the independent calculation results before $Relu$ operation for each span. Thus we decrease the computational complexity of a large part of tensor-based transformation from $O(n^3)$ to

Word/POS Features
One-hot representation of the first two words and of the last word of each span.
One-hot representation of POS tags of the first two words and of the last word of each span.
Shallow Features
Number of EDUs of each span.
Number of words of each span.
Predicted relations of the two subtrees' roots.
Whether each span is included in one sentence.
Whether both spans are included in one sentence.

Table 1: Handcrafted features used in our parsing model.

$O(n^2)$ which is the second time-consuming part of the original parsing process.

The remaining $O(n^3)$ computations include a little part of tensor-based transformation computations, $Relu$ operation and the computations from the three classifiers. These computations take up only a little part of the original parsing model computations and thus we greatly accelerate our parsing process.

4 Max-Margin Training

We use Max-Margin criterion for our model training. We try to learn a function that maps: $X \rightarrow Y$, where X is the set of documents and Y is the set of possible DTs. We define the loss function for predicting a DT $\hat{y_i}$ given the correct DT y_i as:

$$\triangle(y_i, \hat{y_i}) = \sum_{r \in \hat{y_i}} \kappa \mathbf{1}\{r \notin y_i\} \tag{25}$$

where r is a span specified with nucleus and relation in the predicted DT, κ is a hyperparameter referred to as discount parameter and $\mathbf{1}$ is indicator function. We expect the probability of the correct DT to be a larger up to a margin to other possible DTs:

$$Prob(x, y_i) \geq Prob(x_i, \hat{y_i}) + \triangle(y_i, \hat{y_i}) \tag{26}$$

The objective function for m training examples is as follows:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} l_i(\theta), where \tag{27}$$

$$l_i(\theta) = \max_{\hat{y_i}} (Prob(x_i, \hat{y_i}) + \triangle(y_i, \hat{y_i})) - Prob(x_i, y_i) \tag{28}$$

where θ denotes all the parameters including our neural network parameters and all embeddings.

The probabilities of the correct DTs increase and the probabilities of the most probable incorrect DTs decrease during training. We adopt Adadelta (Zeiler, 2012) with mini-batch to minimize the objective function and set the initial learning rate to be 0.012.

5 Experiments

We evaluate our model on RST Discourse Treebank[1] (RST-DT) (Carlson et al., 2003). It is partitioned into a set of 347 documents for training and a set of 38 documents for test. Non-binary relations are converted into a cascade of right-branching binary relations. The standard metrics of RST-style discourse parsing evaluation include blank tree structure referred to as span (S), tree structure with nuclearity (N) indication and tree structure with rhetorical relation (R) indication. Following other RST-style discourse parsing systems, we evaluate the relation metric in 18 coarse-grained relation classes. Since our work focus does not include EDU segmentation, we evaluate our system with gold-standard EDU segmentation and we apply the same setting on this to other discourse parsing systems for fair comparison.

5.1 Experimental Setup

The dimension of word embeddings is set to be 50 and the dimension of POS embeddings is set to be 10. We pre-trained the word embeddings with GloVe (Pennington et al., 2014) on English Gigaword[2] and we fine-tune them during training. Considering some words are pretrained by GloVe but

[1] https://catalog.ldc.upenn.edu/LDC2002T07

[2] https://catalog.ldc.upenn.edu/LDC2011T07

don't appear in the RST-DT training set, we want to use their embeddings if they appear in test set. Following Kiros et al. (2015), we expand our vocabulary with those words using a matrix $W \in \mathbb{R}^{50\times 50}$ that maps word embeddings from the pre-trained word embedding space to the fine-tuned word embedding space. The objective function for training the matrix W is as follows:

$$\min_{W,b} ||V_{tuned} - V_{pretrained}W - b||_2^2 \qquad (29)$$

where $V_{tuned}, V_{pretrained} \in \mathbb{R}^{|V|\times 50}$ contain fine-tuned and pre-trained embeddings of words appearing in training set respectively, $|V|$ is the size of RST-DT training set vocabulary and b is the bias term also to be trained.

We lemmatize all the words appeared and represent all numbers with a special token. We use Stanford CoreNLP toolkit (Manning et al., 2014) to preprocess the text including lemmatization, POS tagging etc. We use Theano library (Bergstra et al., 2010) to implement our parsing model. We randomly initialize all parameters within (-0.012, 0.012) except word embeddings. We adopt dropout strategy (Hinton et al., 2012) to avoid overfitting and we set the dropout rate to be 0.3.

5.2 Results and Analysis

To show the effectiveness of the components incorporated into our model, we firstly test the performance of the basic hierarchical bidirectional LSTM network without attention mechanism (ATT), tensor-based transformation (TE) and handcrafted features (HF). Then we add them successively. The results are shown in Table 2.

The performance is improved by adding each component to our basic model and that shows the effectiveness of attention mechanism and tensor-based transformation function. Even without handcrafted features, the performance is still competitive. It indicates that the semantic representations of text spans produced by our attention-based hierarchical bi-LSTM network are effective and the handcrafted features are complementary to semantic representations produced by the network.

We also experiment without mapping the OOV word embeddings and use the same embedding for all OOV words. The result is shown in Table 3. Without mapping the OOV word embeddings the performance decreases slightly, which demonstrates that the relation between pre-trained embedding space and the fine-tuned embedding space can be learnt and it is beneficial to train a matrix to transform OOV word embeddings from the pre-trained embedding space to the fine-tuned embedding space.

System Setting	S	N	R
Basic	82.7	69.7	55.6
Basic+ATT	83.6*	70.2*	56.0*
Basic+ATT+TE	84.2*	70.4	56.3*
Basic+ATT+TE+HF	85.8*	71.1*	58.9*

Table 2: Performance comparison for different settings of our system on RST-DT. 'Basic' denotes the basic hierarchical bidirectional LSTM network; '+ATT' denotes adding attention mechanism; '+TE' denotes adopting tensor-based transformation; '+HF' denotes adding handcrafted features. * indicates statistical significance in t-test compared to the result in the line above ($p < 0.05$).

System Setting	S	N	R
Without OOV mapping	85.1	70.7	58.2
Full version	85.8*	71.1*	58.9*

Table 3: Performance comparison for whether to map OOV embeddings.

We compare our system with other state-of-the-art systems including (Joty et al., 2013; Ji and Eisenstein, 2014; Feng and Hirst, 2014; Li et al., 2014a; Li et al., 2014b; Heilman and Sagae, 2015). Systems proposed by Joty et al. (2013), Heilman (2015) and Feng and Hirst (2014) are all based on variants of CRFs. Ji and Eisenstein (2014) use a projection matrix acting on one-hot representations of features to learn representations of text spans and build Support Vector Machine (SVM) classifier on them. Li et al. (2014b) adopt dependency parsing methods to deal with this task. These systems are all based on handcrafted features. Li et al. (2014a) adopt a recursive deep model and use some basic handcrafted features to improve their performances which has been stated before.

Table 4 shows the performance for our system and those systems. Our system achieves the best result in span and relatively lower performance in nucleus and relation identification comparing with the corresponding best results but still better than

System	S	N	R
Joty et al. (2013)	82.7	68.4	55.7
Ji and Eisenstein (2014)	82.1	71.1	**61.6**
Feng and Hirst (2014)	**85.7**	71.0	58.2
Li et al. (2014a)	84.0	70.8	58.6
Li et al. (2014b)	83.4	**73.8**	57.8
Heilman and Sagae (2015)	83.5	68.1	55.1
Ours	**85.8**	71.1	58.9
Human	88.7	77.7	65.8

Table 4: Performance comparison with other state-of-the-art systems on RST-DT.

System	S	N	R
Li et al. (2014a) (no feature)	82.4	69.2	56.8
Ours (no feature)	84.2	70.4	56.3

Table 5: Performance comparison with the deep learning model proposed in Li et al. (2014a) without handcrafted features.

most systems. No system achieves the best result on all three metrics. To further show the effectiveness of the deep learning model itself without handcrafted features, we compare the performance between our model and the model proposed by Li et al. (2014a) without handcrafted features and the results are shown in Table 5. It shows our overall performance outperforms the model proposed by Li et al. (2014a) which illustrates our model is effective.

Table 6 shows an example of the weights (W) of EDUs (see Eq. 8) derived from our attention model. For span1 the main semantic meaning is expressed in EDU32 under the condition described in EDU31. Besides, it is EDU32 that explicitly manifests the contrast relation between the two spans. As can be seen, our attention model assigns less weight to EDU30 and focuses more on EDU32 which is reasonable according to our analysis above.

Span1 (EDU30∼EDU32)	W
That means that	0.13
if the offense deals with one part of the business,	0.38
you don't attempt to seize the whole business;	0.49
Span2 (EDU33)	**W**
you attempt to seize assets related to the crime,	1.0

Table 6: An example of the weights derived from our attention model. The relation between span1 and span2 is Contrast.

6 Related Work

Two most prevalent discourse parsing treebanks are RST Discourse Treebank (RST-DT) (Carlson et al., 2003) and Penn Discourse TreeBank (PDTB) (Prasad et al., 2008). We evaluate our system on RST-DT which is annotated in the framework of Rhetorical Structure Theory (Mann and Thompson, 1988). It consists of 385 Wall Street Journal articles and is partitioned into a set of 347 documents for training and a set of 38 documents for test. 110 fine-grained and 18 coarse-grained relations are defined on RST-DT. Parsing algorithms published on RST-DT can mainly be categorized as shift-reduce parsers and probabilistic CKY-like parsers. Shift-reduce parsers are widely used for their efficiency and effectiveness and probabilistic CKY-like parsers lead to the global optimal result for the parsing models. State-of-the-art systems belonging to shift-reduce parsers include (Heilman and Sagae, 2015; Ji and Eisenstein, 2014). Those belonging to probabilistic CKY-like parsers include (Joty et al., 2013; Li et al., 2014a). Besides, Feng and Hirst (2014) adopt a greedy bottom-up approach as their parsing algorithm. Lexical, syntactic, structural and semantic features are extracted in these systems. SVM and variants of Conditional Random Fields (CRFs) are mostly used in these models. Li et al. (2014b) distinctively propose to use dependency structure to represent the relations between EDUs. Recursive deep model proposed by Li et al. (2014a) has been the only proposed deep learning model on RST-DT.

Incorporating attention mechanism into RNN (e.g., LSTM, GRU) has been shown to learn better representation by attending over the output vectors and picking up important information from relevant positions of a sequence and this approach has been utilized in many tasks including neural machine translation (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Hermann et al., 2015), text entailment recognition (Rocktäschel et al., 2015) etc. Some work also uses tensor-based transformation function to make stronger interaction between features and learn combinatorial features and they get performance boost in their tasks (Sutskever et

al., 2009; Socher et al., 2013; Pei et al., 2014).

7 Conclusion

In this paper, we propose an attention-based hierarchical neural network for discourse parsing. Our attention-based hierarchical bi-LSTM network produces effective compositional semantic representations of text spans. We adopt tensor-based transformation function to allow complicated interaction between features. Our two level caches accelerate parsing process significantly and thus make it practical. Our proposed system achieves comparable results to state-of-the-art systems. We will try extending attention mechanism to obtain the representation of a text span by referring to another text span at minimal additional cost.

Acknowledgments

We thank the reviewers for their instructive feedback. We also thank Jiwei Li for his helpful discussions. This work is supported by National Key Basic Research Program of China under Grant No.2014CB340504 and National Natural Science Foundation of China under Grant No.61273318. The Corresponding author of this paper is Baobao Chang.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521.

Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *CoRR*, abs/1505.02425.

Karl Moritz Hermann, Tomá s Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Hugo Hernault, Helmut Prendinger, David A DuVerle, and Mitsuru Ishizuka. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24.

Shafiq R. Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL*.

Daniel Jurafsky and James H Martin. 2008. Speech and language processing, chapter 14. In *Prentice Hall*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014a. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014b. Text-level discourse dependency parsing. In *ACL (1)*, pages 25–35.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL (1)*, pages 293–303.

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

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomá s Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Swapna Somasundaran. 2010. *Discourse-level relations for Opinion Analysis*. Ph.D. thesis, University of Pittsburgh.

Ilya Sutskever, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2009. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–736. ACM.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Multi-view Response Selection for Human-Computer Conversation

Xiangyang Zhou[1]*, Daxiang Dong[1]*, Hua Wu[1], Shiqi Zhao[1],
Dianhai Yu[1,2], Hao Tian[1,2], Xuan Liu[1] and **Rui Yan[1]**
[1]Baidu Inc., Beijing, China
[2]School of Information Science and Technology,
University of Science and Technology of China
{zhouxiangyang, dongdaxiang, wu_hua, zhaoshiqi,
yudianhai, tianhao, liuxuan, yanrui}@baidu.com

Abstract

In this paper, we study the task of response selection for multi-turn human-computer conversation. Previous approaches take word as a unit and view context and response as sequences of words. This kind of approaches do not explicitly take each utterance as a unit, therefore it is difficult to catch utterance-level discourse information and dependencies. In this paper, we propose a multi-view response selection model that integrates information from two different views, i.e., word sequence view and utterance sequence view. We jointly model the two views via deep neural networks. Experimental results on a public corpus for context-sensitive response selection demonstrate the effectiveness of the proposed multi-view model, which significantly outperforms other single-view baselines.

1 Introduction

Selecting a potential response from a set of candidates is an important and challenging task for open-domain human-computer conversation, especially for the retrieval-based human-computer conversation. In general, a set of candidate responses from the indexed conversation corpus are retrieved, and then the best one is selected from the candidates as the system's response (Ji et al., 2014).

Previous Deep Neural Network (DNN) based approaches to response selection represent context and response as two embeddings. The response is selected based on the similarity of these two embeddings (Lowe et al., 2015; Kadlec et al., 2015). In these work, context and response are taken as two separate word sequences without considering the relationship among utterances in the context and response. The response selection in these models is largely influenced by word-level information. We called this kind of models as *word sequence model* in this paper. Besides word-level dependencies, utterance-level semantic and discourse information are also very important to catch the conversation topics to ensure coherence (Grosz and Sidner, 1986). For example an utterance can be an affirmation, negation or deduction to the previous utterances, or starts a new topic for discussion. This kind of utterance-level information is generally ignored in word sequence model, which may be helpful for selecting the next response. Therefore, it is necessary to take each utterance as a unit and model the context and response from the view of utterance sequence.

This paper proposes a multi-view response selection model, which integrates information from both **word sequence view** and **utterance sequence view**. Our assumption is that each view can represent relationships between context and response from a particular aspect, and features extracted from the word sequence and the utterance sequence provide complementary information for response selection. An effective integration of these two views is expected to improve the model performance. To the best of our knowledge, this is the first work to improve the response selection for multi-turn human-computer conversation in a multi-view manner.

We evaluate the performance of the multi-view response selection model on a public corpus containing about one million context-response-label triples.

*These two authors contributed equally

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 372–381,
Austin, Texas, November 1-5, 2016.

This corpus was extracted from an online chatting room for Ubuntu troubleshooting, which is called the Ubuntu Corpus in this paper (Lowe et al., 2015). Experimental results show that the proposed multi-view response selection model significantly outperforms the current best single-view models for multi-turn human-computer conversation.

The rest of this paper is organized as follows. In Section 2, we briefly introduce related works. Then we move on to a detailed description of our model in Section 3. Experimental results are described in Section 4. Analysis of our models is shown in Section 5. We conclude the paper in Section 6.

2 Related Work

2.1 Conversation System

Establishing a machine that can interact with human beings via natural language is one of the most challenging problems in Artificial Intelligent (AI). Early studies of conversation models are generally designed for specific domain, like booking restaurant, and require numerous domain knowledge as well as human efforts in model design and feature engineering (Walker et al., 2001). Hence it is too costly to adapt those models to other domains. Recently leveraging "big dialogs" for open domain conversation draws increasing research attentions. One critical issue for open domain conversation is to produce a reasonable response. Responding to this challenge, two promising solutions have been proposed: 1) retrieval-based model which selects a response from a large corpus (Ji et al., 2014; Yan et al., 2016; Yan et al.,). 2) generation-based model which directly generates the next utterance (Wen et al., 2015a; Wen et al., 2015b).

2.2 Response Selection

Research on response selection for human-computer conversation can be classified into two branches, i.e., single-turn and multi-turn response selection. Single-turn models only leverage the last utterance in the context for selecting resposne and most of them take the word sequence view. Lu and Li (2013) proposed a DNN-based matching model for response selection. Hu et al., (2014) improved the performance using Convolutional Neural Networks (CNN) (LeCun et al., 1989). In 2015, a further study conducted by Wang et al. (2015a) achieved better results using tree structures as the input of a DNN model. Nevertheless, those models built for single-turn response selection ignore the whole context information, which makes it difficult to be implemented in the multi-turn response selection tasks.

On the other hand, research on multi-turn response selection usually takes the whole context into consideration and views the context and response as word sequences. Lowe et al., (2015) proposed a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) based response selection model for multi-turn conversation, where words from context and response are modeled with LSTM. The selection of a response is based on the similarity of embeddings between the context and response. Similar to the work of Lowe et al., Kadlec et al., (2015) replaced LSTM with Temporal Convolutional Neural Networks (TCNN) (Kim, 2014) and Bidirect-LSTM. Their experimental results show that models with LSTM perform better than other neural networks. However, the utterance-level discourse information and dependencies have been left out in these studies since they view the context and response as word sequences.

2.3 Response Generation

Another line of related research focuses on generating responses for human-computer conversation. Ritter et al., (2011) trained a phrase-based statistical machine translation model on a corpus of utterance pairs extracted from Twitter human-human conversation and used it as a response generator for single-turn conversation. Vinyals and Le (2015) regarded single-turn conversation as a sequence-to-sequence problem and proposed an encoder-decoder based response generation model, where the post response is first encoded using LSTM and its embedding used as the initialization state of another LSTM to generate the response. Shang et al., (2015) improved the encoder-decoder based model using attention signals. Sordoni et al., (2015) proposed a context-sensitive response generation model, where the context is represented by bag-of-words and fed into a recurrent language model to generate the next response.

In this paper, we focused on the task of response selection.

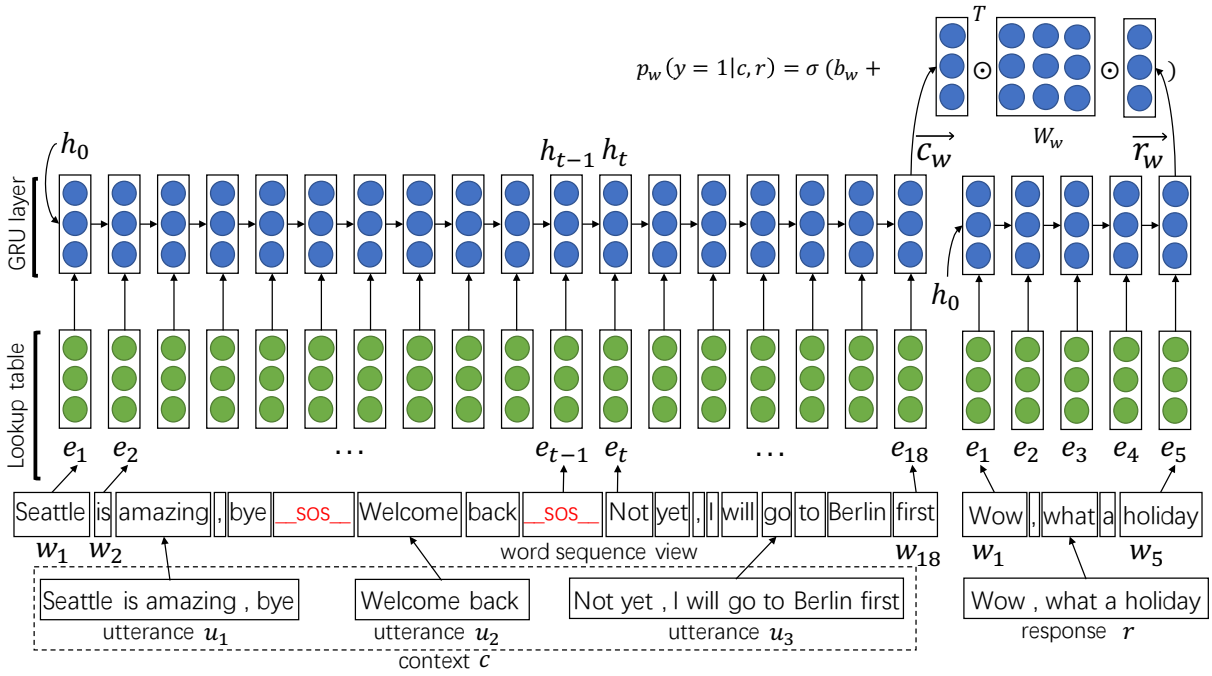


Figure 1: Word sequence model for response selection

3 Response Selection Model

In the task of response selection, a conventional DNN-based architecture represents the context and response as low dimensional embeddings with deep learning models. The response is selected based on the similarity of these two embeddings. We formulate it as

$$p(y = 1|c, r) = \sigma(\overrightarrow{c}^T W \overrightarrow{r} + b) \quad (1)$$

where c and r denote the context and response, $\overrightarrow{c}$ and $\overrightarrow{r}$ are their embeddings constructed with DNNs. $\sigma(x)$ is a sigmoid function defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. $p(y = 1|c, r)$ is the confidence of selecting response r for context c. The matrix W and the scalar b are metric parameters to be learned to measure the similarity between the context and response.

We extend this architecture in a multi-view manner, which jointly models the context and response in two views. In this section, we first briefly describe the word sequence model. Then we introduce the utterance sequence model and multi-view response selection model in details.

3.1 Word Sequence Model

The word sequence model in this paper is similar to the LSTM-based model proposed in Lowe et al. (2015). As shown in Figure 1, three utterances of context c, written as u_1, u_2 and u_3, are connected as a sequence of words. A special word *__sos__* is inserted between every two adjacent utterances, denoting the boundary between utterances. Given the word sequences of context and response, words are mapped into word embeddings through a shared lookup table. A Gated Recurrent Unit neural network (GRU) (Chung et al., 2014) is employed to construct the context embedding and response embedding. It operates recurrently on the two word embedding sequences as Equation 2 to Equation 5, where h_{t-1} is the hidden state of GRU when it reads a word embedding e_{t-1} of word w_{t-1}, h_0 is a zero vector as the initiation state, z_t is an *update gate* and r_t is a *reset gate*. The new hidden state h_t for embedding e_t is a combination of the previous hidden state h_{t-1} and the input embedding e_t, controlled by the update gate z_t and reset gate r_t. U, U_z, U_r, W, W_z and W_r are model parameters of GRU to be learned. $\otimes$ denotes element-wise multiplication.

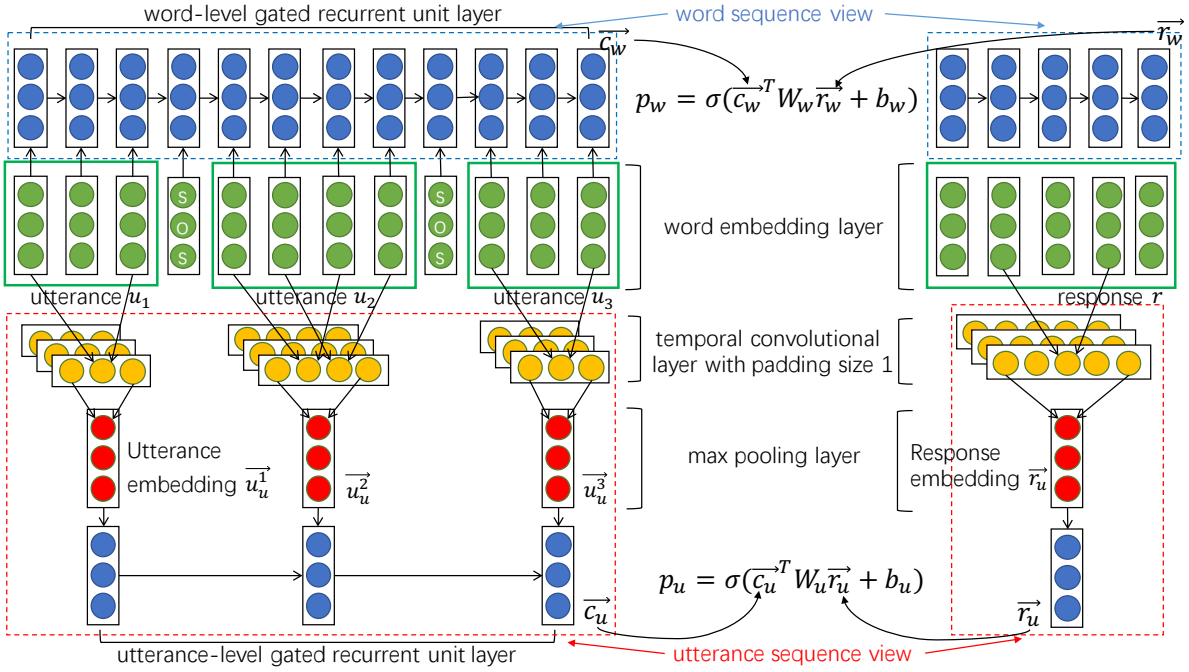


Figure 2: Multi-view response selection model

$$h_t = (\mathbf{1} - z_t) \otimes h_{t-1} + z_t \otimes \hat{h}_t \quad (2)$$

$$z_t = \sigma(W_z e_t + U_z h_{t-1}) \quad (3)$$

$$\hat{h}_t = tanh(W e_t + U(r_t \otimes h_{t-1})) \quad (4)$$

$$r_t = \sigma(W_r e_t + U_r h_{t-1}) \quad (5)$$

After reading the whole word embedding sequence, word-level semantic and dependencies in the whole sequence are encoded in the hidden state of GRU, which represents the meaning of the whole sequence (Karpathy et al., 2015). Therefore we use the last hidden state of GRU as the context embedding and response embedding in word sequence model, named $\overrightarrow{c_w}$ and $\overrightarrow{r_w}$ respectively[1]. The confidence of selecting response in word sequence model is then calculated as in Equation 6:

$$p_w(y = 1|c, r) = \sigma(\overrightarrow{c_w}^T W_w \overrightarrow{r_w} + b_w) \quad (6)$$

where W_w and b_w are metric parameters to be trained in word sequence model. $\overrightarrow{c_w}$ and $\overrightarrow{r_w}$ are constructed by a same GRU in word sequence model.

[1]We use two subscripts, i.e., w and u, to distinguish notation in the two views.

3.2 Utterance Sequence Model

Utterance sequence model regards the context as a hierarchical structure, where the response and each utterance are first represented based on word embeddings, then the context embedding is constructed for the confidence calculation of response selection. As the lower part of Figure 2 illustrates, the construction of the utterance embedding and response embedding is in a convolutional manner, which contains the following layers:

Padding Layer: Given a word embedding sequence belonging to a certain utterance (response), namely $[e_1, ..., e_m]$, the padding layer makes its outer border with $\lfloor n/2 \rfloor$ zero vectors, the padded sequence is $[0_1, .., 0_{\lfloor n/2 \rfloor}, e_1, ..., e_m, 0_1, .., 0_{\lfloor n/2 \rfloor}]$, where n is the size of convolution window used in temporal convolutional layer.

Temporal Convolutional Layer: Temporal convolutional layer reads the padded word embedding sequence through a sliding convolution window with size n. For every step that the sliding window moves, a *region vector* is produced by concatenating the word embeddings within the sliding window, denoted as $[e_i \oplus ... \oplus$

$e_{i+n-1}] \in \mathbb{R}^{n|e|}$, where $\oplus$ denotes the concatenation of embeddings, $|e|$ is the size of word embedding. The temporal convolutional layer consists of k kernels, each of which implies a certain dimension and maps the *region vector* to a value in its dimension by convolution operation. The convolution result of each kernel, termed $conv_i$, is further activated with the *RELU* non-linear activation function (Xu et al., 2015), which is formulated as:

$$f_{relu}(conv_i) = max(conv_i, 0) \quad (7)$$

Pooling Layer: Because utterance and response are naturally variable-sized, we put a *max-over-time pooling layer* on the top of temporal convolutional layer (Kim, 2014), which extracts the max value for each kernel, and gets a fix-sized representation of length k for utterance and response.

In particular, representations constructed by CNN with max-pooling reflect the core meanings of utterance and response. The embeddings of utterance u_i and response r in utterance sequence view are referred to as $\overrightarrow{u_u^i}$ and $\overrightarrow{r_u}$. Utterance embeddings are connected in the sequence and fed into a GRU, which captures utterance-level semantic and discourse information in the whole context and encodes those information as context embedding, written as $\overrightarrow{c_u}$. The confidence of selecting response r for context c in utterance sequence model, named $p_u(y = 1|c, r)$, is calculated using Equation 8:

$$p_u(y = 1|c, r) = \sigma(\overrightarrow{c_u}^T W_u \overrightarrow{r_u} + b_u) \quad (8)$$

It is worth noticing that the TCNN used here is shared in constructing the utterance embedding and response embedding. The word embeddings are also shared for both the context and response. The $\_\_sos\_\_$ tag in word sequence view is not used in the utterance sequence model.

3.3 Multi-view Model

Organic integration of different views has been proven to be very effective in the field of recommendation, representation learning and other research areas (Elkahky et al., 2015; Wang et al., 2015b). Most existing multi-view models integrate different views via a linear/nonlinear combination. Researchers have demonstrated that jointly minimizing two factors, i.e., 1) the *training error* of each view and 2) the *disagreement* between complementary views can significantly improve the performance of the combination of multi-views (Xu et al., 2013).

Our multi-view response selection model is designed as shown in Figure 2. As we can see, the context and response are jointly represented as semantic embeddings in these two views. The underlying word embeddings are shared across the context and response in these two views. The complementary information of these two views is exchanged via the shared word embeddings. The utterance embeddings are modeled through a TCNN in the utterance sequence view. Two independent Gated Recurrent Units are used to model the word embeddings and utterance embeddings separately on word sequence view and utterance sequence view, the former of which captures dependencies in word level and the latter captures utterance-level semantic and discourse information. Confidences for selecting the response in these two views are calculated separately. We optimize the multi-view model by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}} + \mathcal{L}_{\mathcal{L}} + \frac{\lambda}{2}\|\theta\| \quad (9)$$

$$\mathcal{L}_{\mathcal{D}} = \sum_i (p_w(l_i)\bar{p_u}(l_i) + p_u(l_i)\bar{p_w}(l_i)) \quad (10)$$

$$\mathcal{L}_{\mathcal{L}} = \sum_i (1 - p_w(l_i)) + \sum_i (1 - p_u(l_i)) \quad (11)$$

where the object function of the multi-view model $\mathcal{L}$ is comprised of the *disagreement loss* $\mathcal{L}_{\mathcal{D}}$, the *likelihood loss* $\mathcal{L}_{\mathcal{L}}$ and the regular term $\frac{\lambda}{2}\|\theta\|$. $p_w(l_i) = p_w(y = l_i|c, r)$ and $p_u(l_i) = p_u(y = l_i|c, r)$ denote the likelihood of the i-th instance with label l_i from training set in these two views. Only two labels, $\{0, 1\}$, denote the correctness of the response during training. $\bar{p}_w(l_i)$ and $\bar{p}_u(l_i)$ denote the probability $p_w(y \neq l_i)$ and $p_u(y \neq l_i)$ respectively. The multi-view model is trained to jointly minimize the *disagreement loss* and the *likelihood loss*. θ denotes all the parameters of the multi-view model.

The unweighted summation of confidences from these two views is used during prediction, defined as

Model/Metrics	1 in 10 R@1	1 in 10 R@2	1in 10 R@5	1 in 2 R@1
Random-guess	10%	20%	50%	50%
TF-IDF	41.0%	54.5%	70.8%	65.9%
Word-seq-LSTM (Lowe et al., 2015)	60.40%	74.50%	92.60%	87.80%
Word-seq-GRU	60.85%	75.71%	93.13%	88.55%
Utter-seq-GRU	62.19%	76.56%	93.42%	88.83%
Multi-view	**66.15%**	**80.12%**	**95.09%**	**90.80%**

Table 1: Performance comparison between our models and baseline models. In the table, **Word-seq-LSTM** is the experiment result of the LSTM-based word sequence model reported by Lowe et at (2015). **Word-seq GRU** is the word sequence model that we implement with GRU. **Utter-seq-GRU** is the proposed utterance-sequence model. The **Multi-view** is our multi-view response selection model. In addition, we list the performance of **Random-guess** and **TF-IDF**

in Equation 12:

$$s_{mtv}(y = 1|c, r) = p_w(y = 1|c, r) + p_u(y = 1|c, r) \quad (12)$$

The response with larger $s_{mtv}(y = 1|c, r)$ is more likely to be selected. We will investigate other combination models in our future work.

4 Experiment

4.1 Dataset

Our model is evaluated on the public Ubuntu Corpus (Lowe et al., 2015), designed for response selection study of multi-turn human-computer conversation (Serban et al., 2015). The dataset contains 0.93 million human-human dialogues crawled from an Internet chatting room for Ubuntu trouble shooting. Around 1 million context-response-labeled triples, namely $< c, r, l >$, are generated for training after preprocessing[2], where the original context and the corresponding response are taken as the positive instances while the random utterances in the data set taken as the negative instances, and the number of positive instance and negative instance in training set is balanced. The validation set and testing set are constructed in a similar way to the training set, with one notable difference that for each context and the corresponding positive response, 9 negative responses are randomly selected for further evaluation.

4.2 Experiment Setup

Following the work of Lowe et al., (2015), the evaluation metric is 1 in m Recall@k (denoted 1 in m R@k), where a response selection model is designed to select k most likely responses among m candidates, and it gets the score "1" if the correct response is in the k selected ones. This metric can be seen as an adaptation of the precision and recall metrics previously applied to dialogue datasets (Schatzmann et al., 2005). It is worth noticing that 1 in 2 R@1 equals to precision and recall in binary classification.

4.3 Model Training and Hyper-parameters

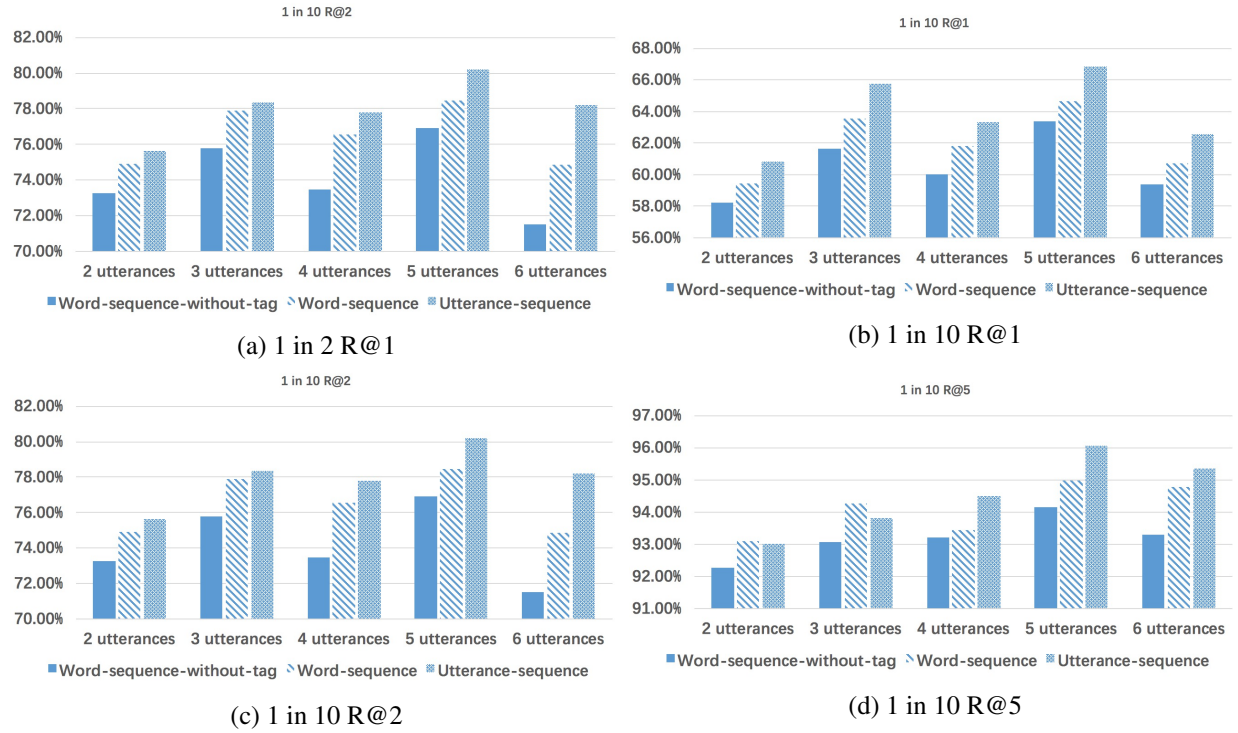
We initialize word embeddings with a pre-trained embedding matrix through GloVe (Pennington et al., 2014) [3]. We use Stochastic Gradient Descent (SGD) for optimizing. Hidden size for a gated recurrent unit is set to 200 in both word sequence model and utterance sequence model. The number of convolutional kernels is set to 200. Our initial learning rate is 0.01 with mini-batch size of 32. Other hyper-parameters are set exactly the same as the baseline. We train our models with a single machine using 12 threads and each model will converge after 4-5 epochs of training data. The best model is selected with a holdout validation dataset.

4.4 Comparison Approaches

We consider the word sequence model implemented by Lowe et at., (2015) with LSTM as our baseline, the best model in context-sensitive response selection so far. Moreover, we also implement the word sequence model and the utterance sequence model with GRU for further analysis. Two simple approaches are also implemented, i.e., the Random-

[2]Preprocessing includes tokenization, recognition of named entity, urls and numbers.

[3]Initialization of word embedding can be obtained on https://github.com/npow/ubottu



(a) 1 in 2 R@1

(b) 1 in 10 R@1

(c) 1 in 10 R@2

(d) 1 in 10 R@5

Figure 3: Performance comparison between word sequence model (with/without $\_\_sos\_\_$ tags) and utterance sequence model. We choose the number of utterances in range of [2,6], since most samples in testset fall in this interval

guess and the TF-IDF, as the bottom line for performance comparison. The performance of Random-guess is calculated by mathematics with an assumption that each response in candidates has the equal probability to be selected. The TF-IDF is implemented in the same way in Lowe et al., (2015). TF for a word is calculated as the count of times it appears in a certain context or response. IDF for each word w is $log(\frac{N}{|d \in D : w \in d|})$, where D denotes the whole training set, N is the size of D, d is a conversation in D. The context and the response in testset are represented as a bag-of-words according to TF-IDF. The selection confidence is estimated as the cosine score between context and response.

4.5 Experimental Result

We summarize the experiment result in Table 1. As shown in Table 1, all DNN-based models achieve significant improvements compared to Random-guess and TF-IDF, which implies the effectiveness of DNN models in the task of response selection. The word sequence models implemented with GRU and LSTM achieve similar performance. The utterance sequence model significantly outperforms word sequence models for 1 in 10 R@1. Multi-view model significantly outperforms all the other models, especially for 1 in 10 R@1, which is more difficult and closer to the real world scenario than other metrics. The experimental result demonstrates the effectiveness of multi-view model and proves that word sequence view and utterance sequence view can bring complementary information for each other.

5 Analysis

We examine the complementarity between word sequence model and utterance sequence model in two folds, i.e., via statistic analysis and case study.

5.1 Statistical Analysis

We compare the performance of word sequence model[4] and utterance sequence model for different number of utterances in the contexts. In addition, we also examine what the contribution $\_\_sos\_\_$ tag makes in word sequence view. The performance

[4]The GRU-based word sequence model that we implemented is used for comparison.

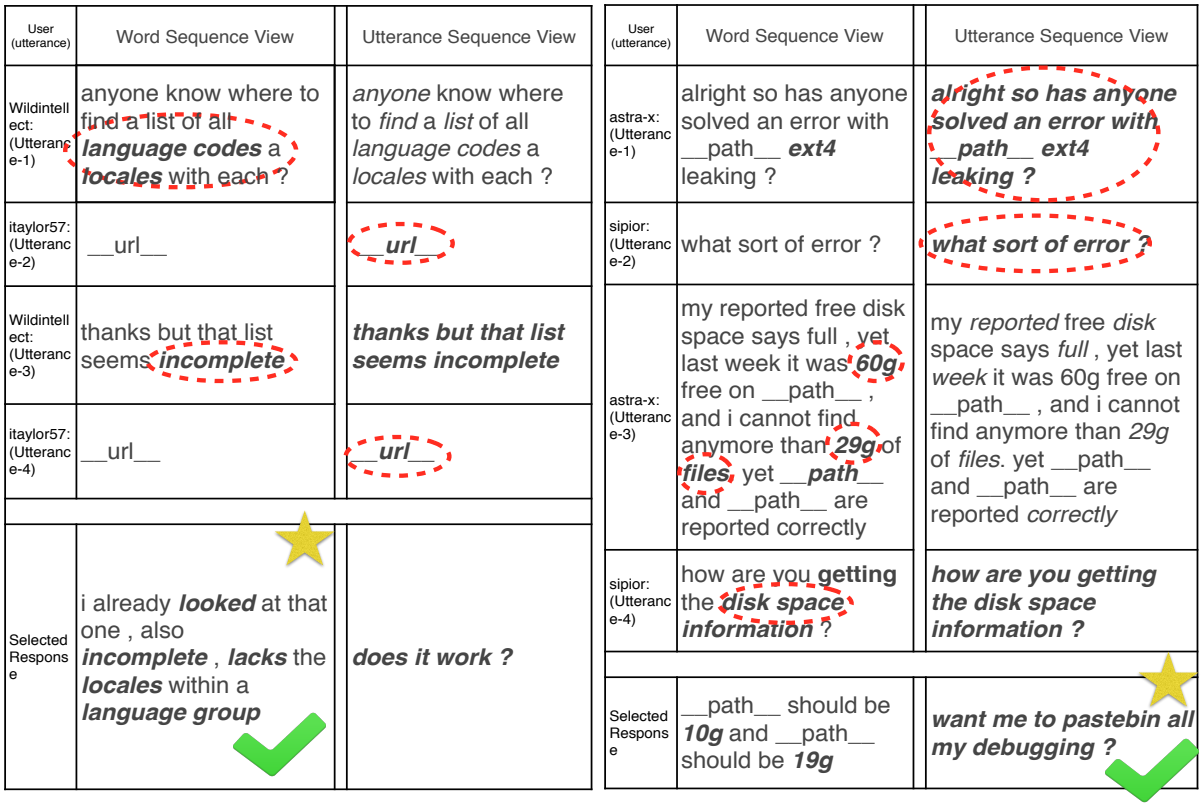


Figure 4: Case studies for analysis of word sequence model and utterance sequence model. The context and the selected responses are collected from testset. Response with a green checkmark means it is a correct one, otherwise it is incorrect. Words (Utterances) in **bold** are the important elements recognized by our importance analysis approach. The yellow start denotes the selection of multi-view model.

is shown in Figure 3. We can see that as the number of turns increases, the utterance sequence model outperforms word sequence model more significantly, which implies that utterance sequence model can provide complementary information to word sequence model for a long context. Furthermore, word sequence model without $\_\_sos\_\_$ tag has an obvious fall in performance compared with word sequence model with $\_\_sos\_\_$, which implies its crucial role in distinguishing utterances for modeling context.

5.2 Case Study

We analyze samples from testset to examine the complementarity between these two views. The key words for word sequence model and core utterances for utterance sequence model are extracted for analysis. These important elements are recognized based on the work of Li et al. (2015), where the gradients of their embeddings are used for importance analysis. After studying the testset, we find that the word sequence model selects responses according to the **matching of key words** while the utterance sequence model selects responses based on the **matching of core utterances**. We list two cases in Figure 4 as examples.

As it shows, the word sequence model prefers to select the response that shares similar key words to the context, such as the words *"incomplete"* and *"locales"* in example 1 or *"60g"* and *"19g"* in example 2. Although key word matching is a useful feature in selecting response for cases such as example 1, it fails in cases like example 2, where incorrect response happens to share similar words with the context. Utterance sequence model, on the other side, leverages core utterances for selecting response. As shown in example 2, utterance-1 and utterance-2 are recognized as the core utterances, the main topic of the two utterance is *"solved"* and *"error"*, which is close to the topic of the correct re-

sponse. However, for cases like example 1, where the core meaning of correct response is jointly combined with different words in different utterances, the utterance sequence model does not perform well.

The multi-view model can successfully select the correct responses in both two examples, which implies its ability to jointly leverage information from these two views.

6 Conclusion

In this paper, we propose a multi-view response selection model for multi-turn human-computer conversation. We integrate the existing word sequence view and a new view, i.e., utterance sequence view, into a unified multi-view model. In the view of utterance sequence, discourse information can be learnt through utterance-level recurrent neural network, different from word sequence view. The representations learnt from the two views provide complementary information for each other in the task of response selection. Experiments show that our multi-view model significantly outperforms the state-of-the-art word sequence view models. We will extend our framework to response generation approaches in our future work. We believe it will help construct a better representation of context in the encoding phrase of DNN-based generation model and thus improve the performance.

Acknowledgement

This paper is supported by National Basic Research Program of China (973 program No. 2014CB340505). We gratefully thank the anonymous reviewers for their insightful comments.

References

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. International World Wide Web Conferences Steering Committee.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375.

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

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *In Proc. EMNLP*, pages 583–593. Association for Computational Linguistics.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th SIGdial Workshop on DISCOURSE and DIALOGUE*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A Walker, Rebecca Passonneau, and Julie E Boland. 2001. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 515–522. Association for Computational Linguistics.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015a. Syntax-based deep matching of short texts. *arXiv preprint arXiv:1503.02427*.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. 2015b. On deep multi-view representation learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1083–1092.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics, September.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. Docchat: An information retrieval approach for chatbot engines using unstructured documents.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Variational Neural Discourse Relation Recognizer

Biao Zhang[1], **Deyi Xiong**[2]*, **Jinsong Su**[1], **Qun Liu**[3,4], **Rongrong Ji**[1], **Hong Duan**[1], **Min Zhang**[2]
Xiamen University, Xiamen, China 361005[1]
Provincial Key Laboratory for Computer Information Processing Technology
Soochow University, Suzhou, China 215006[2]
ADAPT Centre, School of Computing, Dublin City University[3]
Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences[4]
zb@stu.xmu.edu.cn, {jssu, rrji, hduan}@xmu.edu.cn
qun.liu@dcu.ie, {dyxiong, minzhang}@suda.edu.cn

Abstract

Implicit discourse relation recognition is a crucial component for automatic discourse-level analysis and nature language understanding. Previous studies exploit discriminative models that are built on either powerful manual features or deep discourse representations. In this paper, instead, we explore generative models and propose a variational neural discourse relation recognizer. We refer to this model as *VarNDRR*. VarNDRR establishes a directed probabilistic model with a latent continuous variable that generates both a discourse and the relation between the two arguments of the discourse. In order to perform efficient inference and learning, we introduce neural discourse relation models to approximate the prior and posterior distributions of the latent variable, and employ these approximated distributions to optimize a reparameterized variational lower bound. This allows VarNDRR to be trained with standard stochastic gradient methods. Experiments on the benchmark data set show that VarNDRR can achieve comparable results against state-of-the-art baselines without using any manual features.

1 Introduction

Discourse relation characterizes the internal structure and logical relation of a coherent text. Automatically identifying these relations not only plays an important role in discourse comprehension and generation, but also obtains wide applications in many other relevant natural language processing tasks, such as text summarization (Yoshida et al., 2014), conversation (Higashinaka et al., 2014), question answering (Verberne et al., 2007) and information extraction (Cimiano et al., 2005). Generally, discourse relations can be divided into two categories: explicit and implicit, which can be illustrated in the following example:

> *The company was disappointed by the ruling.* <u>*because*</u> *The obligation is totally unwarranted.* (adapted from wsj_0294)

With the discourse connective *because*, these two sentences display an explicit discourse relation CONTINGENCY which can be inferred easily. Once this discourse connective is removed, however, the discourse relation becomes implicit and difficult to be recognized. This is because almost no surface information in these two sentences can signal this relation. For successful recognition of this relation, in the contrary, we need to understand the deep semantic correlation between *disappointed* and *obligation* in the two sentences above. Although explicit discourse relation recognition (DRR) has made great progress (Miltsakaki et al., 2005; Pitler et al., 2008), implicit DRR still remains a serious challenge due to the difficulty in semantic analysis.

Conventional approaches to implicit DRR often treat the relation recognition as a classification problem, where discourse arguments and relations are regarded as the inputs and outputs respectively. Generally, these methods first generate a representation for a discourse, denoted as $\mathbf{x}$[1] (e.g., manual fea-

*Corresponding author

[1]Unless otherwise specified, all variables in the paper, e.g., $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are multivariate. But for notational convenience, we

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 382–391,
Austin, Texas, November 1-5, 2016.

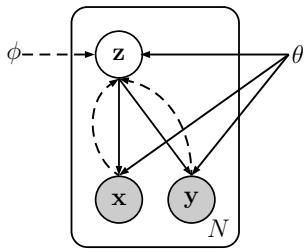


Figure 1: Graphical illustration for VarNDRR. Solid lines denote the generative model $p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{y}|\mathbf{z})$, dashed lines denote the variational approximation $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ to the posterior $p(\mathbf{z}|\mathbf{x},\mathbf{y})$ and $q'_\phi(\mathbf{z}|\mathbf{x})$ to the prior $p(\mathbf{z})$ for inference. The variational parameters ϕ are learned jointly with the generative model parameters θ.

tures in SVM-based recognition (Pitler et al., 2009; Lin et al., 2009) or sentence embeddings in neural networks-based recognition (Ji and Eisenstein, 2015; Zhang et al., 2015)), and then directly model the conditional probability of the corresponding discourse relation $\mathbf{y}$ given $\mathbf{x}$, i.e. $p(\mathbf{y}|\mathbf{x})$. In spite of their success, these discriminative approaches rely heavily on the goodness of discourse representation $\mathbf{x}$. Sophisticated and good representations of a discourse, however, may make models suffer from overfitting as we have no large-scale balanced data.

Instead, we assume that there is a latent continuous variable $\mathbf{z}$ from an underlying semantic space. It is this latent variable that generates both discourse arguments and the corresponding relation, i.e. $p(\mathbf{x},\mathbf{y}|\mathbf{z})$. The latent variable enables us to jointly model discourse arguments and their relations, rather than conditionally model $\mathbf{y}$ on $\mathbf{x}$. However, the incorporation of the latent variable makes the modeling difficult due to the intractable computation with respect to the posterior distribution.

Inspired by Kingma and Welling (2014) as well as Rezende et al. (2014) who introduce a variational neural inference model to the intractable posterior via optimizing a reparameterized variational lower bound, we propose a variational neural discourse relation recognizer (VarNDRR) with a latent continuous variable for implicit DRR in this paper. The key idea behind VarNDRR is that although the posterior distribution is intractable, we can approximate it via a deep neural network. Figure 1 illustrates the graph structure of VarNDRR. Specifically, there are two essential components:

- *neural discourse recognizer* As a discourse $\mathbf{x}$ and its corresponding relation $\mathbf{y}$ are independent with each other given the latent variable $\mathbf{z}$ (as shown by the solid lines), we can formulate the generation of $\mathbf{x}$ and $\mathbf{y}$ from $\mathbf{z}$ in the equation $p_\theta(\mathbf{x},\mathbf{y}|\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{y}|\mathbf{z})$. These two conditional probabilities on the right hand side are modeled via deep neural networks (see section 3.1).
- *neural latent approximator* VarNDRR assumes that the latent variable can be inferred from discourse arguments $\mathbf{x}$ and relations $\mathbf{y}$ (as shown by the dash lines). In order to infer the latent variable, we employ a deep neural network to approximate the posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ as well as the prior $q'_\phi(\mathbf{z}|\mathbf{x})$ (see section 3.2), which makes the inference procedure efficient. We further employ a reparameterization technique to sample z from $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ that not only bridges the gap between the recognizer and the approximator but also allows us to use the standard stochastic gradient ascent techniques for optimization (see section 3.3).

The main contributions of our work lie in two aspects. 1) We exploit a generative graphic model for implicit DRR. To the best of our knowledge, this has never been investigated before. 2) We develop a neural recognizer and two neural approximators specifically for implicit DRR, which enables both the recognition and inference to be efficient.

We conduct a series of experiments for English implicit DRR on the PDTB-style corpus to evaluate the effectiveness of our proposed VarNDRR model. Experiment results show that our variational model achieves comparable results against several strong baselines in term of F1 score. Extensive analysis on the variational lower bound further reveals that our model can indeed fit the data set with respect to discourse arguments and relations.

2 Background: Variational Autoencoder

The variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014), which forms the basis of our model, is a generative model that can be regarded as a regularized version of the standard

treat them as univariate variables in most cases. Additionally, we use bold symbols to denote variables, and plain symbols to denote values.

autoencoder. With a latent random variable $\mathbf{z}$, VAE significantly changes the autoencoder architecture to be able to capture the variations in the observed variable $\mathbf{x}$. The joint distribution of $(\mathbf{x}, \mathbf{z})$ is formulated as follows:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \tag{1}$$

where $p_\theta(\mathbf{z})$ is the prior over the latent variable, usually equipped with a simple Gaussian distribution. $p_\theta(\mathbf{x}|\mathbf{z})$ is the conditional distribution that models the probability of $\mathbf{x}$ given the latent variable $\mathbf{z}$. Typically, VAE parameterizes $p_\theta(\mathbf{x}|\mathbf{z})$ with a highly nonlinear but flexible function approximator such as a neural network.

The objective of VAE is to maximize a variational lower bound as follows:

$$\begin{aligned}\mathcal{L}_{VAE}(\theta, \phi; \mathbf{x}) = &-\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \\ &+\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x})\end{aligned} \tag{2}$$

where $\mathrm{KL}(Q||P)$ is Kullback-Leibler divergence between two distributions Q and P. $q_\phi(\mathbf{z}|\mathbf{x})$ is an approximation of the posterior $p(\mathbf{z}|\mathbf{x})$ and usually follows a diagonal Gaussian $\mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$ whose mean μ and variance σ^2 are parameterized by again, neural networks, conditioned on $\mathbf{x}$.

To optimize Eq. (2) stochastically with respect to both θ and ϕ, VAE introduces a reparameterization trick that parameterizes the latent variable $\mathbf{z}$ with the Gaussian parameters μ and σ in $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\tilde{z} = \mu + \sigma \odot \epsilon \tag{3}$$

where ϵ is a standard Gaussian variable, and $\odot$ denotes an element-wise product. Intuitively, VAE learns the representation of the latent variable not as single points, but as soft ellipsoidal regions in latent space, forcing the representation to fill the space rather than memorizing the training data as isolated representations. With this trick, the VAE model can be trained through standard backpropagation technique with stochastic gradient ascent.

3 The VarNDRR Model

This section introduces our proposed VarNDRR model. Formally, in VarNDRR, there are two observed variables, $\mathbf{x}$ for a discourse and $\mathbf{y}$ for the corresponding relation, and one latent variable $\mathbf{z}$. As illustrated in Figure 1, the joint distribution of the three variables is formulated as follows:

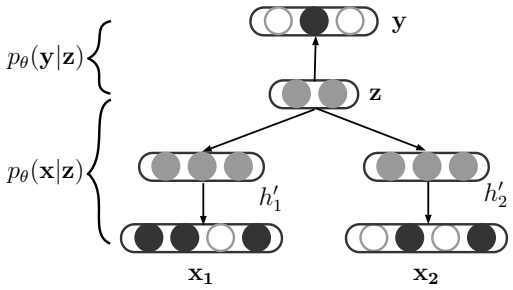


Figure 2: Neural networks for conditional probabilities $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{y}|\mathbf{z})$. The gray color denotes real-valued representations while the white and black color 0-1 representations.

$$p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})p(\mathbf{z}) \tag{4}$$

We begin with this distribution to elaborate the major components of VarNDRR.

3.1 Neural Discourse Recognizer

The conditional distribution $p(\mathbf{x}, \mathbf{y}|\mathbf{z})$ in Eq. (4) shows that both discourse arguments and the corresponding relation are generated from the latent variable. As shown in Figure 1, $\mathbf{x}$ is d-separated from $\mathbf{y}$ by $\mathbf{z}$. Therefore the discourse $\mathbf{x}$ and the corresponding relation $\mathbf{y}$ is independent given the latent variable $\mathbf{z}$. The joint probability can be therefore formulated as follows:

$$p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{y}|\mathbf{z})p(\mathbf{z}) \tag{5}$$

We use a neural model $q'_\phi(\mathbf{z}|\mathbf{x})$ to approximate the prior $p(\mathbf{z})$ conditioned on the discourse $\mathbf{x}$ (see the following section). With respect to the other two conditional distributions, we parameterize them via neural networks as shown in Figure 2.

Before we describe these neural networks, it is necessary to briefly introduce how discourse relations are annotated in our training data. The PDTB corpus, used as our training data, annotates implicit discourse relations between two neighboring arguments, namely *Arg1* and *Arg2*. In VarNDRR, we represent the two arguments with bag-of-word representations, and denote them as $\mathbf{x_1}$ and $\mathbf{x_2}$.

To model $p_\theta(\mathbf{x}|\mathbf{z})$ (the bottom part in Figure 2), we project the representation of the latent variable

$z \in \mathbb{R}^{d_z}$ onto a hidden layer:

$$\begin{aligned} h'_1 &= f(W_{h'_1} z + b_{h'_1}) \\ h'_2 &= f(W_{h'_2} z + b_{h'_1}) \end{aligned} \tag{6}$$

where $h'_1 \in \mathbb{R}^{d_{h'_1}}, h'_2 \in \mathbb{R}^{d_{h'_2}}$, W_* is the transformation matrix, b_* is the bias term, d_u denotes the dimensionality of vector representations of u and $f(\cdot)$ is an element-wise activation function, such as $tanh(\cdot)$, which is used throughout our model.

Upon this hidden layer, we further stack a Sigmoid layer to predict the probabilities of corresponding discourse arguments:

$$\begin{aligned} x'_1 &= \text{Sigmoid}(W_{x'_1} h'_1 + b_{x'_1}) \\ x'_2 &= \text{Sigmoid}(W_{x'_2} h'_2 + b_{x'_2}) \end{aligned} \tag{7}$$

here, $x'_1 \in \mathbb{R}^{d_{x_1}}$ and $x'_2 \in \mathbb{R}^{d_{x_2}}$ are the real-valued representations of the reconstructed x_1 and x_2 respectively.[2] We assume that $p_\theta(\mathbf{x}|\mathbf{z})$ is a multivariate Bernoulli distribution because of the bag-of-word representation. Therefore the logarithm of $p(x|z)$ is calculated as the sum of probabilities of words in discourse arguments as follows:

$$\begin{aligned} &\log\ p(x|z) \\ &= \sum_i x_{1,i} \log x'_{1,i} + (1 - x_{1,i}) \log(1 - x'_{1,i}) \\ &+ \sum_j x_{2,j} \log x'_{2,j} + (1 - x_{2,j}) \log(1 - x'_{2,j}) \end{aligned} \tag{8}$$

where $u_{i,j}$ is the jth element in u_i.

In order to estimate $p_\theta(\mathbf{y}|\mathbf{z})$ (the top part in Figure 2), we stack a softmax layer over the multilayer-perceptron-transformed representation of the latent variable z:

$$y' = \text{SoftMax}(W_{y'}\text{MLP}(z) + b_{y'}) \tag{9}$$

$y' \in \mathbb{R}^{d_y}$, and d_y denotes the number of discourse relations. MLP projects the representation of latent variable $\mathbf{z}$ into a d_m-dimensional space through four internal layers, each of which has dimension d_m. Suppose that the true relation is $y \in \mathbb{R}^{d_y}$, the logarithm of $p(y|z)$ is defined as:

$$\log\ p(y|z) = \sum_{i=1}^{d_y} y_i \log y'_i \tag{10}$$

[2]Notice that the equality of $d_{x_1} = d_{x_2}, d_{h'_1} = d_{h'_2}$ is not necessary though we assume so in our experiments.

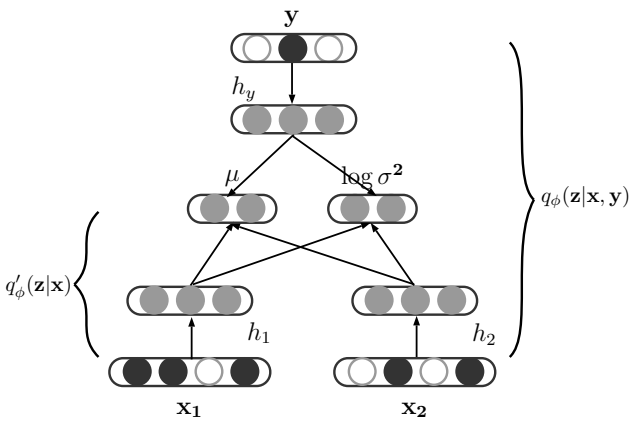


Figure 3: Neural networks for Gaussian parameters μ and $\log \sigma$ in the approximated posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ and prior $q'_\phi(\mathbf{z}|\mathbf{x})$.

In order to precisely estimate these conditional probabilities, our model will force the representation z of the latent variable to encode semantic information for both the reconstructed discourse x' (Eq. (8)) and predicted discourse relation y' (Eq. (10)), which is exactly what we want.

3.2 Neural Latent Approximator

For the joint distribution in Eq. (5), we can define a variational lower bound that is similar to Eq. (2). The difference lies in two aspects: the approximate prior $q'_\phi(\mathbf{z}|\mathbf{x})$ and posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$. We model both distributions as a multivariate Gaussian distribution with a diagonal covariance structure:

$$\mathcal{N}(\mathbf{z}; \mu, \sigma^2\mathbf{I})$$

The mean μ and s.d. σ of the approximate distribution are the outputs of the neural network as shown in Figure 3, where the prior and posterior have different conditions and independent parameters.

Approximate Posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ is modeled to condition on both observed variables: the discourse arguments $\mathbf{x}$ and relations $\mathbf{y}$. Similar to the calculation of $p_\theta(\mathbf{x}|\mathbf{z})$, we first transform the input $\mathbf{x}$ and $\mathbf{y}$ into a hidden representation:

$$\begin{aligned} h_1 &= f(W_{h_1} x_1 + b_{h_1}) \\ h_2 &= f(W_{h_2} x_2 + b_{h_2}) \\ h_y &= f(W_{h_y} y + b_{h_y}) \end{aligned} \tag{11}$$

where $h_1 \in \mathbb{R}^{d_{h_1}}, h_2 \in \mathbb{R}^{d_{h_2}}$ and $h_y \in \mathbb{R}^{d_{h_y}}$.[3]

[3]Notice that d_{h_1}/d_{h_2} are not necessarily equal to $d_{h'_1}/d_{h'_2}$.

We then obtain the Gaussian parameters of the posterior μ and $\log \sigma^2$ through linear regression:

$$\begin{aligned} \mu &= W_{\mu_1} h_1 + W_{\mu_2} h_2 + W_{\mu_y} h_y + b_\mu \\ \log \sigma^2 &= W_{\sigma_1} h_1 + W_{\sigma_2} h_2 + W_{\sigma_y} h_y + b_\sigma \end{aligned} \quad (12)$$

where $\mu, \sigma \in \mathbb{R}^{d_z}$. In this way, this posterior approximator can be efficiently computed.

Approximate Prior $q'_\phi(\mathbf{z}|\mathbf{x})$ is modeled to condition on discourse arguments $\mathbf{x}$ alone. This is based on the observation that discriminative models are able to obtain promising results using only $\mathbf{x}$. Therefore, assuming the discourse arguments encode the prior information for discourse relation recognition is meaningful.

The neural model for prior $q'_\phi(\mathbf{z}|\mathbf{x})$ is the same as that (i.e. Eq (11) and (12)) for posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ (see Figure 3), except for the absence of discourse relation $\mathbf{y}$. For clarity , we use μ' and σ' to denote the mean and s.d. of the approximate prior.

With the parameters of Gaussian distribution, we can access the representation z using different sampling strategies. However, traditional sampling approaches often breaks off the connection between recognizer and approximator, making the optimization difficult. Instead, we employ the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014) as in Eq. (3). During training, we sample the latent variable using $\tilde{z} = \mu + \sigma \odot \epsilon$; during testing, however, we employ the expectation of $\mathbf{z}$ in the approximate prior distribution, i.e. set $\tilde{z} = \mu'$ to avoid uncertainty.

3.3 Parameter Learning

We employ the Monte Carlo method to estimate the expectation over the approximate posterior, that is $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})]$. Given a training instance $(x^{(t)}, y^{(t)})$, the joint training objective is defined:

$$\begin{aligned} \mathcal{L}(\theta, \phi) \simeq & -\mathrm{KL}(q_\phi(\mathbf{z}|x^{(t)}, y^{(t)})||q'_\phi(\mathbf{z}|x^{(t)})) \\ & + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x^{(t)}, y^{(t)}|\tilde{z}^{(t,l)}) \end{aligned} \quad (13)$$

where $\tilde{z}^{(t,l)} = \mu^{(t)} + \sigma^{(t)} \odot \epsilon^{(l)}$ and $\epsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I})$

L is the number of samples. The first term is the KL divergence of two Gaussian distributions which can be computed and differentiated without estimation. Maximizing this objective will minimize the difference between the approximate posterior and prior, thus making the setting $\tilde{z} = \mu'$ during testing reasonable. The second term is the approximate expectation of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})]$, which is also differentiable.

As the objective function in Eq. (13) is differentiable, we can optimize both the model parameters θ and variational parameters ϕ jointly using standard gradient ascent techniques. The training procedure for VarNDRR is summarized in Algorithm 1.

Algorithm 1 Parameter Learning Algorithm of VarNDRR.

Inputs: A, the maximum number of iterations;
M, the number of instances in one batch;
L, the number of samples;
$\theta, \phi \leftarrow$ Initialize parameters
repeat
  $\mathcal{D} \leftarrow$ getRandomMiniBatch(M)
  $\epsilon \leftarrow$ getRandomNoiseFromStandardGaussian()
  $g \leftarrow \nabla_{\theta,\phi} \mathcal{L}(\theta, \phi; \mathcal{D}, \epsilon)$
  $\theta, \phi \leftarrow$ parameterUpdater($\theta, \phi; g$)
until convergence of parameters (θ, ϕ) or reach the maximum iteration A

Relation	#Instance Number		
	Train	Dev	Test
COM	1942	197	152
CON	3342	295	279
EXP	7004	671	574
TEM	760	64	85

Table 1: Statistics of implicit discourse relations for the training (Train), development (Dev) and test (Test) sets in PDTB.

4 Experiments

We conducted experiments on English implicit DRR task to validate the effectiveness of VarNDRR.[4]

4.1 Dataset

We used the largest hand-annotated discourse corpus *PDTB 2.0*[5] (Prasad et al., 2008) (PDTB hereafter). This corpus contains discourse annotations over 2,312 Wall Street Journal articles, and is organized in different sections. Following previous work (Pitler et al., 2009; Zhou et al., 2010; Lan et

[4]Source code is available at https://github.com/DeepLearnXMU/VarNDRR.

[5]http://www.seas.upenn.edu/ pdtb/

Model	Acc	P	R	F1
R & X (2015)	-	-	-	41.00
J & E (2015)	70.27	-	-	35.93
SVM	63.10	22.79	64.47	33.68
SCNN	60.42	22.00	67.76	33.22
VarNDRR	63.30	24.00	71.05	35.88

(a) COM vs Other

Model	Acc	P	R	F1
(R & X (2015))	-	-	-	53.80
(J & E (2015))	76.95	-	-	52.78
SVM	62.62	39.14	72.40	50.82
SCNN	63.00	39.80	75.29	52.04
VarNDRR	53.82	35.39	88.53	50.56

(b) CON vs Other

Model	Acc	P	R	F1
(R & X (2015))	-	-	-	69.40
(J & E (2015))	69.80	-	-	80.02
SVM	60.71	65.89	58.89	62.19
SCNN	63.00	56.29	91.11	69.59
VarNDRR	57.36	56.46	97.39	71.48

(c) EXP vs Other

Model	Acc	P	R	F1
(R & X (2015))	-	-	-	33.30
(J & E (2015))	87.11	-	-	27.63
SVM	66.25	15.10	68.24	24.73
SCNN	76.95	20.22	62.35	30.54
VarNDRR	62.14	17.40	97.65	29.54

(d) TEM vs Other

Table 2: Classification results of different models on the implicit DRR task. **Acc**=Accuracy, **P**=Precision, **R**=Recall, and **F1**=F1 score.

al., 2013; Zhang et al., 2015), we used sections 2-20 as our training set, sections 21-22 as the test set. Sections 0-1 were used as the development set for hyperparameter optimization.

In PDTB, discourse relations are annotated in a predicate-argument view. Each discourse connective is treated as a predicate that takes two text spans as its arguments. The discourse relation tags in PDTB are arranged in a three-level hierarchy, where the top level consists of four major semantic *classes*: TEMPORAL (TEM), CONTINGENCY (CON), EXPANSION (EXP) and COMPARISON (COM). Because the top-level relations are general enough to be annotated with a high inter-annotator agreement and are common to most theories of discourse, in our experiments we only use this level of annotations.

We formulated the task as four separate one-against-all binary classification problems: each top level class vs. the other three discourse relation classes. We also balanced the training set by resampling training instances in each class until the number of positive and negative instances are equal. In contrast, all instances in the test and development set are kept in nature. The statistics of various data sets is listed in Table 1.

4.2 Setup

We tokenized all datasets using *Stanford NLP Toolkit*[6]. For optimization, we employed the Adam algorithm (Kingma and Ba, 2014) to update parameters. With respect to the hyperparameters M, L, A and the dimensionality of all vector representations, we set them according to previous work (Kingma and Welling, 2014; Rezende et al., 2014) and preliminary experiments on the development set. Finally, we set $M = 16, A = 1000, L = 1, d_z = 20, d_{x_1} = d_{x_2} = 10001, d_{h_1} = d_{h_2} = d_{h'_1} = d_{h'_2} = d_m = d_{h_y} = 400, d_y = 2$ for all experiments.[7]. All parameters of VarNDRR are initialized by a Gaussian distribution ($\mu = 0, \sigma = 0.01$). For Adam, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$ with a learning rate 0.001. Additionally, we tied the following parameters in practice: W_{h_1} and W_{h_2}, $W_{x'_1}$ and $W_{x'_2}$.

We compared VarNDRR against the following two different baseline methods:

- **SVM:** a support vector machine (SVM) classifier[8] trained with several manual features.
- **SCNN:** a shallow convolutional neural network proposed by Zhang et al. (2015).

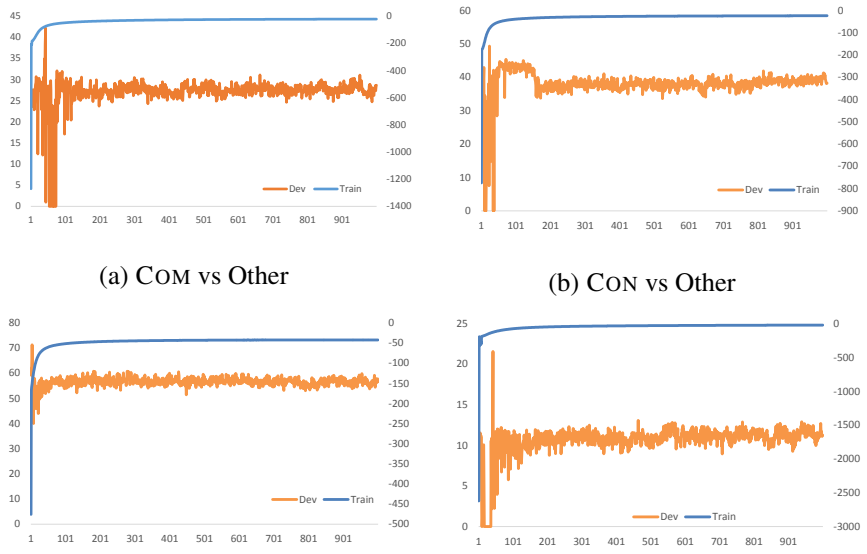
We also provide results from two state-of-the-art systems:

- **Rutherford and Xue (2015)** convert explicit discourse relations into implicit instances.
- **Ji and Eisenstein (2015)** augment discourse representations via entity connections.

[6]http://nlp.stanford.edu/software/corenlp.shtml

[7]There is one dimension in d_{x_1} and d_{x_2} for unknown words.

[8]http://svmlight.joachims.org/



(a) COM vs Other

(b) CON vs Other

(c) EXP vs Other

(d) TEM vs Other

Figure 4: Illustration of the variational lower bound (blue color) on the training set and F-score (brown color) on the development set. Horizontal axis: the epoch numbers; Vertical axis: the F1 score for relation classification (left) and the estimated average variational lower bound per datapoint (right).

Features used in **SVM** are taken from the state-of-the-art implicit discourse relation recognition model, including *Bag of Words*, *Cross-Argument Word Pairs*, *Polarity*, *First-Last, First3*, *Production Rules*, *Dependency Rules* and *Brown cluster pair* (Rutherford and Xue, 2014). In order to collect bag of words, production rules, dependency rules, and cross-argument word pairs, we used a frequency cut-off of 5 to remove rare features, following Lin et al. (2009).

4.3 Classification Results

Because the development and test sets are imbalanced in terms of the ratio of positive and negative instances, we chose the widely-used F1 score as our major evaluation metric. In addition, we also provide the precision, recall and accuracy for further analysis. Table 2 summarizes the classification results.

From Table 2, we observe that the proposed VarNDRR outperforms **SVM** on COM/EXP/TEM and **SCNN** on EXP/COM according to their F1 scores. Although it fails on CON, VarNDRR achieves the best result on EXP/COM among these three models. Overall, VarNDRR is competitive in comparison with these two baselines. With respect to the accuracy, our model does not yield substantial improvements over the two baselines. This may be because that we used the F1 score rather than the accuracy, as our selection criterion on the development set. With respect to the precision and recall, our model tends to produce relatively lower precisions but higher recalls. This suggests that the improvements of VarNDRR in terms of F1 scores mostly benefits from the recall values.

Comparing with the state-of-the-art results of previous work (Ji and Eisenstein, 2015; Rutherford and Xue, 2015), VarNDRR achieves comparable results in term of the F1 scores. Specifically, VarNDRR outperforms Rutherford and Xue (2015) on EXP, and Ji and Eisenstein (2015) on TEM. However, the accuracy of our model fails to surpass these models. We argue that this is because both baselines use many manual features designed with prior human knowledge, but our model is purely neural-based.

Additionally, we find that the performance of our model is proportional to the number of training instances. This suggests that collecting more training instances (in spite of the noises) may be beneficial to our model.

4.4 Variational Lower Bound Analysis

In addition to the classification performance, the efficiency in learning and inference is another concern

for variational methods. Figure 4 shows the training procedure for four tasks in terms of the variational lower bound on the training set. We also provide F1 scores on the development set to investigate the relations between the variational lower bound and recognition performance.

We find that our model converges toward the variational lower bound considerably fast in all experiments (within 100 epochs), which resonates with the previous findings (Kingma and Welling, 2014; Rezende et al., 2014). However, the change trend of the F1 score does not follow that of the lower bound which takes more time to converge. Particularly to the four discourse relations, we further observe that the change paths of the F1 score are completely different. This may suggest that the four discourse relations have different properties and distributions.

In particular, the number of epochs when the best F1 score reaches is also different for the four discourse relations. This indicates that dividing the implicit DRR into four different tasks according to the type of discourse relations is reasonable and better than performing DRR on the mixtures of the four relations.

5 Related Work

There are two lines of research related to our work: *implicit discourse relation recognition* and *variational neural model*, which we describe in succession.

Implicit Discourse Relation Recognition Due to the release of Penn Discourse Treebank (Prasad et al., 2008) corpus, constantly increasing efforts are made for implicit DRR. Upon this corpus, Pilter et al. (2009) exploit several linguistically informed features, such as polarity tags, modality and lexical features. Lin et al. (2009) further incorporate context words, word pairs as well as discourse parse information into their classifier. Following this direction, several more powerful features have been exploited: entities (Louis et al., 2010), word embeddings (Braud and Denis, 2015), Brown cluster pairs and co-reference patterns (Rutherford and Xue, 2014). With these features, Park and Cardie (2012) perform feature set optimization for better feature combination.

Different from feature engineering, predicting discourse connectives can indirectly help the relation classification (Zhou et al., 2010; Patterson and Kehler, 2013). In addition, selecting explicit discourse instances that are similar to the implicit ones can enrich the training corpus for implicit DRR and gains improvement (Wang et al., 2012; Lan et al., 2013; Braud and Denis, 2014; Fisher and Simmons, 2015; Rutherford and Xue, 2015). Very recently, neural network models have been also used for implicit DRR due to its capability for representation learning (Ji and Eisenstein, 2015; Zhang et al., 2015).

Despite their successes, most of them focus on the discriminative models, leaving the field of generative models for implicit DRR a relatively uninvestigated area. In this respect, the most related work to ours is the latent variable recurrent neural network recently proposed by Ji et al. (2016). However, our work differs from theirs significantly, which can be summarized in the following three aspects: 1) they employ the recurrent neural network to represent the discourse arguments, while we use the simple feed-forward neural network; 2) they treat the discourse relations directly as latent variables, rather than the underlying semantic representation of discourses; 3) their model is optimized in terms of the data likelihood, since the discourse relations are observed during training. However, VarNDRR is optimized under the variational theory.

Variational Neural Model In the presence of continuous latent variables with intractable posterior distributions, efficient inference and learning in directed probabilistic models is required. Kingma and Welling (2014) as well as Rezende et al. (2014) introduce variational neural networks that employ an approximate inference model for intractable posterior and reparameterized variational lower bound for stochastic gradient optimization. Kingma et al. (2014) revisit the approach to semi-supervised learning with generative models and further develop new models that allow effective generalization from a small labeled dataset to a large unlabeled dataset. Chung et al. (2015) incorporate latent variables into the hidden state of a recurrent neural network, while Gregor et al. (2015) combine a novel spatial attention mechanism that mimics the foveation of human eyes, with a sequential variational auto-encoding framework that allows the iterative construction of

complex images.

We follow the spirit of these variational models, but focus on the adaptation and utilization of them onto implicit DRR, which, to the best of our knowledge, is the first attempt in this respect.

6 Conclusion and Future Work

In this paper, we have presented a variational neural discourse relation recognizer for implicit DRR. Different from conventional discriminative models that directly calculate the conditional probability of the relation $\mathbf{y}$ given discourse arguments $\mathbf{x}$, our model assumes that it is a latent variable from an underlying semantic space that generates both $\mathbf{x}$ and $\mathbf{y}$. In order to make the inference and learning efficient, we introduce a neural discourse recognizer and two neural latent approximators as our generative and inference model respectively. Using the reparameterization technique, we are able to optimize the whole model via standard stochastic gradient ascent algorithm. Experiment results in terms of classification and variational lower bound verify the effectiveness of our model.

In the future, we would like to exploit the utilization of discourse instances with explicit relations for implicit DRR. For this we can start from two directions: 1) converting explicit instances into pseudo implicit instances and retraining our model; 2) developing a semi-supervised model to leverage semantic information inside discourse arguments. Furthermore, we are also interested in adapting our model to other similar tasks, such as nature language inference.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos 61303082, 61672440, 61402388, 61622209 and 61403269), Natural Science Foundation of Fujian Province (Grant No. 2016J05161), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355), Research fund of the Provincial Key Laboratory for Computer Information Processing Technology in Soochow University (Grant No. KJS1520), and Research fund of the Key Laboratory for Intelligence Information Processing in the Institute of Computing Technology of the Chinese Academy of Sciences (Grant No. IIP2015-4). We also thank the anonymous reviewers for their insightful comments.

References

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *Proc. of COLING*, pages 1694–1705, August.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proc. of EMNLP*, pages 2201–2211.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Proc. of NIPS*.

Philipp Cimiano, Uwe Reyle, and Jasmin Šarić. 2005. Ontology-driven discourse analysis for information extraction. *Data & Knowledge Engineering*, 55:59–83.

Robert Fisher and Reid Simmons. 2015. Spectral semi-supervised discourse relation classification. In *Proc. of ACL-IJCNLP*, pages 89–93, July.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. In *Proc. of COLING*, pages 928–939.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *TACL*, pages 329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proc. of NAACL*, pages 332–342, June.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. of ICLR*.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proc. of NIPS*, pages 3581–3589.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging Synthetic Discourse Data via Multi-task Learning for Implicit Discourse Relation Recognition. In *Proc. of ACL*, pages 476–485, Sofia, Bulgaria, August.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proc. of EMNLP*, pages 343–351.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proc. of SIGDIAL*, pages 59–62, Tokyo, Japan, September.

Eleni Miltsakaki, Nikhil Dinesh, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Experiments on sense annotations and sense disambiguation of discourse connectives. In *Proc. of TLT2005*.

Joonsuk Park and Claire Cardie. 2012. Improving Implicit Discourse Relation Recognition Through Feature Set Optimization. In *Proc. of SIGDIAL*, pages 108–112, Seoul, South Korea, July.

Gary Patterson and Andrew Kehler. 2013. Predicting the presence of discourse connectives. In *Proc. of EMNLP*, pages 914–923.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*, page 884.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proc. of ACL-AFNLP*, pages 683–691, August.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. of ICML*, pages 1278–1286.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proc. of EACL*, pages 645–654, April.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proc. of NAACL-HLT*, pages 799–808, May–June.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proc. of SIGIR*, pages 735–736.

Xun Wang, Sujian Li, Jiwei Li, and Wenjie Li. 2012. Implicit discourse relation recognition by selecting typical training examples. In *Proc. of COLING*, pages 2757–2772.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proc. of EMNLP*, pages 1834–1839, October.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proc. of EMNLP*, September.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proc. of COLING*, pages 1507–1514.

Event Detection and Co-reference with Minimal Supervision

Haoruo Peng[1] and **Yangqiu Song**[2] and **Dan Roth**[1]
[1]University of Illinois, Urbana-Champaign
[2]Department of Computer Science and Engineering,
Hong Kong University of Science and Technology
[1]{hpeng7,danr}@illinois.edu, [2]yqsong@cse.ust.hk

Abstract

An important aspect of natural language understanding involves recognizing and categorizing events and the relations among them. However, these tasks are quite subtle and annotating training data for machine learning based approaches is an expensive task, resulting in supervised systems that attempt to learn complex models from small amounts of data, which they over-fit. This paper addresses this challenge by developing an event detection and co-reference system with minimal supervision, in the form of a few event examples. We view these tasks as semantic similarity problems between event mentions or event mentions and an ontology of types, thus facilitating the use of large amounts of out of domain text data. Notably, our semantic relatedness function exploits the structure of the text by making use of a semantic-role-labeling based representation of an event.

We show that our approach to event detection is competitive with the top supervised methods. More significantly, we outperform state-of-the-art supervised methods for event co-reference on benchmark data sets, and support significantly better transfer across domains.

1 Introduction

Natural language understanding involves, as a key component, the need to understand events mentioned in texts. This entails recognizing elements such as agents, patients, actions, location and time, among others. Understanding events also necessitates understanding relations among them and, as a minimum, determining whether two snippets of text represent the same event or not – the event co-reference problem. Events have been studied for years, but they still remain a key challenge. One reason is that the frame-based structure of events necessitates addressing multiple coupled problems that are not easy to study in isolation. Perhaps an even more fundamental difficulty is that it is not clear whether our current set of events' definitions is adequate (Hovy et al., 2013). Thus, given the complexity and fundamental difficulties, the current evaluation methodology in this area focuses on a limited domain of events, e.g. 33 types in ACE 2005 (NIST, 2005) and 38 types in TAC KBP (Mitamura et al., 2015). Consequently, this allows researchers to train supervised systems that are tailored to these sets of events and that overfit the small domain covered in the annotated data, rather than address the realistic problem of understanding events in text.

In this paper, we pursue an approach to understanding events that we believe to be more feasible and scalable. Fundamentally, event detection is about identifying whether an event in context is semantically related to a set of events of a specific type; and, event co-reference is about whether two event mentions are semantically similar enough to indicate that the author intends to refer to the same thing. Therefore, if we formulate event detection and co-reference as semantic relatedness problems, we can scale it to deal with a lot more types and, potentially, generalize across domains. Moreover, by doing so, we facilitate the use of a lot of data that is not part of the existing annotated event collections and not even from the same domain. The key chal-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 392–402,
Austin, Texas, November 1-5, 2016.

	Supervised	Unsupervised	MSEP
Guideline	✓	✓	✓
In-domain Data	✓	✓	✗
Data Annotation	✓	✗	✗

Table 1: **Comparing requirements of MSEP and other methods.** Supervised methods need all three resources while MSEP only needs an annotation guideline (as event examples).

lenges we need to address are those of how to represent events, and how to model event similarity; both are difficult partly since events have *structure*.

We present a general event detection and coreference framework, which essentially requires no labeled data. In practice, in order to map an event mention to an event ontology, as a way to communicate with a user, we just need a few event examples, in plain text, for each type a user wants to extract. This is a reasonable setting; after all, giving examples is the easiest way of defining event types, and is also how information needs are defined to annotators - by providing examples in the annotation guideline.[1] Our approach makes less assumptions than standard *unsupervised* methods, which typically require a *collection* of instances and exploit similarities among them to eventually learn a model. Here, given event type definitions (in the form of a few examples), we can classify a *single* event into a provided ontology and determine whether two events are co-referent. In this sense, our approach is similar to what has been called *dataless classification* (Chang et al., 2008; Song and Roth, 2014). Table 1 summarizes the difference between our approach, **MSEP** (Minimally Supervised Event Pipeline)[2], and other methods.

Our approach builds on two key ideas. First, to represent event structures, we use the general purpose nominal and verbial semantic role labeling (SRL) representation. This allows us to develop a structured representation of an event. Second, we embed event components, while maintaining the structure, into multiple semantic spaces, induced at a contextual, topical, and syntactic levels. These semantic representations are induced from large amounts of text in a way that is completely independent of the tasks at hand, and are used to represent both event mentions and event types into which we classify our events. The combination of these semantic spaces, along with the structured vector representation of an event, allow us to directly determine whether a candidate event mention is a valid event or not and, if it is, of which type. Moreover, with the same representation, we can evaluate event similarities and decide whether two event mentions are co-referent. Consequently, the proposed MSEP, can also adapt to new domains without any training.

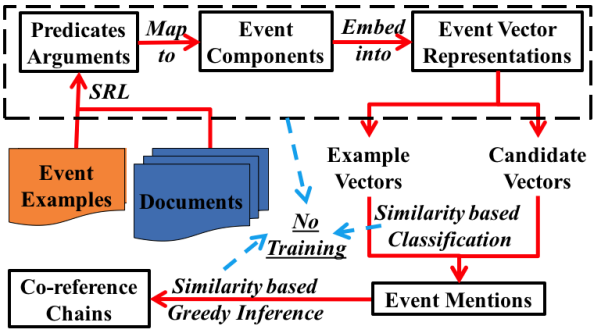


Figure 1: An overview of the end-to-end MSEP system. "Event Examples" are the only supervision here, which produce "Example Vectors". No training is needed for MSEP.

An overview of the system is shown in Figure 1. A few event examples are *all* the supervision MSEP needs; even the few decision thresholds needed to be set are determined on these examples, once and for all, and are used for *all* test cases we evaluate on. We use two benchmark datasets to compare MSEP with baselines and supervised systems. We show that MSEP performs favorably relative to state-of-the-art supervised systems; the co-reference module, in fact, outperforms supervised approaches on B^3 and CEAF metrics. The superiority of MSEP is also demonstrated in across domain settings.

2 The MSEP System

2.1 Structured Vector Representation

There is a parallel between event structures and sentence structures. Event triggers are mostly predicates of sentences or clauses. Predicates can be sense disambiguated, which roughly corresponds to

[1] Event examples also serve for disambiguation purposes. For example, using "U.S. forces bombed Baghdad." to exemplify an *attack* type, disambiguates it from a *heart attack*.

[2] Available at http://cogcomp.cs.illinois.edu/page/download_view/eventPipeline .

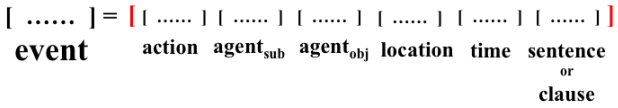


Figure 2: **Basic event vector representation.** Event vector is the concatenation of vectors corresponding to action, agent$_{sub}$, agent$_{obj}$, location, time and sentence/clause.

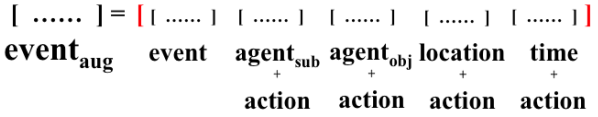


Figure 3: **Augmented event vector representation.** Event vector is the concatenation of vectors corresponding to basic event vector representation, agent$_{sub}$ + action, agent$_{obj}$ + action, location + action and time + action. Here, "+" means that we first put text fragments together and then convert the combined text fragment into an ESA vector.

event types. Event arguments are largely entity mentions or temporal/spatial arguments. They serve as specific roles in events, similarly to SRL arguments that are assigned role labels for predicates.

We use the Illinois SRL (Punyakanok et al., 2004) tool to pre-process the text. We evaluate the SRL coverage on both event triggers and event arguments, shown in Table 2.[3] For event triggers, we only focus on recall since we expect the event mention detection module to filter out most non-trigger predicates. Results show a good coverage of SRL predicates and arguments on event triggers and arguments. Even though we only get approximate event arguments, it is easier and more reliable to categorize them into five abstract roles, than to determine the exact role label with respect to event triggers.

We identify the five most important and abstract event semantic components: action, agent$_{sub}$, agent$_{obj}$, location and time. To map SRL arguments to these event arguments, we run through the following procedures: 1) set predicates as actions, and preserve SRL negations for actions, 2) set SRL subject as agent$_{sub}$, 3) set SRL object and indirect object as agent$_{obj}$, 4) set SRL spatial argument as event location. If there is no such SRL label, we then scan for any NER location label within the sentence/clause to which the action belongs. We set the location according to NER information if it exists. 5) We set the SRL temporal argument as event time. If there is no such SRL label, we then use the Illinois Temporal Expression Extractor (Zhao et al., 2012) to find the temporal argument within an event's sentence/clause. 6) We allow one or more missing event arguments among agent$_{sub}$, agent$_{obj}$, location or time, but require actions to always exist.

Given the above structured information, we convert each event component to its corresponding vector representation, discussed in detail in Section 3. We then concatenate the vectors of all components together in a specific order: action, agent$_{sub}$, agent$_{obj}$, location, time and sentence/clause. We treat the whole sentence/clause, to which the "ac-

ACE		Precision	Recall	F1
Predicates	Verb-SRL	—	93.2	—
over	Nom-SRL	—	87.5	—
Triggers	All	—	91.9	—
SRL Args	Verb-SRL	90.4	85.7	88.0
over	Nom-SRL	92.5	73.5	81.9
Event Args	All	90.9	82.3	86.4
TAC KBP		**Precision**	**Recall**	**F1**
Predicates	Verb-SRL	—	90.6	—
over	Nom-SRL	—	85.5	—
Triggers	All	—	88.1	—
SRL Args	Verb-SRL	89.8	83.6	86.6
over	Nom-SRL	88.2	69.9	78.0
Event Args	All	89.5	81.0	85.0

Table 2: **Semantic role labeling coverage.** We evaluate both "Predicates over Triggers" and "SRL Arguments over Event Arguments". "All" stands for the combination of Verb-SRL and Nom-SRL. The evaluation is done on all data.

[3] We place events in two categories, verb or noun, according to the part-of-speech tag of the trigger. We evaluate verb-SRL on events with verb triggers, nom-SRL on events with noun triggers, and the overall performance on all events. When evaluating, we allow partial overlaps.

tion" belongs, as context, and we append its corresponding vector to the event representation. This basic event vector representation is illustrated in Fig. 2. If there are missing event arguments, we set the corresponding vector to be "NIL" (we set each position as "NaN"). We also augment the event vector representation by concatenating more text fragments to enhance the interactions between the action and other arguments, as shown in Fig. 3. Essentially, we flatten the event structure to preserve the alignment of event arguments so that the structured information can be reflected in our vector space.

2.2 Event Mention Detection

Motivated by the seed-based event trigger labeling technique employed in Bronstein et al. (2015), we turn to ACE annotation guidelines for event examples described under each event type label. For instance, the ACE-2005 guidelines list the example "Mary Smith joined Foo Corp. in June 1998." for label "START-POSITION". Altogether, we collect 172 event examples from 33 event types (5 each on average).[4] We can then get vector representations for these example events following the procedures in Sec. 2.1. We define the *event type representation* as the numerical average of all vector representations corresponding to example events under that type. We use the similarity between an event candidate with the event type representation to determine whether the candidate belongs to an event type:

$$S(e_1, e_2) = \frac{vec(e_1) \cdot vec(e_2)}{\|vec(e_1)\| \cdot \|vec(e_2)\|} = \frac{\sum_a vec(a_1) \cdot vec(a_2)}{\sqrt{\sum_a \|vec(a_1)\|^2} \cdot \sqrt{\sum_a \|vec(a_2)\|^2}}, \quad (1)$$

where e_1 is the candidate, e_2 the type ($vec(e_2)$ is computed as average of event examples), a_1, a_2 are components of e_1, e_2 respectively. We use the notation $vec(\cdot)$ for corresponding vectors. Note that there may be missing event arguments (NIL). In such cases, we use the average of all non-NIL similarity scores for that particular component as the contributed score. Formally, we define $S_{pair}(a = \text{NIL})$ and $S_{single}(a = \text{NIL})$ as follows:

$$\begin{aligned} S_{pair}(a = \text{NIL}) &= vec(\text{NIL}) \cdot vec(a_2) \\ &= vec(a_1) \cdot vec(\text{NIL}) \\ &= \sum_{a_1, a_2 \neq \text{NIL}} \frac{vec(a_1) \cdot vec(a_2)}{\#|a_1, a_2 \neq \text{NIL}|}, \end{aligned}$$

$$S_{single}(a = \text{NIL}) = \sqrt{\frac{\sum_{a \neq \text{NIL}} \|vec(a)\|^2}{\#|a \neq \text{NIL}|}}.$$

Thus, when we encounter missing event arguments, we use $S_{pair}(a = \text{NIL})$ to replace the corresponding term in the numerator in $S(e_1, e_2)$ while using $S_{single}(a = \text{NIL})$ in the denominator. These average contributed scores are corpus independent, and can be pre-computed ahead of time. We use a cut-off threshold to determine that an event does not belong to any event types, and can thus be eliminated. This threshold is set by tuning only on the set of event examples, which is corpus independent.[5]

2.3 Event Co-reference

Similar to the mention-pair model in entity coreference (Ng and Cardie, 2002; Bengtson and Roth, 2008; Stoyanov et al., 2010), we use cosine similarities computed from pairs of event mentions: $S(e_1, e_2)$ (as in Eq. (1)).

Before applying the co-reference model, we first use external knowledge bases to identify conflict events. We use the Illinois Wikification (Cheng and Roth, 2013) tool to link event arguments to Wikipedia pages. Using the Wikipedia IDs, we map event arguments to Freebase entries. We view the top-level Freebase type as the event argument type. An event argument can contain multiple wikified entities, leading to multiple Wikipedia pages and thus a set of Freebase types. We also augment the argument type set with NER labels: PER (person) and ORG (organization). We add either of the NER labels if we detect such a named entity.

For each pair of events, we check event arguments agent$_{sub}$ and agent$_{obj}$ respectively. If none of the types for the aligned event arguments match, this pair is determined to be in conflict. If the event argument is missing, we deem it compatible with any type. In this procedure, we generate a set of event pairs Set_{conflict} that will not get co-reference links.

[4] See supplementary materials for the full list of examples.

[5] See Sec. 4.4 for details.

Given the event mention similarity as well as the conflicts, we perform event co-reference inference via a left-linking greedy algorithm, i.e. co-reference decisions are made on each event from left to right, one at a time. Without loss of generality, for event $e_{k+1}, \forall k \geq 1$, we first choose a linkable event to its left with the highest event-pair similarity:

$$e_p = \arg \max_{\substack{e \in \{e_1, e_2, \ldots, e_k\} \\ e \notin Set_{\text{conflict}}}} S(e, e_{k+1}).$$

We make co-reference links when $S(e_p, e_{k+1})$ is higher than a cut-off threshold, which is also tuned only on event examples ahead of time. Otherwise, event e_{k+1} is not similar enough to any of its antecedents, and we make it the start of a new cluster.

3 Vector Representations

We experiment with different methods to convert event components into vector representations. Specifically, we use Explicit Semantic Analysis (ESA), Brown Cluster (BC), Word2Vec (W2V) and Dependency-Based Word Embedding (DEP) respectively to convert text into vectors. We then concatenate all components of an event together to form a structured vector representation.

Explicit Semantic Analysis ESA uses Wikipedia as an external knowledge base to generate concepts for a given fragment of text (Gabrilovich and Markovitch, 2009). ESA first represents a given text fragment as a TF-IDF vector, then uses an inverted index for each word to search the Wikipedia corpus. The text fragment representation is thus a weighted combination of the concept vectors corresponding to its words. We use the same setting as in Chang et al. (2008) to filter out pages with fewer than 100 words and those containing fewer than 5 hyperlinks. To balance between the effectiveness of ESA representations and its cost, we use the 200 concepts with the highest weights. Thus, we convert each text fragment to a very sparse vector of millions of dimensions (but we just store 200 non-zero values).

Brown Cluster BC was proposed by Brown et al. (1992) as a way to support abstraction in NLP tasks, measuring words' distributional similarities. This method generates a hierarchical tree of word clusters by evaluating the word co-occurrence based on a n-gram model. Then, paths traced from root to leaves can be used as word representations. We use the implementation by Song and Roth (2014), generated over the latest Wikipedia dump. We set the maximum tree depth to 20, and use a combination of path prefixes of length 4,6 and 10 as our BC representation. Thus, we convert each word to a vector of $2^4 + 2^6 + 2^{10} = 1104$ dimensions.

Word2Vec We use the skip-gram tool by Mikolov et al. (2013) over the latest Wikipedia dump, resulting in word vectors of dimensionality 200.

Dependency-Based Embedding DEP is the generalization of the skip-gram model with negative sampling to include arbitrary contexts. In particular, it deals with dependency-based contexts, and produces markedly different embeddings. DEP exhibits more functional similarity than the original skip-gram embeddings (Levy and Goldberg, 2014). We directly use the released 300-dimension word embeddings[6].

Note that it is straightforward text-vector conversion for ESA. But for BC, W2V and DEP, we first remove stop words from the text and then average, element-wise, all remaining word vectors to produce the resulting vector representation of the text fragment.

4 Experiments

4.1 Datasets

ACE The ACE-2005 English corpus (NIST, 2005) contains fine-grained event annotations, including event trigger, argument, entity, and time-stamp annotations. We select 40 documents from newswire articles for event detection evaluation and the rest for training (same as Chen et al. (2015)). We do 10-fold cross-validation for event co-reference.

TAC-KBP The TAC-KBP-2015 corpus is annotated with event nuggets that fall into 38 types and co-reference relations between events. [7] We use the train/test data split provided by the official TAC-

[6] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings

[7] The event ontology of TAC-KBP (based on ERE annotation) is almost the same to that of ACE. To adapt our system to the TAC-KBP corpus, we use all ACE event seeds of "Contact.Phone-Write" for "Contact.Correspondence" and separate ACE event seeds of "Movement.Transport" into "Movement.TransportPerson" and "Movement.TransportArtifact" by manual checking. So, we use exactly the same set of event seeds for TAC-KBP with only these two changes.

	#Doc	#Sent.	#Men.	#Cluster
ACE(All)	599	15,494	5,268	4,046
ACE(Test)	40	672	289	222
TAC-KBP(All)	360	15,824	12,976	7,415
TAC-KBP(Test)	202	8,851	6,438	3,779

Table 3: **Statistics for the ACE and TAC-KBP corpora.** #Sent. is the number of sentences, #Men. is the number of event mentions, and #Cluster is the number of event clusters (including singletons). Note that the proposed MSEP does not need any training data.

2015 Event Nugget Evaluation Task.

Statistics for the ACE and TAC-KBP corpora is shown in Table 3. Note that the training set and cross-validation is only for competing supervised methods. For MSEP, we only need to run on each corpus once for testing.

4.2 Compared Systems

For event detection, we compare with **DM-CNN** (Chen et al., 2015), the state-of-art supervised event detection system. We also implement another supervised model, named *supervised structured event detection* **SSED** system following the work of Sammons et al. (2015). The system utilizes rich semantic features and applies a trigger identification classifier on every SRL predicate to determine the event type. For event co-reference, **Joint** (Chen et al., 2009) is an early work based on supervised learning. We also report **HDP-Coref** results as an unsupervised baseline (Bejan and Harabagiu, 2010), which utilizes nonparametric Bayesian models. Moreover, we create another unsupervised event co-reference baseline (**Type+SharedMen**): we treat events of the same type which share at least one co-referent entity (inside event arguments) as co-referred. On TAC-KBP corpus, we report results from the top ranking system of the TAC-2015 Event Nugget Evaluation Task as **TAC-TOP**.

We name our event mention detection module in MSEP *similarity-based event mention detection* **MSEP-EMD** system. For event co-reference, the proposed similarity based co-reference detection **MSEP-Coref** method has a number of variations depending on the modular text-vector conversion method (**ESA, BC, W2V, DEP**), whether we use augmented ESA vector representation (**AUG**)[8], and whether we use knowledge during co-reference inference (**KNOW**). We also develop a supervised event co-reference system following the work of Sammons et al. (2015), namely **Supervised$_{\textbf{Base}}$**. We also add additional event vector representations[9] as features to this supervised system and get **Supervised$_{\textbf{Extend}}$**.

4.3 Evaluation Metrics

For event detection, we use standard precision, recall and F1 metrics. For event co-reference, we compare all systems using standard F1 metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), Entity-based CEAF (CEAF$_e$) (Luo, 2005) and BLANC (Recasens and Hovy, 2011). We use the average scores (AVG) of these four metrics as the main comparison metric.[10]

4.4 Results for Event detection

The performance comparison for event detection is presented in Table 4. On both ACE and TAC-KBP, parameters of SSED are tuned on a development set (20% of randomly sampled training documents). The cut-off threshold for MSEP-EMD is tuned on the 172 event examples ahead of time by optimizing the F1 score on the event seed examples. Note that different text-vector conversion methods lead to different cut-off thresholds, but they remain fixed for all the test corpus. Results show that SSED achieves state-of-the-art performance. Though MSEP-EMD's performance is below the best supervised system, it is very competitive. Note that both SSED and MSEP-EMD use SRL predicates as input and thus can further improve with a better SRL module.

4.5 Results for Event Co-reference

The performance of different systems for event co-reference based on gold event triggers is shown in Table 5. The co-reference cut-off threshold is tuned by optimizing the CoNLL average score on ten se-

[8]It is only designed for ESA because the ESA vector for two concatenated text fragments is different from the sum of the ESA vectors of individual text fragments, unlike other methods.

[9]We add the best event vector representation empirically.

[10]We use the latest scorer (v1.7) provided by TAC-2015 Event Nugget Evaluation for all metrics.

ACE (Test Data)		Precision	Recall	F1
Span	DMCNN	80.4	67.7	73.5
	SSED	76.6	71.5	**74.0**
	MSEP-EMD	75.6	69.8	72.6
Span+Type	DMCNN	75.6	63.6	**69.1**
	SSED	71.3	66.5	68.8
	MSEP-EMD	70.4	65.0	67.6
TAC-KBP (Test Data)		Precision	Recall	F1
Span	SSED	77.2	55.9	64.8
	TAC-TOP	—	—	**65.3**
	MSEP-EMD	76.5	54.5	63.5
Span+Type	SSED	69.9	48.8	57.5
	TAC-TOP	—	—	**58.4**
	MSEP-EMD	69.2	47.8	56.6

Table 4: **Event detection (trigger identification) results.** "Span"/"Type" means span/type match respectively.

lected ACE documents. The threshold is then fixed, thus we do not change it when evaluating on the TAC-KBP corpus. As we do cross-validation on ACE, we exclude these ten documents from test at all times.[11] Results show that the proposed MSEP event co-reference system significantly outperforms baselines and achieves the same level of performance of supervised methods (82.9 v.s. 83.3 on ACE and 73.8 v.s. 74.4 on TAC-KBP). MSEP achieves better results on B^3 and $CEAF_e$ than supervised methods. Note that supervised methods usually generate millions of features (2.5M on ACE and 1.8M on TAC-KBP for $Supervised_{Base}$). In contrast, MSEP only has several thousands of non-zero dimensions in event representations. This means that our structured vector representations, through derived without explicit annotations, are far more expressive than traditional features. When we add the event vector representation (augmented ESA) as features in $Supervised_{Extend}$, we improve the overall performance by more than 1 point. When tested individually, DEP performs the best among the four text-vector conversion methods while BC performs the worst. A likely reason is that BC has too few dimensions while DEP constructs the longest vector. However, the results show that our augmented ESA representation (Fig. 2) achieves even better results.

When we use knowledge to detect conflicting events during inference, the system further improves. Note that event arguments for the proposed MSEP are predicted by SRL. We show that replacing them with gold event arguments, only slightly improves the overall performance, indicating that SRL arguments are robust enough for the event co-reference task.

4.6 End-to-End Event Co-reference Results

Table 6 shows the performance comparison for end-to-end event co-reference. We use both SSED and MSEP-EMD as event detection modules and we evaluate on standard co-reference metrics. Results on TAC-KBP show that "SSED+$Supervised_{Extend}$" achieves similar performance to the TAC top ranking system while the proposed MSEP event co-reference module helps to outperform supervised methods on B^3 and $CEAF_e$ metrics.

4.7 Domain Transfer Evaluation

To demonstrate the superiority of the adaptation capabilities of the proposed MSEP system, we test its performance on new domains and compare with the supervised system. TAC-KBP corpus contains two genres: newswire (NW) and discussion forum (DF), and they have roughly equal number of documents. When trained on NW and tested on DF, supervised methods encounter out-of-domain situations. However, the MSEP system can adapt well.[12] Table 7 shows that MSEP outperforms supervised methods in out-of-domain situations for both tasks. The differences are statistically significant with $p < 0.05$.

5 Related Work

Event detection has been studied mainly in the newswire domain as the task of detecting event triggers and determining event types and arguments. Most earlier work has taken a pipeline approach where local classifiers identify triggers first, and then arguments (Ji and Grishman, 2008; Liao and

[11] We regard this tuning procedure as "independent" and "ahead of time" because of the following reasons: 1) We could have used as threshold-tuning co-reference examples a few news documents from other sources; we just use ACE documents as a data source for simplicity. 2) We believe that the threshold only depends on event representation (the model) rather than data. 3) Tuning a single decision threshold is much cheaper than tuning a whole set of model parameters.

[12] Note that the supervised method needs to be re-trained and its parameters re-tuned while MSEP does not need training and its cut-off threshold is fixed ahead of time using event examples.

ACE (Cross-Validation)		MUC	B^3	$CEAF_e$	BLANC	AVG
Supervised	Graph	—	—	84.5	—	—
	Joint	74.8	92.2	87.0	—	—
	$Supervised_{Base}$	73.6	91.6	85.9	82.2	83.3
	$Supervised_{Extend}$	**74.9**	92.8	87.1	**83.8**	**84.7**
Unsupervised	Type+SharedMen	59.1	83.2	76.0	72.9	72.8
	HDP-Coref	—	83.8	76.7	—	—
MSEP	$MSEP\text{-}Coref_{ESA}$	65.9	91.5	85.3	81.8	81.1
	$MSEP\text{-}Coref_{BC}$	65.0	89.8	83.7	80.9	79.9
	$MSEP\text{-}Coref_{W2V}$	65.1	90.1	83.6	81.5	80.1
	$MSEP\text{-}Coref_{DEP}$	65.9	92.3	85.6	81.5	**81.3**
	$MSEP\text{-}Coref_{ESA+AUG}$	67.4	92.6	86.0	82.6	**82.2**
	$MSEP\text{-}Coref_{ESA+AUG+KNOW}$	68.0	**92.9**	**87.4**	83.2	**82.9**
	$MSEP\text{-}Coref_{ESA+AUG+KNOW}$ (GA)	68.8	92.5	87.7	83.4	83.1
TAC-KBP (Test Data)		MUC	B^3	$CEAF_e$	BLANC	AVG
Supervised	TAC-TOP	—	—	—	—	**75.7**
	$Supervised_{Base}$	63.8	83.8	75.8	74.0	74.4
	$Supervised_{Extend}$	**65.3**	84.7	76.8	**75.1**	**75.5**
Unsupervised	Type+SharedMen	56.4	77.5	69.6	68.7	68.1
MSEP	$MSEP\text{-}Coref_{ESA}$	57.7	83.9	76.9	72.9	72.9
	$MSEP\text{-}Coref_{BC}$	56.9	81.8	76.2	71.7	71.7
	$MSEP\text{-}Coref_{W2V}$	57.2	82.1	75.9	72.3	71.9
	$MSEP\text{-}Coref_{DEP}$	58.2	83.3	76.7	72.8	**72.8**
	$MSEP\text{-}Coref_{ESA+AUG}$	59.0	84.5	77.3	72.5	**73.3**
	$MSEP\text{-}Coref_{ESA+AUG+KNOW}$	59.9	**84.9**	**77.3**	73.1	**73.8**
	$MSEP\text{-}Coref_{ESA+AUG+KNOW}$ (GA)	60.5	84.0	77.7	73.5	73.9

Table 5: **Event Co-reference Results on Gold Event Triggers.** "$MSEP\text{-}Coref_{ESA,BC,W2V,DEP}$" are variations of the proposed MSEP event co-reference system using ESA, Brown Cluster, Word2Vec and Dependency Embedding representations respectively. "$MSEP\text{-}Coref_{ESA+AUG}$" uses augmented ESA event vector representation and "$MSEP\text{-}Coref_{ESA+AUG+KNOW}$" applies knowledge to detect conflicting events. (GA) means that we use gold event arguments instead of approximated ones from SRL.

Grishman, 2010; Hong et al., 2011; Huang and Riloff, 2012a; Huang and Riloff, 2012b). Li et al. (2013) presented a structured perceptron model to detect triggers and arguments jointly. Attempts have also been made to use a Distributional Semantic Model (DSM) to represent events (Goyal et al., 2013). A shortcoming of DSMs is that they ignore the structure within the context, thus reducing the distribution to a bag of words. In our work, we preserve event structure via structured vector representations constructed from event components.

Event co-reference is much less studied in comparison to the large body of work on entity co-reference. Our work follows the event co-reference definition in Hovy et al. (2013). All previous work on event co-reference except Cybulska and Vossen (2012) deals only with full co-reference. Early works (Humphreys et al., 1997; Bagga and Baldwin, 1999) performed event co-reference on scenario specific events. Both Naughton (2009) and Elkhlifi and Faiz (2009) worked on sentence-level co-reference, which is closer to the definition of Danlos and Gaiffe (2003). Pradhan et al. (2007) dealt with both entity and event coreference by taking a three-layer approach. Chen and Ji (2009) proposed a clustering algorithm using a maximum entropy model with a range of features. Bejan and Harabagiu (2010) built a class of nonparametric Bayesian models using a (potentially infinite) number of features to resolve both within and cross document event co-reference. Lee et al. (2012) formed a system with deterministic layers to make co-reference decisions iteratively while jointly resolving entity and event co-reference. More recently, Hovy et al. (2013) presented an unsupervised model to capture semantic relations and co-reference resolution, but they did not show quantitatively how well their system performed in each of these two cases. Huang et al. (2016) also considered

ACE (Cross-Validation)		MUC	B^3	$CEAF_e$	BLANC	AVG
SSED	+ $Supervised_{Extend}$	**47.1**	59.9	58.7	44.4	**52.5**
SSED	+ MSEP-$Coref_{ESA+AUG+KNOW}$	42.1	**60.3**	**59.0**	44.1	51.4
MSEP-EMD	+ MSEP-$Coref_{ESA+AUG+KNOW}$	40.2	58.6	57.4	43.8	50.0
TAC-KBP (Test Data)		MUC	B^3	$CEAF_e$	BLANC	AVG
TAC-TOP		—	—	—	—	**39.1**
SSED	+ $Supervised_{Extend}$	**34.9**	44.2	39.6	**37.1**	**39.0**
SSED	+ MSEP-$Coref_{ESA+AUG+KNOW}$	33.1	**44.6**	**39.7**	36.8	38.5
MSEP-EMD	+ MSEP-$Coref_{ESA+AUG+KNOW}$	30.2	43.9	38.7	35.7	37.1

Table 6: Event Co-reference End-To-End Results.

	Train	Test	MSEP	Supervised
Event Detection				Span+Type F1
In Domain	NW	NW	58.5	**63.7**
Out of Domain	DF	NW	**55.1**	54.8
In Domain	DF	DF	57.9	**62.6**
Out of Domain	NW	DF	**52.8**	52.3
Event Co-reference				AVG F1
In Domain	NW	NW	73.2	**73.6**
Out of Domain	DF	NW	**71.0**	70.1
In Domain	DF	DF	68.6	**68.9**
Out of Domain	NW	DF	**67.9**	67.0

Table 7: **Domain Transfer Results.** We conduct the evaluation on TAC-KBP corpus with the split of newswire (NW) and discussion form (DF) documents. Here, we choose MSEP-EMD and MSEP-$Coref_{ESA+AUG+KNOW}$ as the MSEP approach for event detection and co-reference respectively. We use SSED and $Supervised_{Base}$ as the supervised modules for comparison. For event detection, we compare F1 scores of span plus type match while we report the average F1 scores for event co-reference.

the problem of event clustering. They represented event structures based on AMR (Abstract Meaning Representation) and distributional semantics, and further generated event schemas composing event triggers and argument roles. Recently, TAC has organized Event Nugget Detection and Co-reference Evaluations, resulting in interesting works, some of which contributed to our comparisons (Liu et al., 2015; Mitamura et al., 2015; Hsi et al., 2015; Sammons et al., 2015).

6 Conclusion

This paper proposes a novel event detection and co-reference approach with minimal supervision, addressing some of the key issues slowing down progress in research on events, including the difficulty to annotate events and their relations. At the heart of our approach is the design of structured vector representations for events which, as we show, supports a good level of generalization within and across domains. The resulting approach outperforms state-of-art supervised methods on some of the key metrics, and adapts significantly better to a new domain. One of the key research directions is to extend this unsupervised approach to a range of other relations among events, including temporal and causality relations, as is (Do et al., 2011; Do et al., 2012).

Acknowledgments

The authors would like to thank Eric Horn for comments that helped to improve this work. This material is based on research sponsored by the US Defense Advanced Research Projects Agency (DARPA) under agreements FA8750-13-2-000 and HR0011-15-2-0025. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *MUC-7*.

A. Bagga and B. Baldwin. 1999. Cross-document event coreference: Annotations, experiments, and observations. In *Proceedings of the Workshop on Coreference and its Applications*.

C. A. Bejan and S. Harabagiu. 2010. Unsupervised event

coreference resolution with rich linguistic features. In *ACL*.

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*.

O. Bronstein, I. Dagan, Q. Li, H. Ji, and A. Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *ACL*.

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

M. Chang, L. Ratinov, D. Roth, and V. Srikumar. 2008. Importance of semantic represenation: Dataless classification. In *AAAI*.

Z. Chen and H. Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the Workshop on Graph-based Methods for Natural Language Processing*.

Z. Chen, H. Ji, and R. Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*.

Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *EMNLP*.

A. Cybulska and P. Vossen. 2012. Using semantic relations to solve event coreference in text. In *Proceedings of the Workshop on Semantic relations*.

L. Danlos and B. Gaiffe. 2003. Event coreference and discourse relations. *Philosophical Studies Series*.

Q. Do, Y. S. Chan, and D. Roth. 2011. Minimally supervised event causality extraction. In *EMNLP*.

Q. Do, W. Lu, and D. Roth. 2012. Joint inference for event timeline construction. In *EMNLP*.

A. Elkhlifi and R. Faiz. 2009. Automatic annotation approach of events in news articles. *International Journal of Computing & Information Sciences*.

Evgeniy Gabrilovich and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Int. Res.*, 34(1):443–498, March.

K. Goyal, S. K. Jauhar, H. Li, M. Sachan, S. Srivastava, and E. Hovy. 2013. A structured distributional semantic model for event co-reference. In *ACL*.

Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

E. Hovy, T.o Mitamura, F. Verdejo, J. Araki, and A. Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *NAACL-HLT*.

A. Hsi, J. Carbonell, and Y. Yang. 2015. Modeling event extraction via multilingual data sources. In *TAC*.

R. Huang and E. Riloff. 2012a. Bootstrapped training of event extraction classifiers. In *EACL*.

R. Huang and E. Riloff. 2012b. Modeling textual cohesion for event extraction. In *AAAI*.

L. Huang, T. Cassidy, X. Feng, H. Ji, C. R. Voss, J. Han, and A. Sil. 2016. Liberal event extraction and event schema induction. In *ACL*.

K. Humphreys, R. Gaizauskas, and S. Azzam. 1997. Event coreference for information extraction. In *Proceedings of Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*.

H. Ji and R. Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

H. Lee, M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *EMNLP*.

O. Levy and Y. Goldberg. 2014. Dependencybased word embeddings. In *ACL*.

Q. Li, H. Ji, and L. Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

S. Liao and R. Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*.

Z. Liu, T. Mitamura, and E. Hovy. 2015. Evaluation algorithms for event nugget detection: A pilot study. In *Proceedings of the Workshop on Events at the NAACL-HLT*.

X. Luo. 2005. On coreference resolution performance metrics. In *EMNLP*.

T. Mikolov, W. Yih, and G. Zweig. 2013. Linguistic regularities in continuous space word representations. In *NAACL*.

T. Mitamura, Y. Yamakawa, S. Holm, Z. Song, A. Bies, S. Kulick, and S. Strassel. 2015. Event nugget annotation: Processes and issues. In *Proceedings of the Workshop on Events at NAACL-HLT*.

M. Naughton. 2009. *Sentence Level Event Detection and Coreference Resolution*. Ph.D. thesis, National University of Ireland, Dublin.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*.

NIST. 2005. The ACE evaluation plan.

S. Pradhan, L. Ramshaw, R. Weischedel, J. MacBride, and L. Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. In *ICSC*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *COLING*.

M. Recasens and E. Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17(04):485–510.

M. Sammons, H. Peng, Y. Song, S. Upadhyay, C.-T. Tsai, P. Reddy, S. Roy, and D. Roth. 2015. Illinois ccg tac 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *TAC*.

Y. Song and D. Roth. 2014. On dataless hierarchical text classification. In *AAAI*.

V. Stoyanov, C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D. Hysom. 2010. Coreference resolution with reconcile. In *ACL*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*.

R. Zhao, Q. Do, and D. Roth. 2012. A robust shallow temporal reasoning system. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT Demo)*.

Learning Term Embeddings for Taxonomic Relation Identification Using Dynamic Weighting Neural Network

Luu Anh Tuan
Institute for Infocomm Research, Singapore
at.luu@i2r.a-star.edu.sg

Yi Tay
Nanyang Technological University
ytay2@e.ntu.edu.sg

Siu Cheung Hui
Nanyang Technological University
asschui@ntu.edu.sg

See Kiong Ng
Institute for Infocomm Research, Singapore
sknq@i2r.a-star.edu.sg

Abstract

Taxonomic relation identification aims to recognize the *'is-a'* relation between two terms. Previous works on identifying taxonomic relations are mostly based on statistical and linguistic approaches, but the accuracy of these approaches is far from satisfactory. In this paper, we propose a novel supervised learning approach for identifying taxonomic relations using term embeddings. For this purpose, we first design a *dynamic weighting neural network* to learn term embeddings based on not only the hypernym and hyponym terms, but also the contextual information between them. We then apply such embeddings as features to identify taxonomic relations using a supervised method. The experimental results show that our proposed approach significantly outperforms other state-of-the-art methods by 9% to 13% in terms of accuracy for both general and specific domain datasets.

1 Introduction

Taxonomies which serve as the backbone of structured knowledge are useful for many NLP applications such as question answering (Harabagiu et al., 2003) and document clustering (Fodeh et al., 2011). However, the hand-crafted, well-structured taxonomies including WordNet (Miller, 1995), OpenCyc (Matuszek et al., 2006) and Freebase (Bollacker et al., 2008) that are publicly available may not be complete for new or specialized domains. It is also time-consuming and error prone to identify taxonomic relations manually. As such, methods for automatic identification of taxonomic relations is highly desirable.

The previous methods for identifying taxonomic relations can be generally classified into two categories: statistical and linguistic approaches. The statistical approaches rely on the idea that frequently co-occurring terms are likely to have taxonomic relationships. While such approaches can result in taxonomies with relatively high coverage, they are usually heavily dependent on the choice of feature types, and suffer from low accuracy. The linguistic approaches which are based on lexical-syntactic patterns (e.g. *'A such as B'*) are simple and efficient. However, they usually suffer from low precision and coverage because the identified patterns are unable to cover the wide range of complex linguistic structures, and the ambiguity of natural language compounded by data sparsity makes these approaches less robust.

Word embedding (Bengio et al., 2001), also known as distributed word representation, which represents words with high-dimensional and real-valued vectors, has been shown to be effective in exploring both linguistic and semantic relations between words. In recent years, word embedding has been used quite extensively in NLP research, ranging from syntactic parsing (Socher et al., 2013a), machine translation (Zou et al., 2013) to sentiment analysis (Socher et al., 2013b). The current methods for learning word embeddings have focused on learning the representations from word co-occurrence so that similar words will have similar embeddings. However, using the co-occurrence based similarity learning alone is not effective for the purpose of identifying taxonomic relations.

Recently, Yu et al. (2015) proposed a super-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 403–413,
Austin, Texas, November 1-5, 2016.

vised method to learn term embeddings based on pre-extracted taxonomic relation data. However, this method is heavily dependent on the training data to discover all taxonomic relations, i.e. if a pair of terms is not in the training set, it may become a negative example in the learning process, and will be classified as a non-taxonomic relation. The dependency on training data is a huge drawback of the method as no source can guarantee that it can cover all possible taxonomic relations for learning. Moreover, the recent studies (Velardi et al., 2013; Levy et al., 2014; Tuan et al., 2015) showed that contextual information between hypernym and hyponym is an important indicator to detect taxonomic relations. However, the term embedding learning method proposed in (Yu et al., 2015) only learns through the pairwise relations of terms without considering the contextual information between them. Therefore, the resultant quality is not good in some specific domain areas.

In this paper, we propose a novel approach to learn term embeddings based on *dynamic weighting neural network* to encode not only the information of hypernym and hyponym, but also the contextual information between them for the purpose of taxonomic relation identification. We then apply the identified embeddings as features to find the positive taxonomic relations using the supervised method SVM. The experimental results show that our proposed term embedding learning approach outperforms other state-of-the-art embedding learning methods for identifying taxonomic relations with much higher accuracy for both general and specific domains. In addition, another advantage of our proposed approach is that it is able to generalize from the training dataset the taxonomic relation properties for unseen pairs. Thus, it can recognize some true taxonomic relations which are not even defined in dictionary and training data. For the rest of this paper, we will discuss the proposed term embedding learning approach and its performance results.

2 Related work

Previous works on taxonomic relation identification can be roughly divided into two main approaches of statistical learning and linguistic pattern matching. Statistical learning methods include co-occurrence analysis (Lawrie and Croft, 2003), hierarchical latent Dirichlet allocation (LDA) (Blei et al., 2004; Petinot et al., 2011), clustering (Li et al., 2013), linguistic feature-based semantic distance learning (Yu et al., 2011), distributional representation (Roller et al., 2014; Weeds et al., 2014; Kruszewski et al., 2015) and co-occurrence subnetwork mining (Wang et al., 2013). Supervised statistical methods (Petinot et al., 2011) rely on hierarchical labels to learn the corresponding terms for each label. These methods require labeled training data which is costly and not always available in practice. Unsupervised statistical methods (Pons-Porrata et al., 2007; Li et al., 2013; Wang et al., 2013) are based on the idea that terms that frequently co-occur may have taxonomic relationships. However, these methods generally achieve low accuracies.

Linguistic approaches rely on lexical-syntactic patterns (Hearst, 1992) (e.g. *'A such as B'*) to capture textual expressions of taxonomic relations, and match them with the given documents or Web information to identify the relations between a term and its hypernyms (Kozareva and Hovy, 2010; Navigli et al., 2011; Wentao et al., 2012). These patterns can be manually created (Kozareva and Hovy, 2010; Wentao et al., 2012) or automatically identified (Snow et al., 2004; Navigli et al., 2011). Such liguistic pattern matching methods can generally achieve higher precision than the statistical methods, but they suffer from lower coverage. To balance the precision and recall, Zhu *et al.* (2013) and Tuan *et al.* (2014) have combined both unsupervised statistical and linguistic methods for finding taxonomic relations.

In recent years, there are a few studies on taxonomic relation identification using word embeddings such as the work of Tan et al. (2015) and Fu et al. (2014). These studies are based on word embeddings from the Word2Vec model (Mikolov et al., 2013a), which is mainly optimized for the purpose of analogy detection using co-occurrence based similarity learning. As such, these studies suffer from poor performance on low accuracy for taxonomic relation identification.

The approach that is closest to our work is the one proposed by Yu et al. (2015), which also learns term embeddings for the purpose of taxonomic relation

identification. In the approach, a distance-margin neural network is proposed to learn term embeddings based on the pre-extracted taxonomic relations from the Probase database (Wentao et al., 2012). However, the neural network is trained using only the information of the term pairs (i.e. hypernym and hyponym) without considering the contextual information between them, which has been shown to be an important indicator for identifying taxonomic relations from previous studies (Velardi et al., 2013; Levy et al., 2014; Tuan et al., 2014). Moreover, if a pair of terms is not contained in the training set, there is high possibility that it will become a negative example in the learning process, and will likely be recognized as a non-taxonomic relation. The key assumption behind the design of this approach is not always true as no available dataset can possibly contain all taxonomic relations.

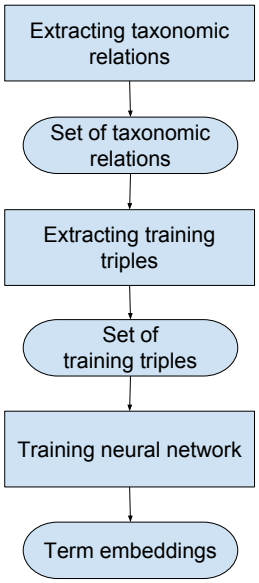


Figure 1: Proposed approach for learning term embeddings.

3 Methodology

In this section, we first propose an approach for learning term embeddings based on hypernym, hyponym and the contextual information between them. We then discuss a supervised method for identifying taxonomic relations based on the term embeddings.

3.1 Learning term embeddings

As shown in Figure 1, there are three steps for learning term embeddings: (i) extracting taxonomic relations; (ii) extracting training triples; and (iii) training neural network. First, we extract from WordNet all taxonomic relations as training data. Then, we extract from Wikipedia all sentences which contain at least one pair of terms involved in a taxonomic relation in the training data, and from that we identify the triples of hypernym, hyponym and contextual words between them. Finally, using the extracted triples as input, we propose a *dynamic weighting neural network* to learn term embeddings based on the information of these triples.

3.1.1 Extracting taxonomic relations

This step aims to extract a set of taxonomic relations for training. For this purpose, we use WordNet hierarchies for extracting all (direct and indirect) taxonomic relations between noun terms in WordNet. However, based on our experience, the relations involving with top-level terms such as 'object', 'entity' or 'whole' are usually ambiguous and become noise for the learning purpose. Therefore, we exclude from the training set all relations which involve with those top-level terms. Note that we also exclude from training set all taxonomic relations that are happened in the datasets used for testing in Section 4.1. As a result, the total number of extracted taxonomic relations is 236,058.

3.1.2 Extracting training triples

This step aims to extract the triples of hypernym, hyponym and the contextual words between them. These triples will serve as the inputs to the neural network for training. In this research, we define contextual words as *all words located between the hypernym and hyponym in a sentence*. We use the latest English Wikipedia corpus as the source for extracting such triples.

Using the set of taxonomic relations extracted from the first step as reference, we extract from the Wikipedia corpus all sentences which contain at least two terms involved in a taxonomic relation. Specifically, for each sentence, we use the Stanford parser (Manning et al., 2014) to parse it, and check whether there is any pair of terms which are nouns or noun phrases in the sentence having a taxonomic relationship. If yes, we extract the hypernym, hyponym and all words between them from the sen-

tence as a training triple. In total, we have extracted 15,499,173 training triples from Wikipedia.

Here, we apply the Stanford parser rather than matching the terms directly in the sentence in order to avoid term ambiguity as a term can serve for different grammatical functions such as noun or verb. For example, consider the following sentence:

- *Many supporters book tickets for the premiere of his new publication.*

The triple (*'publication', 'book', 'tickets for the premiere of his new'*) may be incorrectly added to the training set due to the occurrence of the taxonomic pair (*'publication'*, *'book'*), even though the meaning of *'book'* in this sentence is not about the *'publication'*.

3.1.3 Training neural network

Contextual information is an important indicator for detecting taxonomic relations. For example, in the following two sentences:

- *Dog is a type of animal which you can have as a pet.*
- *Animal such as dog is more sensitive to sound than human.*

The occurrence of contextual words *'is a type of'* and *'such as'* can be used to identify the taxonomic relation between *'dog'* and *'animal'* in the sentences. Many works in the literature (Kozareva and Hovy, 2010; Navigli et al., 2011; Wentao et al., 2012) attempted to manually find these contextual patterns, or automatically learn them. However, due to the wide range of complex linguistic structures, it is difficult to discover all possible contextual patterns between hypernyms and hyponyms in order to detect taxonomic relations effectively.

In this paper, instead of explicitly discovering the contextual patterns of taxonomic relations, we propose a *dynamic weighting neural network* to encode this information, together with the hypernym and hyponym, for learning term embeddings. Specifically, the target of the neural network is to predict the hypernym term from the given hyponym term and contextual words. The architecture of the proposed neural network is shown in Figure 2, which consists of three layers: input layer, hidden layer and output layer.

In our setting, the vocabulary size is V, and the hidden layer size is N. The nodes on adjacent layers are fully connected. Given a term/word t in the vocabulary, the input vector of t is encoded as a one-hot V-dimensional vector x_t, i.e. x_t consists of 0s in all elements except the element used to uniquely identify t which is set as 1. The weights between the input layer and output layer are represented by a $V \times N$ matrix W. Each row of W is a N-dimensional vector representation v_t of the associated word/term t of the input layer.

Given a hyponym term $hypo$ and k context words c_1, c_2, .., c_k in the training triple, the output of hidden layer h is calculated as:

$$\begin{aligned} h &= W^\top \cdot \frac{1}{2k}(k \times x_{hypo} + x_{c_1} + x_{c_2} + ... + x_{c_k}) \\ &= \frac{1}{2k}(k \times v_{hypo} + v_{c_1} + v_{c_2} + ... + v_{c_k}) \end{aligned} \tag{1}$$

where v_t is the vector representation of the input word/term t.

The weight of h in Equation (1) is calculated as the average of the vector representation of hyponym term and contextual words. Therefore, this weight is not based on a fixed number of inputs. Instead, it is dynamically updated based on the number of contextual words k in the current training triple, and the hyponym term. This model is called *dynamic weighting neural network* to reflect its dynamic nature. Note that to calculate h, we also multiply the vector representation of hyponym by k to reduce the bias problem of high number of contextual words, so that the weight of the input vector of hyponym is balanced with the total weight of contextual words.

From the hidden layer to the output layer, there is another weight $N \times V$ for the output matrix W'. Each column of W' is a N-dimensional vector v'_t representing the output vector of t. Using these weights, we can compute an output score u_t for each term/word t in the vocabulary:

$$u_t = {v'_t}^\top \cdot h \tag{2}$$

where v'_t is the output vector of t.

We then use soft-max, a log-linear classification model, to obtain the posterior distribution of hypernym terms as follows:

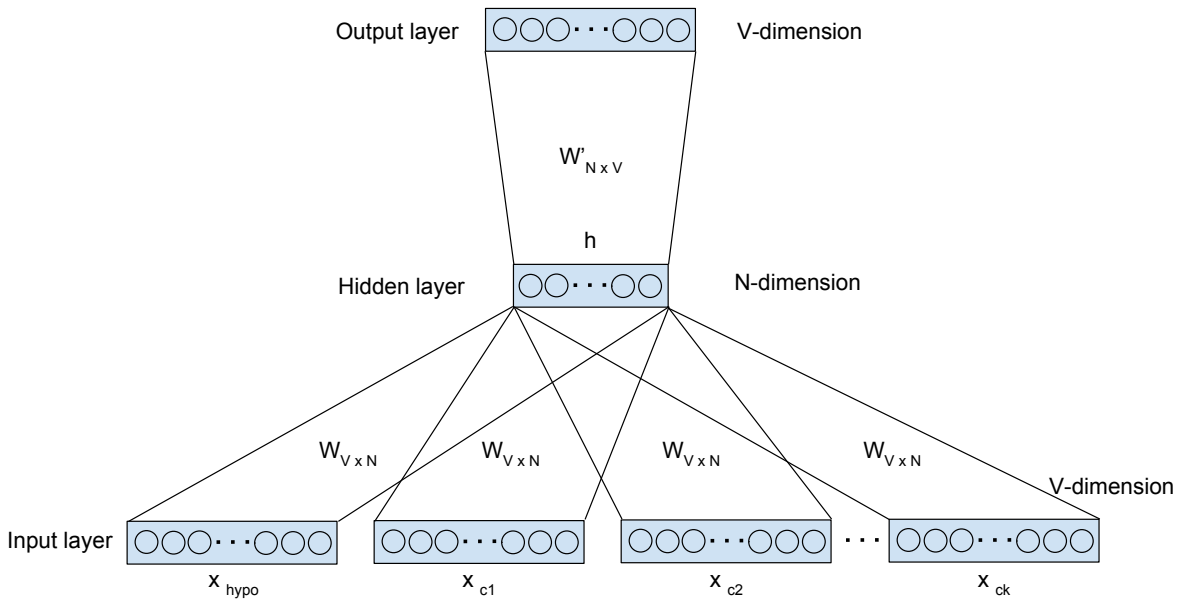


Figure 2: The architecture of the proposed dynamic weighting neural network model.

$$
\begin{aligned}
&p(hype|hypo, c_1, c_2, .., c_k) \\
&= \frac{e^{u_{hype}}}{\sum_{i=1}^{V} e^{u_i}} \\
&= \frac{e^{v'^{\top}_{hype} \cdot \frac{1}{2k}(k \times v_{hypo} + \sum_{j=1}^{k} v_{c_j})}}{\sum_{i=1}^{V} e^{v'^{\top}_{i} \cdot \frac{1}{2k}(k \times v_{hypo} + \sum_{j=1}^{k} v_{c_j})}}
\end{aligned} \tag{3}
$$

The objective function is then defined as:

$$
O = \frac{1}{T} \sum_{t=1}^{T} log(p(hype_t|hypo_t, c_{1t}, c_{2t}, .., c_{kt})) \tag{4}
$$

where T is the number of training triples; $hype_t$, $hypo_t$ and c_{it} are hypernym term, hyponym term and contextual words respectively in the training triple t.

After maximizing the log-likelihood objective function in Equation (4) over the entire training set using stochastic gradient descent, the term embeddings are learned accordingly.

3.2 Supervised taxonomic relation identification

To decide whether a term x is a hypernym of term y, we build a classifier that uses embedding vectors as features for taxonomic relation identification. Specifically, we use Support Vector Machine (SVM) (Cortes and Vapnik, 1995) for this purpose. Given an ordered pair (x, y), the input feature is the concatenation of embedding vectors (v_x,v_y) of x and y. In addition, our term embedding learning approach has the property that the embedding of hypernym is encoded based on not only the information of hyponym but also the information of contextual words. Therefore, we add one more feature to the input of SVM, i.e. the offset vector $(v_x - v_y)$, to contain the information of all contextual words between x and y. In summary, the feature vector is a $3d$ dimensional vector $\langle v_x, v_y, v_x - v_y \rangle$, where d is the dimension of term embeddings. As will be shown later in the experimental results, the offset vector plays an important role in the task of taxonomic relation identification of our approach.

4 Experiments

We conduct experiments to evaluate the performance of our term embedding learning approach on the general domain areas as well as the specific domain areas. In performance evaluation, we compare our approach with two other state-of-the-art supervised term embedding learning methods in Yu et al. (2015) and the Word2Vec model (Mikolov et al., 2013a).

4.1 Datasets

There are five datasets used in the experiments. Two datasets, namely BLESS and ENTAILMENT, are general domain datasets. The other three datasets, namely Animal, Plant and Vehicle, are specific domain datasets.

- BLESS (Baroni and Lenci, 2011) dataset: It covers 200 distinct, unambiguous concepts (terms); each of which is involved with other terms, called *relata*, in some relations. We extract from BLESS 14,547 pairs of terms for the following four types of relations: taxonomic relation, meronymy relation (a.k.a. part-of relation), coordinate relation (i.e. two terms having the same hypernym), and random relation. From these pairs, we set taxonomic relations as positive examples, while other relations form the negative examples.
- ENTAILMENT dataset (Baroni et al., 2012): It consists of 2,770 pairs of terms, with equal number of positive and negative examples of taxonomic relations. Altogether, there are 1,376 unique hyponyms and 1,016 unique hypernyms.
- Animal, Plant and Vehicle datasets (Velardi et al., 2013): They are taxonomies constructed based on the dictionaries and data crawled from the Web for the corresponding domains. The positive examples are created by extracting all possible (direct and indirect) taxonomic relations from the taxonomies. The negative examples are generated by randomly pairing two terms which are not involved in any taxonomic relation.

The number of terms, positive examples and negative examples extracted from the five datasets are summarized in Table 1.

Dataset	# terms	# positive	# negative
BLESS	5229	1337	13210
ENTAILMENT	2392	1385	1385
Animal	659	4164	8471
Plant	520	2266	4520
Vehicle	117	283	586

Table 1: Datasets used in the experiments.

4.2 Comparison models

In the experiments, we use the following supervised models for comparison:

- SVM+Our: This model uses SVM and the term embeddings obtained by our learning approach. The input is a 3d-dimensional vector $\langle v_x, v_y, v_x - v_y \rangle$, where d is the dimension of term embeddings, x and y are two terms used to check whether x is a hypernym of y or not, and v_x, v_y are the term embeddings of x and y respectively.
- SVM+Word2Vec: This model uses SVM and the term embeddings obtained by applying the Skip-gram model (Mikolov et al., 2013a) on the entire English Wikipedia corpus. The input is also a 3d-dimensional vector as in the SVM+Our model. Note that the results of the Skip-gram model are word embeddings. So if a term is a multiword term, its embedding is calculated as the average of all words in the term.
- SVM+Yu: This model uses SVM and the term embeddings obtained by using Yu et al.'s method (2015). According to the best setting stated in (Yu et al., 2015), the input is a 2d+1 dimensional vector $\langle O(x), E(y), \|O(x)\text{-}E(y)\|_1 \rangle$, where $O(x)$, $E(y)$ and $\|\text{O(x)-E(y)}\|_1$ are hyponym embedding of x, hypernym embedding of y and 1-norm distance of the vector ($O(x)$-$E(y)$) respectively.

Parameter settings. The SVM in the three models is trained using a RBF kernel with λ= 0.03125 and penalty term C = 8.0. For term embedding learning, the vector's dimension is set to 100. The tuning of the dimension will be discussed in Section 4.6.

4.3 Performance on general domain datasets

For the general domain datasets, we have conducted two experiments to evaluate the performance of our proposed approach.

Experiment 1. For the BLESS dataset, we hold out one concept for testing and train on the remaining 199 concepts. The hold-out concept and its relatum constitute the testing set, while the remaining 199 concepts and their relatum constitute the training set. To further separate the training and testing sets, we exclude from the training set any pair

of terms that has one term appearing in the testing set. We report the average accuracy across all concepts. For the ENTAILMENT dataset, we use the same evaluation method: hold out one hypernym for testing and train on the remaining hypernyms, and we also report the average accuracy across all hypernyms. Furthermore, to evaluate the effect of the offset vector to taxonomic relation identification, we deploy a setting that removes the offset vector in the feature vectors of SVM. Specifically, for SVM+Our and SVM+Word2Vec, the input vector is changed from $\langle v_x, v_y, v_x - v_y \rangle$ to $\langle v_x, v_y \rangle$. We use the subscript $short$ to denote this setting.

Model	Dataset	Accuracy
SVM+Yu	BLESS	90.4%
SVM+Word2Vec$_{short}$	BLESS	83.8%
SVM+Word2Vec	BLESS	84.0%
SVM+Our$_{short}$	BLESS	91.1%
SVM+Our	BLESS	**93.6%**
SVM+Yu	ENTAIL	87.5%
SVM+Word2Vec$_{short}$	ENTAIL	82.8%
SVM+Word2Vec	ENTAIL	83.3%
SVM+Our$_{short}$	ENTAIL	88.2%
SVM+Our	ENTAIL	**91.7%**

Table 2: Performance results for the BLESS and ENTAILMENT datasets.

Table 2 shows the performance of the three supervised models in Experiment 1. Our approach achieves significantly better performance than Yu's method and Word2Vec method in terms of accuracy (t-test, p-value < 0.05) for both BLESS and ENTAILMENT datasets. Specifically, our approach improves the average accuracy by 4% compared to Yu's method, and by 9% compared to the Word2Vec method. The Word2Vec embeddings have the worst result because it is based only on co-occurrence based similarity, which is not effective for the classifier to accurately recognize all the taxonomic relations. Our approach performs better than Yu's method and it shows that our approach can learn embeddings more effectively. Our approach encodes not only hypernym and hyponym terms but also the contextual information between them, while Yu's method ignores the contextual information for taxonomic relation identification.

Moreover, from the experimental results of SVM+Our and SVM+Our$_{short}$, we can observe that the offset vector between hypernym and hyponym, which captures the contextual information, plays an important role in our approach as it helps to improve the performance in both datasets. However, the offset feature is not so important for the Word2Vec model. The reason is that the Word2Vec model is targeted for the analogy task rather than taxonomic relation identification.

Experiment 2. This experiment aims to evaluate the generalization capability of our extracted term embeddings. In the experiment, we train the classifier on the BLESS dataset, test it on the ENTAILMENT dataset and vice versa. Similarly, we exclude from the training set any pair of terms that has one term appearing in the testing set. The experimental results in Table 3 show that our term embedding learning approach performs better than other methods in accuracy. It also shows that the taxonomic properties identified by our term embedding learning approach have great generalization capability (i.e. less dependent on the training set), and can be used generically for representing taxonomic relations.

Model	Training	Testing	Accuracy
SVM+Yu	BLESS	ENTAIL	83.7%
SVM+Word2Vec$_{short}$	BLESS	ENTAIL	76.5%
SVM+Word2Vec	BLESS	ENTAIL	77.1%
SVM+Our$_{short}$	BLESS	ENTAIL	85.8%
SVM+Our	BLESS	ENTAIL	**89.4%**
SVM+Yu	ENTAIL	BLESS	87.1%
SVM+Word2Vec$_{short}$	ENTAIL	BLESS	78.0%
SVM+Word2Vec	ENTAIL	BLESS	78.9%
SVM+Our$_{short}$	ENTAIL	BLESS	87.1%
SVM+Our	ENTAIL	BLESS	**90.6%**

Table 3: Performance results for the general domain datasets when using one domain for training and another domain for testing.

4.4 Performance on specific domain datasets

Similarly, for the specific domain datasets, we have conducted two experiments to evaluate the performance of our proposed approach.

Experiment 3. For each of the Animal, Plant and Vehicle datasets, we also hold out one term for testing and train on the remaining terms. The positive and negative examples which contain the hold-out term constitute the testing set, while other positive and negative examples constitute the training

set. We also exclude from the training set any pair of terms that has one term appearing in the testing set. The experimental results are given in Table 4. We can observe that not only for general domain datasets but also for specific domain datasets, our term embedding learning approach has achieved significantly better performance than Yu's method and the Word2Vec method in terms of accuracy (t-test, p-value < 0.05). Specifically, our approach improves the average accuracy by 22% compared to Yu's method, and by 9% compared to the Word2Vec method.

Model	Dataset	Accuracy
SVM+Yu	Animal	67.8%
SVM+Word2Vec	Animal	80.2%
SVM+Our	Animal	**89.3%**
SVM+Yu	Plant	65.7%
SVM+Word2Vec	Plant	81.5%
SVM+Our	Plant	**92.1%**
SVM+Yu	Vehicle	70.5%
SVM+Word2Vec	Vehicle	82.1%
SVM+Our	Vehicle	**89.6%**

Table 4: Performance results for the Animal, Plant and Vehicle datasets.

Another interesting point to observe is that the accuracy of Yu's method drops significantly in specific domain datasets (as shown in Table 4) when compared to the general domain datasets (as shown in Table 2). One possible explanation is the accuracy of Yu's method depends on the training data. As Yu's method learns the embeddings using pre-extracted taxonomic relations from Probase, and if a relation does not exist in Probase, there is high possibility that it becomes a negative example and be recognized as a non-taxonomic relation by the classifier. Therefore, the training data extracted from Probase plays an important role in Yu's method. For general domain datasets (BLESS and ENTAILMENT), there are about 75%-85% of taxonomic relations in these datasets found in Probase, while there are only about 25%-45% of relations in the specific domains (i.e. Animal, Plant and Vehicle) found in Probase. Therefore, Yu's method achieves better performance in general domain datasets than the specific ones. Our approach, in contrast, less depends on the training relations. Therefore, it can achieve high accuracy in both the general and specific domain datasets.

Experiment 4. Similar to experiment 2, this experiment aims to evaluate the generalization capability of our term embeddings. In this experiment, for each of the Animal, Plant and Vehicle domains, we train the classifier using the positive and negative examples in each domain and test the classifier in other domains. The experimental results in Table 5 show that our approach achieves the best performance compared to other state-of-the-art methods for all the datasets. As also shown in Table 3, our approach has achieved high accuracy for both general and specific domain datasets, while in Yu's method, there is a huge difference in accuracy between these domain datasets.

Model	Training	Testing	Accuracy
SVM+Yu	Animal	Plant	65.5%
SVM+Word2Vec	Animal	Plant	82.4%
SVM+Our	Animal	Plant	**91.9%**
SVM+Yu	Animal	Vehicle	66.2%
SVM+Word2Vec	Animal	Vehicle	81.3%
SVM+Our	Animal	Vehicle	**89.5%**
SVM+Yu	Plant	Animal	68.4%
SVM+Word2Vec	Plant	Animal	81.8%
SVM+Our	Plant	Animal	**91.5%**
SVM+Yu	Plant	Vehicle	65.2%
SVM+Word2Vec	Plant	Vehicle	81.0%
SVM+Our	Plant	Vehicle	**88.5%**
SVM+Yu	Vehicle	Animal	70.9%
SVM+Word2Vec	Vehicle	Animal	79.7%
SVM+Our	Vehicle	Animal	**87.6%**
SVM+Yu	Vehicle	Plant	66.2%
SVM+Word2Vec	Vehicle	Plant	78.7%
SVM+Our	Vehicle	Plant	**87.7%**

Table 5: Performance results for the specific domain datasets when using one domain for training and another domain for testing.

4.5 Empirical comparison with WordNet

By error analysis, we found that our results may complement WordNet. For example, in the Animal domain, our approach identifies *'wild sheep'* as a hyponym of *'sheep'*, but in WordNet, they are siblings. However, many references [1, 2] consider *'wild sheep'* as a species of *'sheep'*. Another such example is shown in the Plant domain, where our ap-

[1] http://en.wikipedia.org/wiki/Ovis

[2] http://www.bjornefabrikken.no/side/norwegian-sheep/

proach recognizes *'lily'* as a hyponym of *'flowering plant'*, but WordNet places them in different subtrees incorrectly [3]. Therefore, our results may help restructure and even extend WordNet.

Note that these taxonomic relations are not in our training set. They are also not recognized by the term embeddings obtained from the Word2Vec method and Yu et al.'s method. It again shows that our term embedding learning approach has the capability to identify taxonomic relations which are not even defined in dictionary or training data.

4.6 Tuning vector dimensions

We also conduct experiments to learn term embeddings from the general domain datasets with different dimensions (i.e. 50, 100, 150 and 300) using our proposed approach. We then use these embeddings to evaluate the performance of taxonomic relation identification based on training time and accuracy, and show the results in Table 6. The experiments are carried out on a PC with Intel(R) Xeon(R) CPU at 3.7GHz and 16GB RAM.

Dimension	Dataset	Training time	Accuracy
50	BLESS	1825s	87.7%
100	BLESS	2991s	89.4%
150	BLESS	4025s	89.9%
300	BLESS	7113s	90.0%
50	ENTAIL	1825s	88.5%
100	ENTAIL	2991s	90.6%
150	ENTAIL	4025s	90.9%
300	ENTAIL	7113s	90.9%

Table 6: Performance results based on training time and accuracy of the SVM+Our model using different vector dimensions.

In general, when increasing the vector dimension, the accuracy of our term embedding learning approach will be increased gradually. More specifically, the accuracy improves slightly when the dimension is increased from 50 to 150. But after that, increasing the dimension has very little effect on the accuracy. We observe that the vector dimension for learning term embeddings can be set between 100 to 150 to achieve the best performance, based on the trade-off between accuracy and training time.

[3] https://en.wikipedia.org/wiki/Lilium

5 Conclusion

In this paper, we proposed a novel approach to learn term embeddings using dynamic weighting neural network. This model encodes not only the hypernym and hyponym terms, but also the contextual information between them. Therefore, the extracted term embeddings have good generalization capability to identify unseen taxonomic relations which are not even defined in dictionary and training data. The experimental results show that our approach significantly outperforms other state-of-the-art methods in terms of accuracy in identifying taxonomic relation identification.

References

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32.

Yoshua Bengio, Rjean Ducharme, and Pascal Vincent. 2001. A Neural Probabilistic Language Model. *Proceedings of the NIPS conference*, pages 932–938.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems*, pages 17–24.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Samah Fodeh, Bill Punch, and Pang N. Tan. 2011. On ontology-driven document clustering using core semantic features. *Knowledge and information systems*, 28(2):395–421.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. *Proceedings of the 52nd Annual Meeting of the ACL*, pages 1199–1209.

Sanda M. Harabagiu, Steven J. Maiorano, and Marius A. Pasca. 2003. Open-domain textual question an-

swering techniques. *Natural Language Engineering*, 9(3):231–267.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. *Proceedings of the 14th Conference on Computational Linguistics*, pages 539–545.

Zornitsa Kozareva and Eduard Hovy. 2010. A Semi-supervised Method to Learn and Construct Taxonomies Using the Web. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118.

German Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Dawn J. Lawrie and W. Bruce Croft. 2003. Generating hierarchical summaries for web searches. *Proceedings of the 26th ACM SIGIR conference*, pages 457–463.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2014. Do supervised distributional methods really learn lexical inference relations. *Proceedings of the NAACL conference*, pages 1390–1397.

Baichuan Li, Jing Liu, Chin Y. Lin, Irwin King, and Michael R. Lyu. 2013. A Hierarchical Entity-based Approach to Structuralize User Generated Content in Social Media: A Case of Yahoo! Answers. *Proceedings of the EMNLP conference*, pages 1521–1532.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. *Proceedings of the 52nd Annual Meeting of the ACL*, pages 55–60.

Cynthia Matuszek, John Cabral, Michael J. Witbrock, and John DeOliveira. 2006. An introduction to the syntax and content of cyc. *Proceedings of the AAAI Spring Symposium*, pages 44–49.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George A. Miller. 1995. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A Graph-based Algorithm for Inducing Lexical Taxonomies from Scratch. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1872–1877.

Yves Petinot, Kathleen McKeown, and Kapil Thadani. 2011. A hierarchical model of web summaries. *Proceedings of the 49th Annual Meeting of the ACL*, pages 670–675.

Aurora Pons-Porrata, Rafael Berlanga-Llavori, and Jose Ruiz-Shulcloper. 2007. Topic discovery based on text mining techniques. *Information processing & management*, 43(3):752–768.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. *Proceedings of the COLING conference*, pages 1025–1036.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. *Proceedings of the 51st Annual Meeting of the ACL*, pages 932–937.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the EMNLP conference*, pages 1631–1642.

Liling Tan, Rohit Gupta, and Josef van Genabith. 2015. Usaar-wlv: Hypernym generation with deep neural nets. *Proceedings of the SemEval*, pages 932–937.

Luu A. Tuan, Jung J. Kim, and See K. Ng. 2014. Taxonomy Construction using Syntactic Contextual Evidence. *Proceedings of the EMNLP conference*, pages 810–819.

Luu A. Tuan, Jung J. Kim, and See K. Ng. 2015. Incorporating Trustiness and Collective Synonym/Contrastive Evidence into Taxonomy Construction. *Proceedings of the EMNLP conference*, pages 1013–1022.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Chi Wang, Marina Danilevsky, Nihit Desai, Yinan Zhang, Phuong Nguyen, Thrivikrama Taula, and Jiawei Han. 2013. A phrase mining framework for recursive construction of a topical hierarchy. *Proceedings of the 19th ACM SIGKDD conference*, pages 437–445.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. *Proceedings of the COLING conference*, pages 2249–2259.

Wu Wentao, Li Hongsong, Wang Haixun, and Kenny. Q. Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. *Proceedings of the ACM SIGMOD conference*, pages 481–492.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, Kai Wang, and Tat-Seng Chua. 2011. Domain-assisted product as-

pect hierarchy generation: towards hierarchical organization of unstructured consumer reviews. *Proceedings of the EMNLP conference*, pages 140–150.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1390–1397.

Xingwei Zhu, Zhao Y. Ming, and Tat-Seng Chua. 2013. Topic hierarchy construction for the organization of multi-source user generated contents. *Proceedings of the 36th ACM SIGIR conference*, pages 233–242.

Will Y Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. *Proceedings of the EMNLP conference*, pages 1393–1398.

Relation Schema Induction using Tensor Factorization with Side Information

Madhav Nimishakavi
Indian Institute of Science
Bangalore, India
madhav@csa.iisc.ernet.in

Uday Singh Saini
Indian Institute of Science
Bangalore, India
uday.s.saini@gmail.com

Partha Talukdar
Indian Institute of Science
Bangalore, India
ppt@cds.iisc.ac.in

Abstract

Given a set of documents from a specific domain (e.g., medical research journals), how do we automatically build a Knowledge Graph (KG) for that domain? Automatic identification of relations and their schemas, i.e., type signature of arguments of relations (e.g., *undergo(Patient, Surgery)*), is an important first step towards this goal. We refer to this problem as *Relation Schema Induction (RSI)*. In this paper, we propose Schema Induction using Coupled Tensor Factorization (SICTF), a novel tensor factorization method for relation schema induction. SICTF factorizes Open Information Extraction (OpenIE) triples extracted from a domain corpus along with additional side information in a principled way to induce relation schemas. To the best of our knowledge, this is the first application of tensor factorization for the RSI problem. Through extensive experiments on multiple real-world datasets, we find that SICTF is not only more accurate than state-of-the-art baselines, but also significantly faster (about 14x faster).

1 Introduction

Over the last few years, several techniques to build Knowledge Graphs (KGs) from large unstructured text corpus have been proposed, examples include NELL (Mitchell et al., 2015) and Google Knowledge Vault (Dong et al., 2014). Such KGs consist of millions of entities (e.g., *Oslo, Norway*, etc.), their types (e.g., *isA(Oslo, City)*, *isA(Norway, Country)*), and relationships among them (e.g., *cityLocatedInCountry(Oslo, Norway)*). These KG construction techniques are called ontology-guided as they require as input list of relations, their schemas (i.e., their type signatures, e.g., *cityLocatedInCountry(City, Country)*), and seed instances of each such relation. Listing of such relations and their schemas are usually prepared by human domain experts.

The reliance on domain expertise poses significant challenges when such ontology-guided KG construction techniques are applied to domains where domain experts are either not available or are too expensive to employ. Even when such a domain expert may be available for a limited time, she may be able to provide only a partial listing of relations and their schemas relevant to that particular domain. Moreover, this expert-mediated model is not scalable when new data in the domain becomes available, bringing with it potential new relations of interest. In order to overcome these challenges, we need automatic techniques which can discover relations and their schemas from unstructured text data itself, without requiring extensive human input. We refer to this problem as *Relation Schema Induction (RSI)*.

In contrast to ontology-guided KG construction techniques mentioned above, Open Information Extraction (OpenIE) techniques (Etzioni et al., 2011) aim to extract surface-level triples from unstructured text. Such OpenIE triples may provide a suitable starting point for the RSI problem. In fact, KB-LDA,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 414–423,
Austin, Texas, November 1-5, 2016.

	Target task	Interpretable latent factors?	Can induce relation schema?	Can use NP side info?	Can use relation side info?
Typed RESCAL (Chang et al., 2014a)	Embedding	No	No	Yes	No
Universal Schema (Singh et al., 2015)	Link Prediction	No	No	No	No
KB-LDA (Movshovitz-Attias and Cohen, 2015)	Ontology Induction	Yes	Yes	Yes	No
SICTF (this paper)	Schema Induction	Yes	Yes	Yes	Yes

Table 1: Comparison among SICTF (this paper) and other related methods. KB-LDA is the most related prior method which is extensively compared against SICTF in Section 4

a topic modeling-based method for inducing an ontology from SVO (Subject-Verb-Object) triples was recently proposed in (Movshovitz-Attias and Cohen, 2015). We note that ontology induction (Velardi et al., 2013) is a more general problem than RSI, as we are primarily interested in identifying categories and relations from a domain corpus, and not necessarily any hierarchy over them. Nonetheless, KB-LDA maybe used for the RSI problem and we use it as a representative of the state-of-the-art of this area.

Instead of a topic modeling approach, we take a tensor factorization-based approach for RSI in this paper. Tensors are a higher order generalization of matrices and they provide a natural way to represent OpenIE triples. Applying tensor factorization methods over OpenIE triples to identify relation schemas is a natural approach, but one that has not been explored so far. Also, a tensor factorization-based approach presents a flexible and principled way to incorporate various types of side information. Moreover, as we shall see in Section 4, compared to state-of-the-art baselines such as KB-LDA, tensor factorization-based approach results in better and faster solution for the RSI problem. In this paper, we make the following contributions:

- We present Schema Induction using Coupled Tensor Factorization (SICTF), a novel and principled tensor factorization method which jointly factorizes a tensor constructed out of OpenIE triples extracted from a domain corpus, along with various types of additional side information for relation schema induction.

- We compare SICTF against state-of-the-art baseline on various real-world datasets from diverse domains. We observe that SICTF is not only significantly more accurate than such baselines, but also much faster. For example, SICTF achieves 14x speedup over KB-LDA (Movshovitz-Attias and Cohen, 2015).

- We have made the data and code available [1].

2 Related Work

Schema Induction: Properties of SICTF and other related methods are summarized in Table 1[2]. A method for inducing (binary) relations and the categories they connect was proposed by (Mohamed et al., 2011). However, in that work, categories and their instances were known a-priori. In contrast, in case of SICTF, both categories and relations are to be induced. A method for event schema induction, the task of learning high-level representations of complex events and their entity roles from unlabeled text, was proposed in (Chambers, 2013). This gives the schemas of slots per event, but our goal is to find schemas of relations. (Chen et al., 2013) and (Chen et al., 2015) deal with the problem of finding semantic slots for unsupervised spoken language understanding, but we are interested in finding schemas of relations relevant for a given domain. Methods for link prediction in the Universal Schema setting using matrix and a combination of matrix and tensor factorization are proposed in (Riedel et al., 2013) and (Singh et al., 2015), respectively. Instead of link prediction where relation schemas are assumed to be given, SICTF focuses on discovering such relation schemas. Moreover, in contrast to such

[1] https://github.com/malllabiisc/sictf

[2] Please note that not all methods mentioned in the table are directly comparable with SICTF, the table only illustrates the differences. KB-LDA is the only method which is directly comparable.

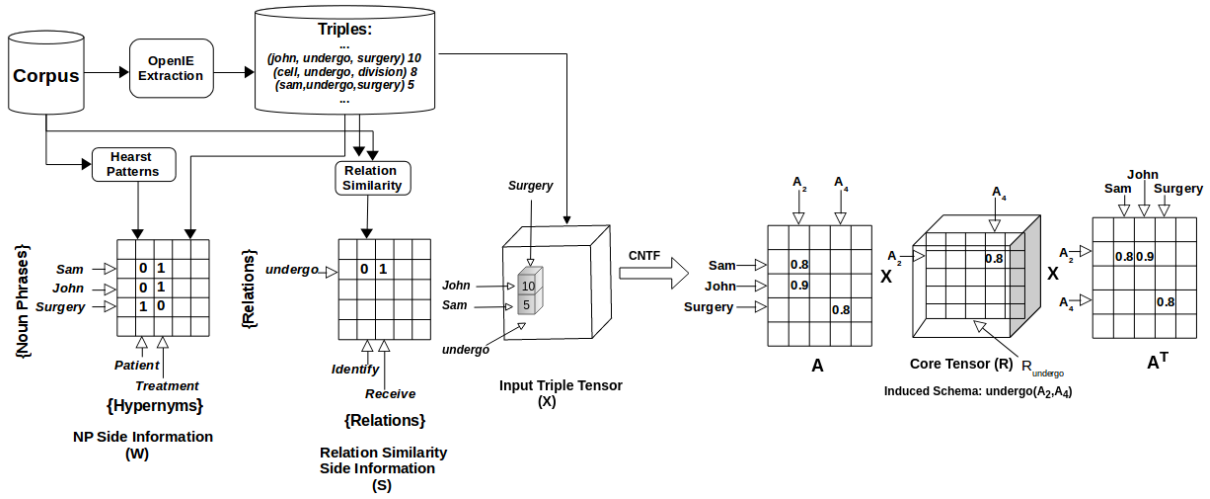


Figure 1: Relation Schema Induction (RSI) by SICTF, the proposed method. First, a tensor (X) is constructed to represent OpenIE triples extracted from a domain corpus. Noun phrase side information in the form of (noun phrase, hypernym), and relation-relation similarity side information are separately calculated and stored in two separate matrices (W and S, respectively). SICTF then performs coupled factorization of the tensor and the two side information matrices to identify relation schemas which are stored in the core tensor (R) in the output. Please see Section 3 for details.

methods which assume access to existing KGs, the setting in this paper is unsupervised.

Tensor Factorization: Due to their flexibility of representation and effectiveness, tensor factorization methods have seen increased application in Knowledge Graph (KG) related problems over the last few years. Methods for decomposing ontological KGs such as YAGO (Suchanek et al., 2007) were proposed in (Nickel et al., 2012; Chang et al., 2014b; Chang et al., 2014a). In these cases, relation schemas are known in advance, while we are interested in inducing such relation schemas from unstructured text. A PARAFAC (Harshman, 1970) based method for jointly factorizing a matrix and tensor for data fusion was proposed in (Acar et al., 2013). In such cases, the matrix is used to provide auxiliary information (Narita et al., 2012; Erdos and Miettinen, 2013). Similar PARAFAC-based ideas are explored in Rubik (Wang et al., 2015) to factorize structured electronic health records. In contrast to such structured data sources, SICTF aims at inducing relation schemas from unstructured text data. Propstore, a tensor-based model for distributional semantics, a problem different from RSI, was presented in (Goyal et al., 2013). Even though coupled factorization of tensor and matrices constructed out of unstructured text corpus provide a natural and plausible approach for the RSI problem, they have not yet been explored – we fill this gap in this paper.

Ontology Induction: Relation Schema Induction can be considered a sub problem of Ontology Induction (Velardi et al., 2013). Instead of building a full-fledged hierarchy over categories and relations as in ontology induction, we are particularly interested in finding relations and their schemas from unstructured text corpus. We consider KB-LDA[3] (Movshovitz-Attias and Cohen, 2015), a topic-modeling based approach for ontology induction, as a representative of this area. Among all prior work, KB-LDA is most related to SICTF. While both KB-LDA and SICTF make use of noun phrase side information, SICTF is also able to exploit relational side information in a principled manner. In Section 4, through experiments on multiple real-world datasets, we observe that SICTF is not only more accurate than KB-LDA but also significantly faster with a speedup of 14x.

A method for canonicalizing noun and relation phrases in OpenIE triples was recently proposed in (Galárraga et al., 2014). The main focus of this approach is to cluster lexical variants of a *single* entity or relation. This is not directly relevant for RSI, as

[3]In this paper, whenever we refer to KB-LDA, we only refer to the part of it that learns relations from unstructured data.

we are interested in grouping *multiple* entities of the same type into one cluster, and use that to induce relation schema.

3 Our Approach: Schema Induction using Coupled Tensor Factorization (SICTF)

3.1 Overview

SICTF poses the relation schema induction problem as a coupled factorization of a tensor along with matrices containing relevant side information. Overall architecture of the SICTF system is presented in Figure 1. First, a tensor $X \in \mathbb{R}_+^{n \times n \times m}$ is constructed to store OpenIE triples and their scores extracted from the text corpus[4]. Here, n and m represent the number of NPs and relation phrases, respectively. Following (Movshovitz-Attias and Cohen, 2015), SICTF makes use of noun phrase (NP) side information in the form of (noun phrase, hypernym). Additionally, SICTF also exploits relation-relation similarity side information. These two side information are stored in matrices $W \in \{0,1\}^{n \times h}$ and $S \in \{0,1\}^{m \times m}$, where h is the number of hypernyms extracted from the corpus. SICTF then performs collective non-negative factorization over X, W, and S to output matrix $A \in \mathbb{R}_+^{n \times c}$ and the core tensor $R \in \mathbb{R}_+^{c \times c \times m}$. Each row in A corresponds to an NP, while each column corresponds to an induced category (latent factor). For brevity, we shall refer to the induced category corresponding to the q^{th} column of A as A_q. Each entry A_{pq} in the output matrix provides a membership score for NP p in induced category A_q. Please note that each induced category is represented using the NPs participating in it, with the NPs ranked by their membership scores in the induced category. In Figure 1, $A_2 = [(John, 0.9), (Sam, 0.8), \ldots]$ is an induced category.

Each slice of the core tensor R is a matrix which corresponds to a specific relation, e.g., the matrix $R_{undergo}$ highlighted in Figure 1 corresponds to the relation *undergo*. Each cell in this matrix corresponds to an induced schema connecting two induced categories (two columns of the A matrix), with the cell value representing model's score of the induced schema. For example, in Figure 1, *undergo*(A_2, A_4) is an induced relation schema with score 0.8 involving relation *undergo* and induced categories A_2 and A_4.

In Section 3.2, we present details of the side information used by SICTF, and then in Section 3.3 present details of the optimization problem solved by SICTF.

[4] $\mathbb{R}_+$ is the set of non-negative reals.

MEDLINE
(hypertension, disease), (hypertension, state), (hypertension, disorder) , (neutrophil, blood element), (neutrophil, effector cell), (neutrophil, cell type)
StackOverflow
(image, resource), (image, content), (image, file), (perl, language), (perl, script), (perl, programs)

Table 2: Noun Phrase (NP) side information in the form of (Noun Phrase, Hypernym) pairs extracted using Hearst patterns from two different datasets. Please see Section 3.2 for details.

MEDLINE	StackOverflow
(evaluate, analyze), (evaluate, examine), (indicate, confirm), (indicate, suggest)	(provides, confirms), (provides, offers), (allows, lets), (allows, enables)

Table 3: Examples of relation similarity side information in the form of automatically identified similar relation pairs. Please see Section 3.2 for details.

3.2 Side Information

- **Noun Phrase Side Information**: Through this type of side information, we would like to capture type information of as many noun phrases (NPs) as possible. We apply Hearst patterns (Hearst, 1992), e.g., *"<Hypernym> such as <NP>"*, over the corpus to extract such *(NP, Hypernym)* pairs. Please note that neither hypernyms nor NPs are pre-specified, and they are all extracted from the data by the patterns. Examples of a few such pairs extracted from two different datasets are shown in Table 2. These extracted tuples are stored in a matrix $W_{n \times h}$ whose rows correspond to NPs and columns correspond to extracted hypernyms. We define,

$$W_{ij} = \begin{cases} 1, & \text{if NP}_\text{i} \text{ belongs to Hypernym}_\text{j} \\ 0, & \text{otherwise} \end{cases}.$$

  Please note that we don't expect W to be a fully specified matrix, i.e., we don't assume that we know all possible hypernyms for a given NP.

- **Relation Side Information**: In addition to the side information involving NPs, we would also

like to take prior knowledge about textual relations into account during factorization. For example, if we know two relations to be similar to one another, then we also expect their induced schemas to be similar as well. Consider the following sentences *"Mary purchased a stuffed animal toy."* and *"Janet bought a toy car for her son."*. From these we can say that both relations *purchase* and *buy* have the schema (Person, Item). Even if one of these relations is more abundant than the other in the corpus, we still want to learn similar schemata for both the relations. As mentioned before, $S \in \mathbb{R}_{+}^{m \times m}$ is the relation similarity matrix, where m is the number of textual relations. We define,

$$S_{ij} = \begin{cases} 1, & \text{if Similarity(Rel}_\text{i}\text{, Rel}_\text{j}\text{)} \geq \gamma \\ 0, & \text{otherwise} \end{cases}$$

where γ is a threshold[5]. For the experiments in this paper, we use cosine similarity over word2vec (Mikolov et al., 2013) vector representations of the relational phrases. Examples of a few similar relation pairs are shown in Table 3.

3.3 SICTF Model Details

SICTF performs coupled non-negative factorization of the input triple tensor $X_{n \times n \times m}$ along with the two side information matrices $W_{n \times h}$ and $S_{m \times m}$ by solving the following optimization problem.

$$\min_{A,V,R} \sum_{k=1}^{m} f(X_k, A, R_k) + f_{np}(W, A, V) + f_{rel}(S, R) \tag{1}$$

where,

$$f(X_k, A, R_k) = \| X_{:,:,k} - A R_{:,:,k} A^T \|_F^2 + \lambda_R \| R_{:,:,k} \|_F^2$$

$$f_{np}(W, A, V) = \lambda_{np} \| W - AV \|_F^2 + \lambda_A \| A \|_F^2 + \lambda_V \| V \|_F^2$$

$$f_{rel}(S, R) = \lambda_{rel} \sum_{i=1}^{m} \sum_{j=1}^{m} S_{ij} \| R_{:,:,i} - R_{:,:,j} \|_F^2$$

$$A_{i,j} \geq 0, V_{j,r} \geq 0, R_{p,q,k} \geq 0 \quad \text{(non negative)}$$
$$\forall\, 1 \leq i \leq n, 1 \leq r \leq h,$$
$$1 \leq j, p, q \leq c, 1 \leq k \leq m$$

In the objective above, the first term $f(X_k, A, R_k)$ minimizes reconstruction error for the k^{th} relation, with additional regularization on the $R_{:,:,k}$ matrix[6]. The second term, $f_{np}(W, A, V)$, factorizes the NP side information matrix $W_{n \times h}$ into two matrices $A_{n \times c}$ and $V_{c \times h}$, where c is the number of induced categories. We also enforce A to be non-negative. Typically, we require $c \ll h$ to get a lower dimensional embedding of each NP (rows of A). Finally, the third term $f_{rel}(S, R)$ enforces the requirement that two similar relations as given by the matrix S should have similar signatures (given by the corresponding R matrix). Additionally, we require V and R to be non-negative, as marked by the (non-negative) constraints. In this objective, λ_R, λ_{np}, λ_A, λ_V, and λ_{rel} are all hyper-parameters.

We derive non-negative multiplicative updates for A, R_k and V following the rules proposed in (Lee and Seung, 2000), which has the following general form:

$$\theta_i = \theta_i \left(\frac{\frac{\partial C(\theta)^-}{\partial \theta_i}}{\frac{\partial C(\theta)^+}{\partial \theta_i}} \right)^{\alpha}$$

Here $C(\theta)$ represents the cost function of the non-negative variables θ and $\frac{\partial C(\theta)^-}{\partial \theta_i}$ and $\frac{\partial C(\theta)^-}{\partial \theta_i}$ are the negative and positive parts of the derivative of $C(\theta)$ (Mørup et al., 2008). (Lee and Seung, 2000) proved that for $\alpha = 1$, the cost function $C(\theta)$ monotonically decreases with the multiplicative updates [7]. $C(\theta)$ for SICTF is given in equation (1). The above procedure will give the following updates:

$$A \leftarrow A * \frac{\sum_k (X_k A R_k^T + X_k^T A R_k) + \lambda_{np} W V^T}{A(\tilde{B} + \lambda_A I + \lambda_{np} V V^T)}$$

$$\tilde{B} = \sum_k (R_k A^T A R_k^T + R_k^T A^T A R_k)$$

$$R_k \leftarrow R_k * \frac{A^T X_k A + 2\, \lambda_{rel} \sum_{j=1}^{m} R_j S_{kj}}{A^T A R_k A^T A + \tilde{D}}$$

$$\tilde{D} = 2\, \lambda_{rel}\, R_k \sum_{j=1}^{m} S_{kj} + \lambda_R R_k$$

$$V \leftarrow V * \frac{\lambda_{np} A^T W}{\lambda_{np} A^T A V + \lambda_V V}$$

[5] For the experiments in this paper, we set $\gamma = 0.7$, a relatively high value, to focus on highly similar relations and thereby justifying the binary S matrix.

[6] For brevity, we also refer to $R_{:,:,k}$ as R_k, and similarly $X_{:,:,k}$ as X_k

[7] We also use $\alpha = 1$.

Dataset	# Docs	# Triples
MEDLINE	50,216	2,499
StackOverflow	5.5m	37,439

Table 4: Datasets used in the experiments.

In the equations above, $*$ is the Hadamard or element-wise product[8]. In all our experiments, we find the iterative updates above to converge in about 10-20 iterations.

4 Experiments

In this section, we evaluate performance of different methods on the Relation Schema Induction (RSI) task. Specifically, we address the following questions.

- Which method is most effective on the RSI task? (Section 4.3.1)
- How important are the additional side information for RSI? (Section 4.3.2)
- What is the importance of non-negativity in RSI with tensor factorization? (Section 4.3.3)

4.1 Experimental Setup

Datasets: We used two datasets for the experiments in this paper, they are summarized in Table 4. For MEDLINE dataset, we used Stanford CoreNLP (Manning et al., 2014) for coreference resolution and Open IE v4.0[9] for triple extraction. Triples with Noun Phrases that have Hypernym information were retained. We obtained the StackOverflow triples directly from the authors of (Movshovitz-Attias and Cohen, 2015), which were also prepared using a very similar process. In both datasets, we use corpus frequency of triples for constructing the tensor.

Side Information: Seven Hearst patterns such as *"<hypernym> such as <NP>", "<NP> or other <hypernym>"* etc., given in (Hearst, 1992) were used to extract NP side information from the MEDLINE documents. NP side information for the StackOverflow dataset was obtained from the authors of (Movshovitz-Attias and Cohen, 2015).

As described in Section 3, word2vec embeddings of the relation phrases were used to extract relation-similarity based side-information. This was done for both datasets. Cosine similarity threshold of $\gamma = 0.7$ was used for the experiments in the paper.

Samples of side information used in the experiments are shown in Table 2 and Table 3. A total of 2067 unique NP-hypernym pairs were extracted from MEDLINE data and 16,639 were from StackOverflow data. 25 unique pairs of relation phrases out of 1172 were found to be similar in MEDLINE data, whereas 280 unique pairs of relation phrases out of approximately 3200 were found similar in StackOverflow data.

Hyperparameters were tuned using grid search and the set which gives minimum reconstruction error for both X and W was chosen. We set $\lambda_{np} = \lambda_{rel} = 100$ for StackOverflow, and $\lambda_{np} = 0.05$ and $\lambda_{rel} = 0.001$ for Medline and we use $c = 50$ for our experiments. Please note that our setting is unsupervised, and hence there is no separate train, dev and test sets.

4.2 Evaluation Protocol

In this section, we shall describe how the induced schemas are presented to human annotators and how final accuracies are calculated. In factorizations produced by SICTF and other ablated versions of SICTF, we first select a few top relations with best reconstruction score. The schemas induced for each selected relation k is represented by the matrix slice R_k of the core tensor obtained after factorization (see Section 3). From each such matrix, we identify the indices (i, j) with highest values. The indices i and j select columns of the matrix A. A few top ranking NPs from the columns A_i and A_j along with the relation k are presented to the human annotator, who then evaluates whether the tuple $\text{Relation}_k(A_i, A_j)$ constitutes a valid schema for relation k. Examples of a few relation schemas induced by SICTF are presented in Table 5. A human annotator would see the first and second columns of this table and then offer judgment as indicated in the third column of the table. All such judgments across all top-reconstructed relations are aggregated to get the final accuracy score. This evaluation protocol was also used in (Movshovitz-Attias and Cohen, 2015) to measure learned relation accuracy.

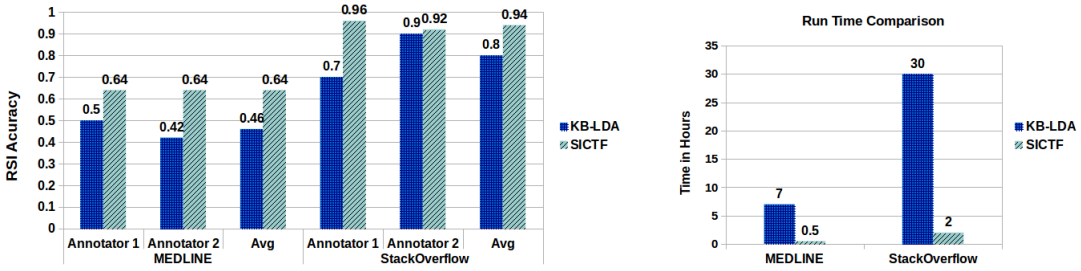
All evaluations were blind, i.e., the annotators were not aware of the method that generated the output they were evaluating. Moreover, the anno-

[8] $(A * B)_{i,j} = A_{i,j} \times B_{i,j}$

[9] Open IE v4.0: http://knowitall.github.io/openie/

Relation Schema	Top 3 NPs in Induced Categories which were presented to annotators	Annotator Judgment
StackOveflow		
clicks(A_0, A_1)	A_0: *users, client, person* A_1: *link, image, item*	valid
refreshes(A_{19}, A_{13})	A_{19}: *browser, window, tab* A_{13}: *page, activity, app*	valid
can_parse(A_{41}, A_{17})	A_{41}: *access, permission, ability* A_{17}: *image file, header file, zip file*	invalid
MEDLINE		
suffer_from(A_{38}, A_{40})	A_{38}: *patient, first patient, anesthetized patient* A_{40}: *viral disease, renal disease, von recklin ghausen's disease*	valid
have_undergo(A_3, A_{37})	A_3: *fifth patient, third patient, sixth patient* A_{37}: *initial liver biopsy, gun biopsy, lymph node biopsy*	valid
have_discontinue(A_{41}, A_{20})	A_{41}: *patient, group, no patient* A_{20}: *endemic area, this area, fiber area*	invalid

Table 5: Examples of relation schemas induced by SICTF from the StackOverflow and MEDLINE datasets. Top NPs from each of the induced categories, along with human judgment of the induced schema are also shown. See Section 4.3.1 for more details.



(a) (b)

Figure 2: (a) Relation Schema Induction (RSI) accuracies of different methods on the two datasets. SICTF, our proposed method, significantly outperforms state-of-the-art method KBLDA. This is the main result of the paper. Results for KB-LDA on StackOveflow are directly taken from the paper. Please see Section 4.3.1 for details. (b) Runtime comparison between KB-LDA and SICTF. We observe that SICTF results in 14x speedup over KB-LDA. Please see Section 4.3.1 (Runtime Comparison) for details.

tators are experts in software domain and has high-school level knowledge in medical domain. Though recall is a desirable statistic to measure, it is very challenging to calculate it in our setting due to the non-availability of relation schema annotated text on large scale.

4.3 Results

4.3.1 Effectiveness of SICTF

Experimental results comparing performance of various methods on the RSI task in the two datasets are presented in Figure 2(a). RSI accuracy is calculated based on the evaluation protocol described in Section 4.2. Performance number of KB-LDA for StackOveflow dataset is taken directly from the (Movshovitz-Attias and Cohen, 2015) paper, we used our implementation of KB-LDA for the MEDLINE dataset. Annotation accuracies from two annotators were averaged to get the final accuracy. From Figure 2(a), we observe that SICTF outperforms KB-LDA on the RSI task. Please note that the inter-annotator agreement for SICTF is 88% and 97% for MEDLINE and StackOverflow datasets respectively. This is the main result of the paper.

In addition to KB-LDA, we also compared SICTF with PARAFAC, a standard tensor factorization method. PARAFAC induced extremely poor and small number of relation schemas, and hence we didn't consider it any further.

Runtime comparison: Runtimes of SICTF and KB-LDA over both datasets are compared in Figure 2(b). From this figure, we find that SICTF is able to achieve a 14x speedup on average over KB-LDA[10]. In other words, SICTF is not only able to

[10]Runtime of KB-LDA over the StackOverflow dataset was obtained from the authors of (Movshovitz-Attias and Cohen, 2015) through personal communication. Our own implementation also resulted in similar runtime over this dataset.

Ablation	MEDLINE			StackOverflow		
	A1	A2	Avg	A1	A2	Avg
SICTF	0.64	0.64	**0.64**	0.96	0.92	**0.94**
SICTF (λ_{rel} = 0)	0.60	0.56	0.58	0.83	0.70	0.77
SICTF (λ_{np} = 0)	0.46	0.40	0.43	0.89	0.90	0.90
SICTF (λ_{rel}=0, λ_{np} = 0)	0.46	0.50	0.48	0.84	0.33	0.59
SICTF (λ_{rel}=0, λ_{np} = 0, and no non-negativity constraints)	0.14	0.10	0.12	0.20	0.14	0.17

Table 6: RSI accuracy comparison of SICTF with its ablated versions when no relation side information is used (λ_{rel} = 0), when no NP side information is used (λ_{np} = 0), when no side information of any kind is used (λ_{rel} = 0, λ_{np} = 0), and when additionally there are no non-negative constraints. From this, we observe that additional side information improves performance, validating one of the central thesis of this paper. Please see Section 4.3.2 and Section 4.3.3 for details.

induce better relation schemas, but also do so at a significantly faster speed.

4.3.2 Importance of Side Information

One of the central hypothesis of our approach is that coupled factorization through additional side information should result in better relation schema induction. In order to evaluate this thesis further, we compare performance of SICTF with its ablated versions: (1) SICTF (λ_{rel} = 0), which corresponds to the setting when no relation side information is used, (2) SICTF ($\lambda_{np} = 0$), which corresponds to the setting when no noun phrases side information is used, and (3) SICTF (λ_{rel} = 0, $\lambda_{np} = 0$), which corresponds to the setting when no side information of any kind is used. Hyperparameters are separately tuned for the variants of SICTF. Results are presented in the first four rows of Table 6. From this, we observe that additional coupling through the side information significantly helps improve SICTF performance. This further validates the central thesis of our paper.

4.3.3 Importance of Non-Negativity on Relation Schema Induction

In the last row of Table 6, we also present an ablated version of SICTF when no side information no non-negativity constraints are used. Comparing the last two rows of this table, we observe that non-negativity constraints over the A matrix and core tensor R result in significant improvement in performance. We note that the last row in Table 6 is equivalent to RESCAL (Nickel et al., 2011) and the fourth row is equivalent to Non-Negative RESCAL (Krompaß et al., 2013), two tensor factorization techniques. We also note that none of these tensor factorization techniques have been previously used for the relation schema induction problem.

The reason for this improved performance may be explained by the fact that absence of non-negativity constraint results in an under constrained factorization problem where the model often overgenerates incorrect triples, and then compensates for this overgeneration by using negative latent factor weights. In contrast, imposition of non-negativity constraints restricts the model further forcing it to commit to specific semantics of the latent factors in A. This improved interpretability also results in better RSI accuracy as we have seen above. Similar benefits of non-negativity on interpretability have also been observed in matrix factorization (Murphy et al., 2012).

5 Conclusion

Relation Schema Induction (RSI) is an important first step towards building a Knowledge Graph (KG) out of text corpus from a given domain. While human domain experts have traditionally prepared listing of relations and their schemas, this expert-mediated model poses significant challenges in terms of scalability and coverage. In order to overcome these challenges, in this paper, we present SICTF, a novel non-negative coupled tensor factorization method for relation schema induction. SICTF is flexible enough to incorporate various types of side information during factorization. Through extensive experiments on real-world datasets, we find that SICTF is not only more accurate but also significantly faster (about 14x speedup) compared to state-of-the-art baselines. As part of future work, we hope to analyze SICTF further, as-

sign labels to induced categories, and also apply the model to more domains.

Acknowledgement

Thanks to the members of MALL Lab, IISc who read our drafts and gave valuable feedback and we also thank the reviewers for their constructive reviews. This research has been supported in part by Bosch Engineering and Business Solutions and Google.

References

Evrim Acar, Morten Arendt Rasmussen, Francesco Savorani, Tormod Ns, and Rasmus Bro. 2013. Understanding data fusion within the framework of coupled matrix and tensor factorizations. *Chemometrics and Intelligent Laboratory Systems*, 129(Complete):53–63.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, pages 1797–1807. ACL.

Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014a. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL Association for Computational Linguistics, October.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014b. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579.

Yun-Nung Chen, William Y. Wang, and Alexander I. Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 120–125. IEEE.

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL (1)*, pages 483–494. The Association for Computer Linguistics.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Dora Erdos and Pauli Miettinen. 2013. Discovering facts with boolean tensor tucker decomposition. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1569–1572, New York, NY, USA. ACM.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian Suchanek. 2014. Canonicalizing Open Knowledge Bases. CIKM.

Kartik Goyal, Sujay Kumar, Jauhar Huiying, Li Mrinmaya, Sachan Shashank, and Srivastava Eduard Hovy. 2013. A structured distributional semantic model: Integrating structure with semantics.

R. A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

Denis Krompaß, Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2013. Non-negative tensor factorization with rescal. *Tensor Methods for Machine Learning, ECML workshop*.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Thahir P. Mohamed, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2011. Discovering relations be-

tween noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1447–1455, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Mørup, L. K. Hansen, and S. M. Arnfred. 2008. Algorithms for sparse non-negative TUCKER. *Neural Computation*, 20(8):2112–2131, aug.

Dana Movshovitz-Attias and William W. Cohen. 2015. Kb-lda: Jointly learning a knowledge base of hierarchy, relations, and facts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Brian Murphy, Partha Pratim Talukdar, and Tom M Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING*, pages 1933–1950.

Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. 2012. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25(2):298–324.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA, June. ACM.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271–280, New York, NY, USA. ACM.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84.

Sameer Singh, Tim Rocktäschel, and Sebastian Riedel. 2015. Towards Combined Matrix and Tensor Factorization for Universal Schema Relation Extraction. In *NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C. Denny, Abel N. Kho, You Chen, Bradley A. Malin, and Jimeng Sun. 2015. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams, editors, *KDD*, pages 1265–1274. ACM.

Supervised Distributional Hypernym Discovery via Domain Adaptation

Luis Espinosa-Anke[1], **Jose Camacho-Collados**[2], **Claudio Delli Bovi**[2] and **Horacio Saggion**[1]

[1]Department of Information and Communication Technologies, Universitat Pompeu Fabra
[2]Department of Computer Science, Sapienza University of Rome
[1]{luis.espinosa,horacio.saggion}@upf.edu
[2]{collados,dellibovi}@di.uniroma1.it

Abstract

Lexical taxonomies are graph-like hierarchical structures that provide a formal representation of knowledge. Most knowledge graphs to date rely on *is-a* (hypernymic) relations as the backbone of their semantic structure. In this paper, we propose a supervised distributional framework for hypernym discovery which operates at the sense level, enabling large-scale automatic acquisition of disambiguated taxonomies. By exploiting semantic regularities between hyponyms and hypernyms in embeddings spaces, and integrating a domain clustering algorithm, our model becomes sensitive to the target data. We evaluate several configurations of our approach, training with information derived from a manually created knowledge base, along with hypernymic relations obtained from Open Information Extraction systems. The integration of both sources of knowledge yields the best overall results according to both automatic and manual evaluation on ten different domains.

1 Introduction

Lexical taxonomies (taxonomies henceforth) are graph-like hierarchical structures where terms are nodes, and are typically organized over a predefined merging or splitting criterion (Hwang et al., 2012). By embedding cues about how we perceive concepts, and how these concepts generalize in a domain of knowledge, these resources bear a capacity for generalization that lies at the core of human cognition (Yu et al., 2015) and have become key in Natural Language Processing (NLP) tasks where inference and reasoning have proved to be essential. In fact, taxonomies have enabled a remarkable number of novel NLP techniques, e.g. the contribution of WordNet (Miller, 1995) to lexical semantics (Pilehvar et al., 2013; Yu and Dredze, 2014) as well as various tasks, from word sense disambiguation (Agirre et al., 2014) to information retrieval (Varelas et al., 2005), question answering (Harabagiu et al., 2003) and textual entailment (Glickman et al., 2005). To date, the application of taxonomies in NLP has consisted mainly of, on one hand, formally representing a domain of knowledge (e.g. `Food`), and, on the other hand, constituting the semantic backbone of large-scale knowledge repositories such as ontologies or Knowledge Bases (KBs).

In domain knowledge formalization, prominent work has made use of the web (Kozareva and Hovy, 2010), lexico-syntactic patterns (Navigli and Velardi, 2010), syntactic evidence (Luu Anh et al., 2014), graph-based algorithms (Fountain and Lapata, 2012; Velardi et al., 2013; Bansal et al., 2014) or popularity of web sources (Luu Anh et al., 2015). As for enabling large-scale knowledge repositories, this task often tackles the additional problem of disambiguating word senses and entity mentions. Notable approaches of this kind include Yago (Suchanek et al., 2007), WikiTaxonomy (Ponzetto and Strube, 2008), and the Wikipedia Bitaxonomy (Flati et al., 2014). In addition, while not being taxonomy learning systems *per se*, semi-supervised systems for Information Extraction such as NELL (Carlson et al., 2010) rely crucially on taxonomized concepts and their relations within their learning process.

Taxonomy learning is roughly based on a two-step process, namely *is-a* (hypernymic) *relation de-*

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 424–435,
Austin, Texas, November 1-5, 2016.

tection, and *graph induction*. The hypernym detection phase has gathered much interest not only for taxonomy learning but also for lexical semantics. It has been addressed by means of pattern-based methods[1] (Hearst, 1992; Snow et al., 2004; Kozareva and Hovy, 2010; Carlson et al., 2010; Boella and Di Caro, 2013; Espinosa-Anke et al., 2016), clustering (Yang and Callan, 2009) and graph-based approaches (Fountain and Lapata, 2012; Velardi et al., 2013). Moreover, work stemming from distributional semantics introduced notions of linguistic regularities found in vector representations such as word embeddings (Mikolov et al., 2013d). In this area, supervised approaches, arguably the most popular nowadays, learn a feature vector between term-hypernym vector pairs and train classifiers to predict hypernymic relations. These pairs may be represented either as a concatenation of both vectors (Baroni et al., 2012), difference (Roller et al., 2014), dot-product (Mikolov et al., 2013c), or including additional linguistic information for LSTM-based learning (Shwartz et al., 2016).

In this paper we propose TAXOEMBED[2], a hypernym detection algorithm based on sense embeddings, which can be easily applied to the construction of lexical taxonomies. It is designed to discover hypernymic relations by exploiting linear transformations in embedding spaces (Mikolov et al., 2013b) and, unlike previous approaches, leverages this intuition to learn a specific *semantically-aware transformation matrix* for each domain of knowledge. Our best configuration (ranking first in two thirds of the experiments conducted) considers two training sources: (1) Manually curated pairs from Wikidata (Vrandečić and Krötzsch, 2014); and (2) Hypernymy relations from a KB which integrates several Open Information Extraction (OIE) systems (Delli Bovi et al., 2015a). Since our method uses a very large semantic network as reference sense inventory, we are able to perform jointly hypernym extraction and disambiguation, from which expanding existing ontologies becomes a trivial task. Compared to word-level taxonomy learning, TAXOEMBED results in more refined and unambiguous hypernymic relations at the sense level, with a direct application in tasks such as semantic search. Evaluation (both manual and automatic) shows that we can effectively replicate the Wikidata *is-a* branch, and capture previously unseen relations in other reference taxonomies (YAGO or WIBI).

2 Related Work

Pattern-based methods for hypernym identification exploit the joint co-ocurrence of term and hypernym in text corpora. Building up on Hearst's patterns (Hearst, 1992), these approaches have focused on, for instance, exploiting templates for harvesting candidate instances which are ranked via mutual information (Etzioni et al., 2005), training a classifier with WordNet hypernymic relations combined with syntactic dependencies (Snow et al., 2006), or applying a doubly-anchored method (Kozareva and Hovy, 2010), which queries the web with two semantically related terms for collecting domain-specific corpora. Syntactic information is also used for supervised definition and hypernym extraction (Navigli and Velardi, 2010; Boella and Di Caro, 2013), or together with Wikipedia-specific heuristics (Flati et al., 2014). One of the main drawbacks of these methods is that they require both term and hypernym to co-occur in text within a certain window, which strongly hinders their recall. Higher recall can be achieved thanks to distributional methods, as they do not have co-occurrence requirements. In addition, they can be tailored to cover any number of predefined semantic relations such as co-hyponymy or meronymy (Baroni and Lenci, 2011), but also cause-effect or entity-origin (Hendrickx et al., 2009). However, they are often more imprecise and seem to perform best in discovering broader semantic relations (Shwartz et al., 2016).

One way to surmount the issue of generality was proposed by Fu et al. (2014), who explored the possibility to learn a *hypernymic* transformation matrix over a word embeddings space. As shown empirically in Fu et al.'s original work, the hypernymic relation that holds for the pair (*dragonfly*, *insect*) differs from the one of e.g. (*carpenter*, *man*). Prior to

[1]The terminology is not entirely unified in this respect. In addition to *pattern-based* (Fountain and Lapata, 2012; Bansal et al., 2014; Yu et al., 2015), other terms like *path-based* (Shwartz et al., 2016) or *rule-based* (Navigli and Velardi, 2010) are also used.

[2]Data and source code available from the following link: `www.taln.upf.edu/taxoembed`.

training, their system addresses this discrepancy via k-means clustering using a held-out development set for tuning.

The previously described methods for hypernym and taxonomy learning operate inherently at the surface level. This is partly due to the way evaluation is conducted, which is often limited to very specific domains with no integrative potential (e.g. taxonomies in `food`, `science` or `equipment` from Bordea et al. (2015)), or restricted to lists of word pairs. Hence, a drawback of surface-level taxonomy learning, apart from ambiguity issues, is that they require additional and error-prone steps to identify semantic clusters (Fu et al., 2014).

Alternatively, recent advances in OIE based on disambiguation and deeper semantic analysis (Nakashole et al., 2012; Grycner and Weikum, 2014; Delli Bovi et al., 2015b) have shown their potential to construct taxonomized disambiguated resources both at node and at relation level. However, in addition to their inherently broader scope, OIE approaches are designed to achieve high coverage, and hence they tend to produce noisier data compared to taxonomy learning systems.

In our sense-based approach, instead, not only do we leverage an unambiguous vector representation for hypernym discovery, but we also take advantage of a domain-wise clustering strategy to directly obtain specific term-hypernym training pairs, thereby substantially refining this step. Additionally, we exploit the complementary knowledge of OIE systems by incorporating high-confidence relation triples drawn from OIE-derived resources, yielding the best average configuration as evaluated on ten different domains of knowledge.

3 Preliminaries

TAXOEMBED leverages the vast amounts of training data available from structured and unstructured knowledge resources, along with the mapping among these resources and a state-of-the-art vector representation of word senses.

BabelNet[3] (Navigli and Ponzetto, 2012) constitutes our sense inventory, as it is currently the largest single multilingual repository of named entities and concepts, integrating various resources such as WordNet, Wikipedia or Wikidata. As in WordNet, BabelNet is structured in synsets. Each synset is composed of a set of words (*lexicalizations* or *senses*) representing the same meaning. For instance, the synset referring to *the members of a business organization* is represented by the set of senses *firm, house, business firm*. BabelNet contains around 14M synsets in total. We exploit BabelNet[4] as (1) A repository for the manually-curated hypernymic relations included in **Wikidata**; (2) A semantic pivot of the integration of several OIE systems into one single resource, namely **KB-UNIFY**; and (3) A sense inventory for the **SENSEMBED** vector representations. In the following we provide further details about each of these resources.

3.1 Training Data

Wikidata[5] (Vrandečić and Krötzsch, 2014) is a document-oriented semantic database operated by the Wikimedia Foundation with the goal of providing a common source of data that can be used by other Wikimedia projects. Our initial training set $\mathcal{W}$ consists of the hypernym branch of Wikidata, specifically the version included in BabelNet. Each term-hypernym $\in \mathcal{W}$ is in fact a pair of BabelNet synsets, e.g. the synset for *Apple* (with the company sense), and the concept *company*.

KB-UNIFY[6] (Delli Bovi et al., 2015a) (KB-U) is a knowledge-based approach, based on BabelNet, for integrating the output of different OIE systems into a single unified and disambiguated knowledge repository. The unification algorithm takes as input a set **K** of OIE-derived resources, each of which is modeled as a set of ⟨entity, relation, entity⟩ triples, and comprises two subsequent stages: in the first *disambiguation* stage, each KB in **K** is linked to the sense inventory of BabelNet by disambiguating its relation argument pairs; in the following *alignment* stage, equivalent relations across different KB in **K** are merged together. As a result, KB-U generates a KB of triples where arguments are linked to the corresponding BabelNet synsets, and relations are replaced by *relation synsets* of semantically

[3] `http://babelnet.org`

[4] We use BabelNet 3.0 release version in our experiments.

[5] `https://www.wikidata.org`

[6] `http://lcl.uniroma1.it/kb-unify`

similar OIE-derived relation patterns. The original experimental setup of KB-UNIFY included NELL (Carlson et al., 2010) as one of its input resources: since NELL features its own manually-built taxonomic structure and relation type inventory (hence its own *is-a* relation type), we identified the relation synset containing NELL's *is-a*[7] and then drew from the unified KB all the corresponding triples, which we denote as $\mathcal{K}$. These triples constitute, similarly as in the previous case, a set of term-hypernym pairs automatically extracted from OIE-derived resources, with a disambiguation confidence of above 0.9 according to the disambiguation strategy described in the original paper.

Initially, $|\mathcal{W}|$ = 5,301,867 and $|\mathcal{K}|$ = 1,358,949.

3.2 Sense vectors

SENSEMBED (Iacobacci et al., 2015)[8] constitutes the sense embeddings space that we use for training our hypernym detection algorithm. Vectors in the SENSEMBED space, denoted as $\mathcal{S}$, are latent continuous representations of word senses based on the Word2Vec architecture (Mikolov et al., 2013a), which was applied on a disambiguated Wikipedia corpus. Each vector $\vec{v} \in \mathcal{S}$ represents a BabelNet sense, i.e. a synset along with one of its lexicalizations (e.g. *album_chart_bn:00002488n*). This differs from unsupervised approaches (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014) that learn sense representations from text corpora only and are not mapped to any lexical resource, limiting their application in our task.

4 Methodology

Our approach can be summarized as follows. First, we take advantage of a clustering algorithm for allocating each BabelNet synset of the training set into a domain cluster C (Section 4.1). Then, we expand the training set by exploiting the different lexicalizations available for each BabelNet synset (Section 4.2). Finally, we learn a cluster-wise linear projection (a *hypernym transformation matrix*) over all pairs (term-hypernym) of the expanded training set (Section 4.3).

[7] represented by the relation `generalizations`.

[8] `http://lcl.uniroma1.it/sensembed`

4.1 Domain Clustering

Fu et al. (2014) induced semantic clusters via k-means, where k was tuned on a development set. Instead, we aim at learning a function sensitive to a predefined knowledge domain, under the assumption that vectors clustered with this criterion are likely to exhibit similar semantic properties (e.g. similarity). First, we allocate each synset into its most representative domain, which is achieved by exploiting the set of thirty four domains available in the Wikipedia featured articles page[9]. `Warfare`, `transport`, or `music` are some of these domains. In the Wikipedia featured articles page each domain is composed of 128 Wikipedia pages on average. Then, in order to expand the set of concepts associated with each domain, we leverage NASARI[10] (Camacho-Collados et al., 2015), a distributional approach that has been used to construct explicit vector representations of BabelNet synsets.

Our goal is to associate BabelNet synsets with domains. To this end, we follow Camacho-Collados et al. (2016) and build a lexical vector for each Wikipedia domain by concatenating all Wikipedia pages representing the given domain into a single text. Finally, given a BabelNet synset b, we calculate the similarity between its corresponding NASARI lexical vector and all the domain vectors, selecting the domain leading to the highest similarity score:

$$\hat{d}(b) = \max_{d \in D} WO(\vec{d}, \vec{b}) \quad (1)$$

where D is the set of all thirty-three domains, $\vec{d}$ is the vector of the domain $d \in D$, $\vec{b}$ is the vector of the BabelNet synset b, and WO refers to the *Weighted Overlap* comparison measure (Pilehvar et al., 2013), which is defined as follows:

$$WO(\vec{v_1}, \vec{v_2}) = \sqrt{\frac{\sum_{w \in O} \left(rank_{w,\vec{v_1}} + rank_{w,\vec{v_2}}\right)^{-1}}{\sum_{i=1}^{|O|}(2i)^{-1}}} \quad (2)$$

where $rank_{w,\vec{v_i}}$ is the rank of the word w in the vector $\vec{v_i}$ according to its weight, and O is the set of overlapping words between the two vectors. In order to have a highly reliable set of domain labels, those

[9] `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`

[10] `http://lcl.uniroma1.it/nasari`

synsets whose maximum similarity score is below a certain threshold are not annotated with any domain. We fixed the threshold to 0.35, which provided a fine balance between precision (estimated in around 85%) and recall in our development set. By following this approach almost 2 million synsets are labelled with a domain.

4.2 Training Data Expansion

Prior to training our model, we benefit from the fact that a given BabelNet synset may be associated with a fixed number of lexicalizations or senses, i.e. different ways of referring to the same concept, to expand our set of training pairs. For instance, the synset b associated with the concept *music_album* is represented by the set of lexicalizations $\mathcal{L}_b = \{$album, music_album ... album_project$\}$. We take advantage of this synset representation to expand each term-hypernym synset pair. For each term-hypernym pair, both concepts are expanded to their given lexicalizations and thus, each synset pair term-hypernym in the training data is expanded to a set of $|\mathcal{L}_t|.|\mathcal{L}_h|$ sense training pairs.

This expansion step results in much larger sets $\mathcal{W}^*$ and $\mathcal{K}^*$, where $|\mathcal{W}^*| = 18{,}291{,}330$ and $|\mathcal{K}^*| = 15{,}362{,}268$. Specifically, they are 3 and 11 times bigger than the original training sets described in Section 3.1. These numbers are higher than those reported in recent approaches for hypernym detection, which exploited Chinese semantic thesauri along with manual validation of hypernym pairs (Fu et al., 2014) (obtaining a total of 1,391 instances), or pairs from knowledge resources such as Wikidata, Yago, WordNet and DBpedia (Shwartz et al., 2016), where the maximum reported split for training data (70%) amounted to 49,475 pairs.

4.3 Learning a Hypernym Detection Matrix

The gist of our approach lies on the property of current semantic vector space models to capture relations between vectors, in our case hypernymy. This can be found even in disjoint spaces, where this property has been exploited for machine translation (Mikolov et al., 2013b) or language normalization (Tan et al., 2015). For our purposes, however, instead of learning a global linear transformation function in two spaces over a broad relation like hypernymy, we learn a function sensitive to a given domain of knowledge. Thus, our training data becomes restricted to those term-hypernym BabelNet sense pairs $\left(x^d, y^d\right) \in C_d \times C_d$, where C_d is the cluster of BabelNet synsets labelled with the domain d.

For each domain-wise expanded training set T^d, we construct a hyponym matrix $\mathbf{X}^d = [\vec{x}_1^d \dots \vec{x}_n^d]$ and a hypernym matrix $\mathbf{Y}^d = [\vec{y}_i^d \dots \vec{y}_n^d]$, which are composed by the corresponding SENSEMBED vectors of the training pairs $\left(x_i^d, y_i^d\right) \in C_d \times C_d, 0 \leq i \leq n$.

Under the intuition that there exists a matrix Ψ so that $\vec{y}^d = \Psi\vec{x}^d$, we learn a transformation matrix for each domain cluster C_d by minimizing:

$$\min_{\Psi^C} \sum_{i=1}^{|T^d|} \|\Psi^C \vec{x}_i^d - \vec{y}_i^d\|^2 \tag{3}$$

Then, for any unseen term x^d, we obtain a ranked list of the most likely hypernyms of its lexicalization vectors $\vec{x_j}^d$, using as measure cosine similarity:

$$\text{argmax}_{\vec{v}\in\mathcal{S}} \frac{\vec{v} \cdot \Psi^C \vec{x_j}^d}{||\vec{v}||||\Psi^C \vec{x_j}^d||} \tag{4}$$

At this point, we have associated with each sense vector a ranked list of candidate hypernym vectors. However, in the (frequent) cases in which one synset has more than one lexicalization, we need to condense the results into one single list of candidates, which we achieve with a simple ranking function $\lambda(\cdot)$, which we compute as $\lambda(\vec{v}) = \frac{cos(\vec{v}, \Psi^C \vec{x}^d)}{rank(\vec{v})}$, where $rank(\vec{v})$ is the rank of $\vec{v}$ according to its cosine similarity with $\Psi^C \vec{x}^d$.

The above operations allow us to cast the hypernym detection task as a ranking problem. This is also particularly interesting to enable a flexible evaluation framework where we can combine highly demanding metrics for the quality of the candidate given at a certain rank, as well as other measures which consider the rank of the first valid retrieved candidate.

5 Evaluation

The performance of TAXOEMBED is evaluated by conducting several experiments, both automatic and manual. Specifically, we assess its ability to return valid hypernyms for a given unseen term with

a held-out evaluation dataset of 250 Wikidata term-hypernym pairs (Section 5.1). In addition, we assess the extent to which TAXOEMBED is able to correctly identify hypernyms *outside of Wikidata* (Section 5.2).

5.1 Experiment 1: Automatic Evaluation

5.1.1 Experimental setting

For each domain, we retain 5k, 10k, 15k, 20k and 25k Wikidata term-hypernym training pairs for different experiments, and evaluate on 250 test pairs for each of the 10 domains. Moreover, we aim at improving TAXOEMBED by including 1k and 25k extra OIE-derived training pairs per domain (generating two more systems, namely 25k+K^d_{1k} and 25k+K^d_{25k}). These OIE-derived instances are those contained in KB-U (see Section 3.1). Moreover, in order to quantify the empirically grounded intuition of the need to train a cluster-wise transformation matrix (Fu et al., 2014), we also introduce an additional configuration at 25k (25k+K^r_{50k}), where we include 50k additional pairs randomly from KB-U, and two more settings with only random pairs coming from Wikidata (100k$^r_{wd}$) and KB-U (100k+$^r_{kbu}$).

We also include a distributional supervised baseline[11] based on word analogies (Mikolov et al., 2013a), computed as follows. First, we calculate the difference vector of each training SENSEMBED vector pair ($\vec{x}^d$,$\vec{y}^d$) of a given domain d. Then, we average all the difference vectors of all training pairs to obtain a global vector $\vec{V}_d$ for the domain d. Finally, given a test term t we calculate the closest vector of the sum of the corresponding term vector and $\vec{V}_d$:

$$\hat{t} = \text{argmax}_{\vec{v} \in \mathcal{S}} \vec{V}_d + \vec{t} \quad (5)$$

This baseline has shown to capture different semantic relations and to improve as training data increases (Mikolov et al., 2013a).

Evaluation metrics. We computed, for each domain and for the above configurations, the following metrics: Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and R-Precision (R-P). These measures provide insights on different aspects of the outcome of the task, e.g. how often valid hypernyms were retrieved in the first positions of the rank (MRR), and if there were more than one valid hypernym, whether this set was correctly retrieved, (MAP and R-P)[12].

5.1.2 Results and discussion

We summarize the main outcome of our experiments in Table 1. Results suggest that the performance of TAXOEMBED increases as training data expands. This is consistent with the findings shown in Mikolov et al. (2013b), who showed a substantial improvement in accuracy in the machine translation task by gradually increasing the training set. Additionally, the improvement of TAXOEMBED over the baseline is consistent across most evaluation domain clusters and metrics, with domain-filtered data from KB-U contributing to the learning process in about two thirds of the evaluated configurations. These are very encouraging results considering the noisy nature of OIE systems, and that the resource we obtained from KB-U is the result of error-prone steps such as Word Sense Disambiguation and Entity Linking, as well as relation clustering.

As far as the individual domains are concerned, the `biology` domain seems to be easier to model than the rest, likely due to the fact that fauna and flora are areas where hierarchical division of species is a field of study in itself, which traces back to Aristotelian times (Mayr, 1982), and therefore has been constantly refined over the years. Also, it is notable how well the 100k$^r_{wd}$ configuration performs on this domain. This is the only domain in which training with no semantic awareness gives good results. We argue that this is highly likely due to the fact that a vast amount of synsets are allocated into the `biology` cluster (60% of them, and up to 80% in hypernym position). This produces the so-called lexical memorization phenomenon (Levy et al., 2015), as the system memorizes prototypical biology-related hypernyms like *taxon* as valid hypernyms for many concepts. This contrasts with the lower presence of other domains, e.g. 5% in `media`, 4% in `music`, or 2% in `transport`.

Another remarkable case involves the `education` and `media` domains, which experience the highest improvement when training with KB-U (5 and 6 MRR points, respectively).

[11] Using the 25k domain-filtered expanded Wikidata pairs as training set.

[12] See Bian et al. (2008) for an in-depth analysis of these metrics.

	art			biology			education			geography			health		
Train	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P
5k	0.12	0.12	0.12	0.63	0.63	0.59	0.00	0.00	0.00	0.08	0.07	0.07	0.08	0.08	0.07
15k	0.21	0.20	0.18	**0.84**	0.72	0.79	0.22	0.22	0.21	0.15	0.14	0.14	0.08	0.07	0.07
25k	**0.29**	0.27	**0.26**	**0.84**	**0.83**	**0.81**	0.33	0.32	0.30	**0.23**	**0.22**	**0.21**	0.09	0.09	0.08
25k+K^d_{1k}	**0.29**	**0.28**	**0.26**	**0.84**	0.80	0.79	0.32	0.29	0.27	0.22	**0.22**	**0.21**	0.09	0.09	0.08
25k+K^d_{25k}	0.26	0.24	0.22	0.70	0.63	0.56	**0.38**	**0.36**	**0.33**	0.15	0.13	0.12	0.11	0.11	0.10
25k+K^r_{50k}	0.28	0.26	0.24	0.82	0.77	0.72	0.36	0.33	0.30	0.17	0.16	0.16	**0.12**	0.11	0.10
100k$^r_{wd}$	0.00	0.00	0.00	**0.84**	0.81	0.77	0.00	0.00	0.00	0.01	0.01	0.01	0.07	0.06	0.06
100k$^r_{kbu}$	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	**0.12**	0.12	0.11
Baseline	0.13	0.12	0.10	0.58	0.57	**0.57**	0.10	0.10	0.09	0.12	0.09	0.05	0.07	**0.13**	**0.14**
	media			music			physics			transport			warfare		
Train	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P	MRR	MAP	R-P
5k	0.28	0.28	0.27	0.10	0.10	0.09	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01
15k	0.14	0.13	0.12	0.08	0.07	0.07	0.36	0.35	0.34	0.25	0.23	0.21	0.01	0.01	0.01
25k	0.46	0.45	0.43	0.30	0.28	0.26	**0.41**	**0.40**	**0.38**	0.46	0.43	0.39	**0.05**	**0.05**	**0.04**
25k+K^d_{1k}	0.43	0.42	0.41	**0.32**	**0.30**	**0.28**	0.39	0.38	0.37	0.47	0.44	0.40	0.04	0.04	0.01
25k+K^d_{25k}	0.52	**0.51**	0.49	0.26	0.25	0.23	0.37	0.36	0.34	0.48	0.45	0.41	0.04	0.03	0.03
25k+K^r_{50k}	0.46	0.45	0.43	0.29	0.28	0.25	0.31	0.30	0.29	**0.52**	**0.49**	**0.46**	**0.05**	0.04	**0.04**
100k$^r_{wd}$	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.01
100k$^r_{kbu}$	0.08	0.07	0.07	0.01	0.01	0.00	0.00	0.00	0.00	0.10	0.10	0.10	0.00	0.00	0.00
Baseline	**0.57**	0.43	**0.52**	0.03	0.03	0.03	0.05	0.04	0.04	0.29	0.25	0.21	0.04	0.04	**0.04**

Table 1: Overview of the performance of TAXOEMBED using different training data samples.

One of the main sources for *is-a* relations in KB-U is NELL, which contains a vast amount of relation triples between North American academic entities (professors, sports teams, alumni, donators; as well as media celebrities). Many of these entities are missing in Wikidata, and relations among them encoded in NELL are likely to be correct because in most cases these are unambiguous entities which occur in the same communicative contexts. For example, leveraging KB-U we were able to include the pair (*university_of_north_wales*, *four_year_college*), which is absent in Wikidata. In fact, many high quality *is-a* pairs like this can be found in KB-U for these two domains.

We also computed P@k (number of valid hypernyms on the first k returned candidates), where k ranges from 1 to 5. Numbers are on the line of the results shown in Table 1 and therefore are not provided in detail. The main trend we found is showcased in Figure 1, which shows an illustrative example from the `transport` domain. As we can see, all values of k exhibit a similar performance curve, with a gradual increase of performance as the training set becomes larger.

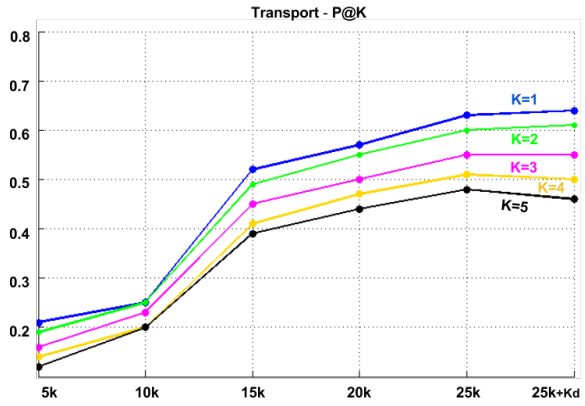


Figure 1: P@k scores for the `transport` domain.

False positives. We complement this experiment with a manual evaluation of *theoretical* false positives. Our intuition is that due to the nature of the task, some domains may be more flexible in allow-

ing two terms to encode an *is-a* relation, while others may be more restrictive. We asked human judges to manually validate a sample of 200 *wrong pairs* from our best run in each domain, and estimated precision over them. As expected, *hard science* domains like `physics` obtain very low results (about 1% precision). In contrast, other domains like `education` (12% precision), or `transport` (16% precision), probably due to their multidisciplinary nature, allow more valid hypernyms for a given term than what is currently encoded in Wikidata.

5.2 Experiment 2: Extra-Coverage

In this experiment we evaluate the performance of TAXOEMBED on instances not included in Wikidata. We describe the experimental setting in Section 5.2.1 and present the results in Section 5.2.2.

5.2.1 Experimental setting

For this experiment we use two configurations of TAXOEMBED: the first one includes 25k domain-wise expanded training pairs (TaxE_{25k}), whereas the second one adds 1k pairs from KB-U (TaxE_{25k+K^d}). We randomly extract 200 test BabelNet synsets (20 per domain) whose hypernyms are missing in Wikidata. We compare against a number of taxonomy learning and Information Extraction systems, namely Yago (Suchanek et al., 2007), WiBi (Flati et al., 2014) and DefIE (Delli Bovi et al., 2015b). Yago and WiBi are used as *upper bounds* due to the nature of their hypernymic relations. They include a great number of manually-encoded taxonomies (e.g. exploiting WordNet and Wikipedia categories). Yago derives its taxonomic relations from an automatic mapping between WordNet and Wikipedia categories. WiBi, on the other hand, exploits, among a number of different Wikipedia-specific heuristics, categories and the syntactic structure of the introductory sentence of Wikipedia pages. Finally, DefIE is an automaic OIE system relying on the syntactic structure of pre-disambiguated definitions[13]. Three annotators manually evaluated the validity of the hypernyms extracted by each system (one per test instance).

5.2.2 Results and discussion

Table 2 shows the results of TAXOEMBED and all comparison systems. As expected, Yago and WiBi achieve the best overall results. However, TAXOEMBED, based solely on distributional information, performed competitively in detecting new hypernyms when compared to DefIE, improving its recall in most domains, and even surpassing Yago in technical areas like `biology` or `health`. However, our model does not perform particularly well on `media` and `physics`. In most domains our model is able to discover novel hypernym relations that are not captured by any other system (e.g. *therapy* for *radiation treatment planning* in the `health` domain or *decoration* for *molding* in the `art` domain)[14].

In fact, the overlap between our approach and the remaining systems is actually quite small (on average less than 25% with all of them on the Extra-Coverage experiment). This is mainly due to the fact that TAXOEMBED only exploits distributional information and does not make use of predefined syntactic heuristics, suggesting that the information it provides and the rule-based comparison systems may be complementary. We foresee a potential avenue focused on combining a supervised distributional approach such as TAXOEMBED with syntactically-motivated systems such as Wibi or Yago. This combination of a distributional system and manual patterns was already introduced by Shwartz et al. (2016) on the hypernym detection task with highly encouraging results.

6 Conclusion

We have presented TAXOEMBED, a supervised taxonomy learning framework exploiting the property that was observed in Fu et al. (2014), namely that there exists, for a given domain-specific terminology, a shared linear projection among term-hypernym pairs. We showed how this can be used to learn a hypernym transformation matrix for discovering novel *is-a* relations, which are the backbone of lexical taxonomies. First, we allocate almost 2M BabelNet synsets into a predefined domain of knowledge. Then, we collect training data both from a manually constructed knowledge base (Wiki-

[13]For this experiment, we included DefIE's *is-a* relations only.

[14]For simplicity, we use the word surface form to refer to BabelNet synsets.

	art			biology			education			geography			health		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
TaxE$_{25k}$	0.45	0.45	0.45	**0.40**	**0.40**	**0.40**	**0.60**	**0.60**	**0.60**	0.35	0.35	0.35	**0.45**	**0.45**	**0.45**
TaxE$_{25k+K^d}$	0.50	**0.50**	**0.50**	**0.40**	**0.40**	**0.40**	0.55	0.55	0.55	0.35	0.35	0.35	**0.45**	**0.45**	**0.45**
DefIE	**0.63**	0.35	0.45	0.36	0.20	0.25	0.57	0.20	0.29	**0.66**	**0.40**	**0.50**	0.25	0.15	0.18
Yago	0.88	0.75	0.81	0.62	0.25	0.36	0.94	0.80	0.86	0.79	0.75	0.77	0.28	0.10	0.15
Wibi	0.70	0.70	0.70	0.58	0.50	0.54	0.94	0.80	0.86	0.75	0.75	0.75	0.66	0.50	0.57
	media			music			physics			transport			warfare		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
TaxE$_{25k}$	0.10	0.10	0.10	0.45	0.45	0.45	0.15	**0.15**	0.15	0.35	**0.35**	0.35	0.25	0.25	0.25
TaxE$_{25k+K^d}$	0.10	0.10	0.10	0.40	0.40	0.40	0.15	**0.15**	0.15	0.25	0.25	0.25	0.45	**0.45**	**0.45**
DefIE	**0.81**	**0.45**	**0.58**	**0.71**	**0.50**	**0.58**	**0.42**	**0.15**	**0.22**	**0.54**	0.30	**0.38**	**0.60**	0.30	0.40
Yago	0.76	0.65	0.70	0.84	0.55	0.67	0.80	0.40	0.53	0.93	0.70	0.80	0.81	0.65	0.72
Wibi	0.90	0.90	0.90	0.89	0.85	0.87	0.68	0.55	0.61	0.87	0.70	0.77	0.66	0.50	0.57

Table 2: Precision, recall and F-Measure between TAXOEMBED, two taxonomy learning systems (Yago and WiBi), and a pattern-based approach that performs hypernym extraction (DefIE).

data), and from OIE systems. We substantially expand our initial training set by expanding both terms and hypernyms to all their available senses, and in a last step, to their corresponding disambiguated vector representations.

Evaluation shows that the general trend is that our hypernym matrix improves as we increase training data. Our best domain-wise configuration combines 25k training pairs from Wikidata and additional pairs from an OIE-derived KB, achieving promising results. The domains in which the addition of the OIE-based information contributed the most are `education`, `transport` and `media`. For instance, in the case of `education`, this may be due to the over representation of the North American educational system in IE systems like NELL. We accompany this quantitative evaluation with manual assessment of precision of false positives, and an analysis of the potential coverage comparing it with knowledge taxonomies like Yago or WiBi, and with DefIE, a *quasi*-OIE system.

7 Future Work

For future work we are planning to apply this strategy to learn large-scale semantic relations beyond hypernymy. This may constitute a first step towards a global and fully automatic ontology learning system. In the context of semantic web, we would like to include semantic parsers and distant supervision to our algorithm in order to capture n-ary relations between pairs of concepts to further create and improve existing KBs.

As mentioned in Section 5.2.2, we are also planning to combine our distributional approach with rule-based heuristics, following the line of work introduced by Shwartz et al. (2016). Finally, we see potential in the domain clustering approach for improving graph-based taxonomy learning systems, as it can serve as a weighting measure as to how pertinent a given set of concepts in a taxonomy are for a specific domain.

Acknowledgments

This work is partially funded by the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502) and under the TUNER project (TIN2015-65308-C5-5-R, MINECO/FEDER, UE). The authors also acknowledge support from Dr. Inventor (FP7-ICT-2013.8.1611383). José Camacho-Collados is supported by a Google Doctoral Fellowship in Natural Language Processing. We would also like to thank Tommaso Pasini for helping us to compute the Wibi and Yago baselines in our second experiment.

References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *ACL (1)*, pages 1041–1051.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

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

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web*, pages 467–476. ACM.

Guido Boella and Luigi Di Caro. 2013. Supervised learning of syntactic contexts for uncovering definitions and extracting hypernym relations in text databases. In *Machine learning and knowledge discovery in databases*, pages 64–79. Springer.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (texeval). In *Proceedings of the SemEval workshop*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A Unified Multilingual Semantic Representation of Concepts. In *Proceedings of ACL*, pages 741–751, Beijing, China.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, pages 1306–1313.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015a. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of EMNLP*, pages 726–736, Lisbon, Portugal, September. Association for Computational Linguistics.

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015b. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. *TACL*, 3:529–543.

Luis Espinosa-Anke, Horacio Saggion, Francesco Ronzano, and Roberto Navigli. 2016. Extasem! extending, taxonomizing and semantifying domain terminologies. AAAI.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *ACL*.

Trevor Fountain and Mirella Lapata. 2012. Taxonomy induction using hierarchical random graphs. In *Proceedings of NAACL*, pages 466–476. Association for Computational Linguistics.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of ACL*, volume 1.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. A probabilistic classification approach for lexical textual entailment. In *Proceedings of the National Conference On Artificial Intelligence*, page 1050.

Adam Grycner and Gerhard Weikum. 2014. Harpy: Hypernyms and alignment of relational paraphrases. In *Proceedings of COLING*, pages 2195–2204, Dublin, Ireland.

Sanda M Harabagiu, Steven J Maiorano, and Marius A Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(03):231–267.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval: Recent Achievements and Future Directions*, pages 94–99.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

Sung Ju Hwang, Kristen Grauman, and Fei Sha. 2012. Semantic kernel forests from multiple taxonomies. In

Advances in Neural Information Processing Systems, pages 1718–1726.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of ACL*, pages 95–105, Beijing, China.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of EMNLP*, pages 1110–1118.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations? In *NAACL 2015*, Denver, Colorado, USA.

Tuan Luu Anh, Jung-jae Kim, and See Kiong Ng. 2014. Taxonomy construction using syntactic contextual evidence. In *Proceedings of EMNLP*, pages 810–819.

Tuan Luu Anh, Jung-jae Kim, and See-Kiong Ng. 2015. Incorporating trustiness and collective synonym/contrastive evidence into taxonomy construction. In *Proceedings of EMNLP*, pages 1013–1022.

Ernst Mayr. 1982. *The growth of biological thought: Diversity, evolution, and inheritance*. Harvard University Press.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of EMNLP-CoNLL*, pages 1135–1145.

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

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *ACL*, pages 1318–1327.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*, pages 1059–1069, Doha, Qatar.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: a Unified Approach for Measuring Semantic Similarity. In *Proceedings of ACL*, pages 1341–1351, Sofia, Bulgaria.

Simone Paolo Ponzetto and Michael Strube. 2008. Wikitaxonomy: A large scale knowledge resource. In *ECAI*, volume 178, pages 751–752.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014*, Dublin, Ireland.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. *arXiv preprint arXiv:1603.06076*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL 2006*, pages 801–808.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *WWW*, pages 697–706. ACM.

L. Tan, H. Zhang, C. Clarke, and M. Smucker. 2015. Lexical comparison between wikipedia and twitter corpora by using word embeddings. In *Proceedings of ACL (2)*, pages 657–661, Beijing, China, July.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160.

Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides GM Petrakis, and Evangelos E Milios. 2005. Semantic similarity methods in wordnet and their application to information retrieval on the web. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16. ACM.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of ACL/IJCNLP*, pages 271–279. Association for Computational Linguistics.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL (2)*, pages 545–550.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of IJCAI*, pages 1390–1397.

Latent Tree Language Model

Tomáš Brychcín
NTIS – New Technologies for the Information Society,
Faculty of Applied Sciences, University of West Bohemia,
Technická 8, 306 14 Plzeň, Czech Republic
brychcin@kiv.zcu.cz
nlp.kiv.zcu.cz

Abstract

In this paper we introduce Latent Tree Language Model (LTLM), a novel approach to language modeling that encodes syntax and semantics of a given sentence as a tree of word roles.

The learning phase iteratively updates the trees by moving nodes according to Gibbs sampling. We introduce two algorithms to infer a tree for a given sentence. The first one is based on Gibbs sampling. It is fast, but does not guarantee to find the most probable tree. The second one is based on dynamic programming. It is slower, but guarantees to find the most probable tree. We provide comparison of both algorithms.

We combine LTLM with 4-gram Modified Kneser-Ney language model via linear interpolation. Our experiments with English and Czech corpora show significant perplexity reductions (up to 46% for English and 49% for Czech) compared with standalone 4-gram Modified Kneser-Ney language model.

1 Introduction

Language modeling is one of the core disciplines in natural language processing (NLP). Automatic speech recognition, machine translation, optical character recognition, and other tasks strongly depend on the language model (LM). An improvement in language modeling often leads to better performance of the whole task. The goal of language modeling is to determine the joint probability of a sentence. Currently, the dominant approach is n-gram language modeling, which decomposes the joint probability into the product of conditional probabilities by using the *chain rule*. In traditional n-gram LMs the words are represented as distinct symbols. This leads to an enormous number of word combinations.

In the last years many researchers have tried to capture words contextual meaning and incorporate it into the LMs. Word sequences that have never been seen before receive high probability when they are made of words that are semantically similar to words forming sentences seen in training data. This ability can increase the LM performance because it reduces the *data sparsity* problem. In NLP a very common paradigm for word meaning representation is the use of the *Distributional hypothesis*. It suggests that two words are expected to be semantically similar if they occur in similar contexts (they are similarly distributed in the text) (Harris, 1954). Models based on this assumption are denoted as distributional semantic models (DSMs).

Recently, semantically motivated LMs have begun to surpass the ordinary n-gram LMs. The most commonly used architectures are *neural network LMs* (Bengio et al., 2003; Mikolov et al., 2010; Mikolov et al., 2011) and *class-based LMs*. Class-based LMs are more related to this work thus we investigate them deeper.

Brown et al. (1992) introduced class-based LMs of English. Their unsupervised algorithm searches classes consisting of words that are most probable in the given context (one word window in both directions). However, the computational complexity of this algorithm is very high. This approach was later extended by (Martin et al., 1998; Whit-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 436–446,
Austin, Texas, November 1-5, 2016.

taker and Woodland, 2003) to improve the complexity and to work with wider context. Deschacht et al. (2012) used the same idea and introduced Latent Words Language Model (LWLM), where word classes are latent variables in a graphical model. They apply Gibbs sampling or the expectation maximization algorithm to discover the word classes that are most probable in the context of surrounding word classes. A similar approach was presented in (Brychcín and Konopík, 2014; Brychcín and Konopík, 2015), where the word clusters derived from various semantic spaces were used to improve LMs.

In above mentioned approaches, the meaning of a word is inferred from the surrounding words independently of their relation. An alternative approach is to derive contexts based on the syntactic relations the word participates in. Such syntactic contexts are automatically produced by dependency parse-trees. Resulting word representations are usually less topical and exhibit more functional similarity (they are more syntactically oriented) as shown in (Padó and Lapata, 2007; Levy and Goldberg, 2014).

Dependency-based methods for syntactic parsing have become increasingly popular in NLP in the last years (Kübler et al., 2009). Popel and Mareček (2010) showed that these methods are promising direction of improving LMs. Recently, unsupervised algorithms for dependency parsing appeared in (Headden III et al., 2009; Cohen et al., 2009; Spitkovsky et al., 2010; Spitkovsky et al., 2011; Mareček and Straka, 2013) offering new possibilities even for poorly-resourced languages.

In this work we introduce a new DSM that uses tree-based context to create word roles. The word role contains the words that are similarly distributed over similar tree-based contexts. The word role encodes the semantic and syntactic properties of a word. We do not rely on parse trees as a prior knowledge, but we jointly learn the tree structures and word roles. Our model is a soft clustering, i.e. one word may be present in several roles. Thus it is theoretically able to capture the word polysemy. The learned structure is used as a LM, where each word role is conditioned on its parent role. We present the unsupervised algorithm that discovers the tree structures only from the distribution of words in a training corpus (i.e. no labeled data or external sources of information are needed). In our work we were inspired by class-based LMs (Deschacht et al., 2012), unsupervised dependency parsing (Mareček and Straka, 2013), and tree-based DSMs (Levy and Goldberg, 2014).

This paper is organized as follows. We start with the definition of our model (Section 2). The process of learning the hidden sentence structures is explained in Section 3. We introduce two algorithms for searching the most probable tree for a given sentence (Section 4). The experimental results on English and Czech corpora are presented in Section 6. We conclude in Section 7 and offer some directions for future work.

2 Latent Tree Language Model

In this section we describe Latent Tree Language Model (LTLM). LTLM is a generative statistical model that discovers the tree structures hidden in the text corpus.

Let $\boldsymbol{L}$ be a word vocabulary with total of $|\boldsymbol{L}|$ distinct words. Assume we have a training corpus $\boldsymbol{w}$ divided into S sentences. The goal of LTLM or other LMs is to estimate the probability of a text $P(\boldsymbol{w})$. Let N_s denote the number of words in the s-th sentence. The s-th sentence is a sequence of words $\boldsymbol{w}_s = \{w_{s,i}\}_{i=0}^{N_s}$, where $w_{s,i} \in \boldsymbol{L}$ is a word at position i in this sentence and $w_{s,0} = <\mathrm{s}>$ is an artificial symbol that is added at the beginning of each sentence.

Each sentence s is associated with the dependency graph $\boldsymbol{G}_s$. We define the dependency graph as a labeled directed graph, where nodes correspond to the words in the sentence and there is a label for each node that we call *role*. Formally, it is a triple $\boldsymbol{G}_s = (\boldsymbol{V}_s, \boldsymbol{E}_s, \boldsymbol{r}_s)$ consisting of:

- The set of nodes $\boldsymbol{V}_s = \{0, 1, ..., N_s\}$. Each token $w_{s,i}$ is associated with node $i \in \boldsymbol{V}_s$.
- The set of edges $\boldsymbol{E}_s \subseteq \boldsymbol{V}_s \times \boldsymbol{V}_s$.
- The sequence of roles $\boldsymbol{r}_s = \{r_{s,i}\}_{i=0}^{N_s}$, where $1 \leq r_{s,i} \leq K$ for $i \in \boldsymbol{V}_s$. K is the number of roles.

The artificial word $w_{s,0} = <\mathrm{s}>$ at the beginning of the sentence has always role 1 ($r_{s,0} = 1$). Analogously to $\boldsymbol{w}$, the sequence of all $\boldsymbol{r}_s$ is denoted as $\boldsymbol{r}$ and sequence of all $\boldsymbol{G}_s$ as $\boldsymbol{G}$.

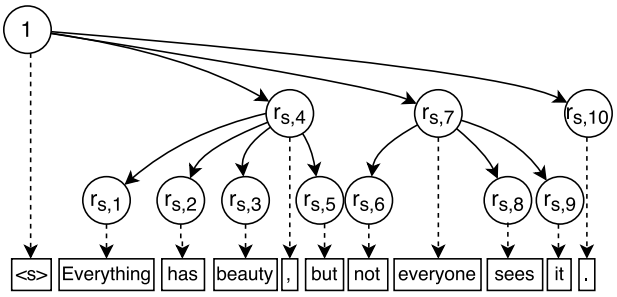


Figure 1: Example of LTLM for the sentence "*Everything has beauty, but not everyone sees it.*"

Edge $e \in \boldsymbol{E}_s$ is an ordered pair of nodes (i, j). We say that i is the *head* or the *parent* and j is the *dependent* or the *child*. We use the notation $i \rightarrow j$ for such edge. The directed path from node i to node j is denoted as $i \overset{*}{\rightarrow} j$.

We place a few constraints on the graph $\boldsymbol{G}_s$.

- The graph $\boldsymbol{G}_s$ is a *tree*. It means it is the acyclic graph (if $i \rightarrow j$ then not $j \overset{*}{\rightarrow} i$), where each node has one parent (if $i \rightarrow j$ then not $k \rightarrow j$ for every $k \neq i$).
- The graph $\boldsymbol{G}_s$ is *projective* (there are no cross edges). For each edge (i, j) and for each k between i and j (i.e. $i < k < j$ or $i > k > j$) there must exist the directed path $i \overset{*}{\rightarrow} k$.
- The graph $\boldsymbol{G}_s$ is always rooted in the node 0.

We denote these graphs as the *projective dependency trees*. Example of such a tree is on Figure 1. For the tree $\boldsymbol{G}_s$ we define a function

$$h_s(j) = i, \quad \text{when } (i, j) \in \boldsymbol{E}_s \tag{1}$$

that returns the parent for each node except the root.

We use graph $\boldsymbol{G}_s$ as a representation of the *Bayesian network* with random variables $\boldsymbol{E}_s$ and $\boldsymbol{r}_s$. The roles $r_{s,i}$ represent the node labels and the edges express the dependences between the roles. The conditional probability of the role at position i given its parent role is denoted as $P(r_{s,i}|r_{s,h_s(i)})$. The conditional probability of the word at position i in the sentence given its role $r_{s,i}$ is denoted as $P(w_{s,i}|r_{s,i})$.

We model the distribution over words in the sentence s as the mixture

$$P(\boldsymbol{w}_s) = P(\boldsymbol{w}_s|r_{s,0}) = \prod_{i=1}^{N_s} \sum_{k=1}^{K} P(w_{s,i}|r_{s,i} = k) P(r_{s,i} = k|r_{s,h_s(i)}). \tag{2}$$

The root role is kept fixed for each sentence ($r_{s,0}$ = 1) so $P(\boldsymbol{w}_s) = P(\boldsymbol{w}_s|r_{s,0})$.

We look at the roles as mixtures over child roles and simultaneously as mixtures over words. We can represent dependency between roles with a set of K multinomial distributions $\boldsymbol{\theta}$ over K roles, such that $P(r_{s,i}|r_{s,h_s(i)} = k) = \theta^{(k)}_{r_{s,i}}$. Simultaneously, dependency of words on their roles can be represented as a set of K multinomial distributions $\boldsymbol{\phi}$ over $|\boldsymbol{L}|$ words, such that $P(w_{s,i}|r_{s,i} = k) = \phi^{(k)}_{w_{s,i}}$. To make predictions about new sentences, we need to assume a prior distribution on the parameters $\boldsymbol{\theta}^{(k)}$ and $\boldsymbol{\phi}^{(k)}$.

We place a Dirichlet prior D with the vector of K hyper-parameters $\boldsymbol{\alpha}$ on a multinomial distribution $\boldsymbol{\theta}^{(k)} \sim D(\boldsymbol{\alpha})$ and with the vector of $|\boldsymbol{L}|$ hyper-parameters $\boldsymbol{\beta}$ on a multinomial distribution $\boldsymbol{\phi}^{(k)} \sim D(\boldsymbol{\beta})$. In general, D is not restricted to be Dirichlet distribution. It could be any distribution over discrete children, such as logistic normal. In this paper, we focus only on Dirichlet as a conjugate prior to the multinomial distribution and derive the learning algorithm under this assumption.

The choice of the child role depends only on its parent role, i.e. child roles with the same parent are mutually independent. This property is especially important for the learning algorithm (Section 3) and also for searching the most probable trees (Section 4). We do not place any assumption on the length of the sentence N_s or on how many children the parent node is expected to have.

3 Parameter Estimation

In this section we present the learning algorithm for LTLM. The goal is to estimate $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ in a way that maximizes the predictive ability of the model (generates the corpus with maximal joint probability $P(\boldsymbol{w})$).

Let $\chi^k_{(i,j)}$ be an operation that changes the tree $\boldsymbol{G}_s$ to $\boldsymbol{G}'_s$

$$\chi^k_{(i,j)} : \boldsymbol{G}_s \rightarrow \boldsymbol{G}'_s, \tag{3}$$

such that the newly created tree $\boldsymbol{G}'(\boldsymbol{V}'_s, \boldsymbol{E}'_s, \boldsymbol{r}'_s)$ consists of:

- $\boldsymbol{V}'_s = \boldsymbol{V}_s$.
- $\boldsymbol{E}'_s = (\boldsymbol{E}_s \setminus \{(h_s(i), i)\}) \cup \{(j, i)\}$.
- $r'_{s,a} = \begin{cases} r_{s,a} & \text{for } a \neq i \\ k & \text{for } a = i \end{cases}$, where $0 \leq a \leq N_s$.

It means that we change the role of the selected node i so that $r_{s,i} = k$ and simultaneously we change the parent of this node to be j. We call this operation a *partial change*.

The newly created graph $\boldsymbol{G}'$ must satisfy all conditions presented in Section 2, i.e. it is a projective dependency tree rooted in the node 0. Thus not all partial changes $\chi^k_{(i,j)}$ are possible to perform on graph $\boldsymbol{G}_s$.

Clearly, for the sentence s there is at most $\frac{N_s(1+N_s)}{2}$ parent changes[1].

To estimate the parameters of LTLM we apply Gibbs sampling and gradually sample $\chi^k_{(i,j)}$ for trees $\boldsymbol{G}_s$. For doing so we need to determine the posterior predictive distribution[2]

$$\boldsymbol{G}'_s \sim P(\chi^k_{(i,j)}(\boldsymbol{G}_s)|\boldsymbol{w}, \boldsymbol{G}), \tag{4}$$

from which we will sample partial changes to update the trees. In the equation, $\boldsymbol{G}$ denote the sequence of all trees for given sentences $\boldsymbol{w}$ and $\boldsymbol{G}'_s$ is a result of one sampling. In the following text we derive this equation under assumptions from Section 2.

The posterior predictive distribution of Dirichlet multinomial has the form of additive smoothing that is well known in the context of language modeling. The hyper-parameters of Dirichlet prior determine how much is the predictive distribution smoothed. Thus the predictive distribution for the word-in-role distribution can be expressed as

$$P(w_{s,i}|r_{s,i}, \boldsymbol{w}_{\setminus s,i}, \boldsymbol{r}_{\setminus s,i}) = \frac{n^{(w_{s,i}|r_{s,i})}_{\setminus s,i} + \beta}{n^{(\bullet|r_{s,i})}_{\setminus s,i} + |\boldsymbol{L}|\,\beta}, \tag{5}$$

where $n^{(w_{s,i}|r_{s,i})}_{\setminus s,i}$ is the number of times the role $r_{s,i}$ has been assigned to the word $w_{s,i}$, excluding the position i in the s-th sentence. The symbol $\bullet$ represents any word in the vocabulary so that $n^{(\bullet|r_{s,i})}_{\setminus s,i} = \sum_{l \in \boldsymbol{L}} n^{(l|r_{s,i})}_{\setminus s,i}$. We use the symmetric Dirichlet distribution for the word-in-role probabilities as it could be difficult to estimate the vector of hyper-parameters $\boldsymbol{\beta}$ for large word vocabulary. In the above mentioned equation, β is a scalar.

The predictive distribution for the role-by-role distribution is

$$P\left(r_{s,i}|r_{s,h_s(i)}, \boldsymbol{r}_{\setminus s,i}\right) = \frac{n^{(r_{s,i}|r_{s,h_s(i)})}_{\setminus s,i} + \alpha_{r_{s,i}}}{n^{(\bullet|r_{s,h_s(i)})}_{\setminus s,i} + \sum_{k=1}^{K} \alpha_k}. \tag{6}$$

Analogously to the previous equation, $n^{(r_{s,i}|r_{s,h_s(i)})}_{\setminus s,i}$ denote the number of times the role $r_{s,i}$ has the parent role $r_{s,h_s(i)}$, excluding the position i in the s-th sentence. The symbol $\bullet$ represents any possible role to make the probability distribution summing up to 1. We assume an asymmetric Dirichlet distribution.

We can use predictive distributions of above mentioned Dirichlet multinomials to express the joint probability that the role at position i is k ($r_{s,i} = k$) with parent at position j conditioned on current values of all variables, except those in position i in the sentence s

$$\begin{aligned} P(r_{s,i} = k, j|\boldsymbol{w}, \boldsymbol{r}_{\setminus s,i}) \propto \\ & P(w_{s,i}|r_{s,i} = k, \boldsymbol{w}_{\setminus s,i}, \boldsymbol{r}_{\setminus s,i}) \\ \times \quad & P(r_{s,i} = k|r_{s,j}, \boldsymbol{r}_{\setminus s,i}) \\ \times \quad & \prod_{a:h_s(a)=i} P(r_{s,a}|r_{s,i} = k, \boldsymbol{r}_{\setminus s,i}). \end{aligned} \tag{7}$$

The choice of the node i role affects the word that is produced by this role and also all the child roles of the node i. Simultaneously, the role of the node i depends on its parent j role. Formula 7 is derived from the joint probability of a sentence s and a tree $\boldsymbol{G}_s$, where all probabilities which do not depend on the choice of the role at position i are removed and equality is replaced by proportionality ($\propto$).

We express the final predictive distribution for sampling partial changes $\chi^k_{(i,j)}$ as

[1]The most parent changes are possible for the special case of the tree, where each node i has parent $i-1$. Thus for each node i we can change its parent to any node $j < i$ and keep the projectivity of the tree. That is $\frac{N_s(1+N_s)}{2}$ possibilities.

[2]The posterior predictive distribution is the distribution of an unobserved variable conditioned by the observed data, i.e. $P(X_{n+1}|X_1, ..., X_n)$, where X_i are i.i.d. (independent and identically distributed random variables).

$$P(\chi^k_{(i,j)}(\boldsymbol{G}_s)|\boldsymbol{w},\boldsymbol{G}) \propto \frac{P(r_{s,i}=k,j|\boldsymbol{w},\boldsymbol{r}_{\setminus s,i})}{P(r_{s,i},h_s(i)|\boldsymbol{w},\boldsymbol{r}_{\setminus s,i})} \tag{8}$$

that is essentially the fraction between the joint probability of $r_{s,i}$ and its parent after the partial change and before the partial change (conditioned on all other variables). This fraction can be interpreted as the necessity to perform this partial change.

We investigate two strategies of sampling partial changes:

- **Per sentence**: We sample a single partial change according to Equation 8 for each sentence in the training corpus. It means during one pass through the corpus (one iteration) we perform S partial changes.
- **Per position**: We sample a partial change for each position in each sentence. We perform in total $N = \sum_{s=1}^{S} N_s$ partial changes during one pass. Note that the denominator in Equation 8 is constant for this strategy and can be removed.

We compare both training strategies in Section 6. After enough training iterations, we can estimate the conditional probabilities $\phi_l^{(k)}$ and $\theta_k^{(p)}$ from actual samples as

$$\phi_l^{(k)} \approx \frac{n^{(w_{s,i}=l|r_{s,i}=k)} + \beta}{n^{(\bullet|r_{s,i}=k)} + |\boldsymbol{L}|\,\beta} \tag{9}$$

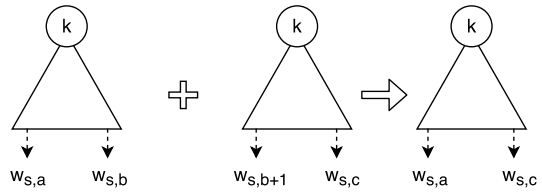
$$\theta_k^{(p)} \approx \frac{n^{(r_{s,i}=k|r_{s,h_s(i)}=p)} + \alpha_k}{n^{(\bullet|r_{s,h_s(i)}=p)} + \sum_{m=1}^{K} \alpha_m}. \tag{10}$$

These equations are similar to equations 5 and 6, but here the counts n do not exclude any position in a corpus.

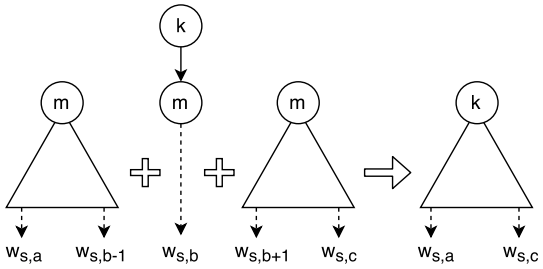
Note that in the Gibbs sampling equation, we assume that the Dirichlet parameters $\boldsymbol{\alpha}$ and β are given. We use a fixed point iteration technique described in (Minka, 2003) to estimate them.

4 Inference

In this section we present two approaches for searching the most probable tree for a given sentence assuming we have already estimated the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.



(a) The root has two or more children.



(b) The root has only one child.

Figure 2: Searching the most probable subtrees.

4.1 Non-deterministic Inference

We use the same sampling technique as for estimating parameters (Equation 8), i.e. we iteratively sample the partial changes $\chi^k_{(i,j)}$. However, we use equations 9 and 10 for predictive distributions of Dirichlet multinomials instead of 5 and 6. In fact, these equations correspond to the predictive distributions over the newly added word $w_{s,i}$ with the role $r_{s,i}$ into the corpus, conditioned on $\boldsymbol{w}$ and $\boldsymbol{r}$. This sampling technique rarely finds the best solution, but often it is very near.

4.2 Deterministic Inference

Here we present the deterministic algorithm that guarantees to find the most probable tree for a given sentence. We were inspired by Cocke-Younger-Kasami (CYK) algorithm (Lange and Leiß, 2009).

Let $\boldsymbol{T}^n_{s,a,c}$ denote the subtree of $\boldsymbol{G}_s$ (subgraph of $\boldsymbol{G}_s$ that is also a tree) containing subsequence of nodes $\{a, a+1, ..., c\}$. The superscript n denotes the number of children the root of this subtree has. We denote the joint probability of a subtree from position a to position c with the corresponding words conditioned by the root role k as $P^n(\{w_{s,i}\}_{i=a}^c, \boldsymbol{T}^n_{s,a,c}|k)$. Our goal is to find the tree $\boldsymbol{G}_s = \boldsymbol{T}^{1+}_{s,0,N_s}$ that maximizes probability $P(\boldsymbol{w}_s, \boldsymbol{G}_s) = P^{1+}(\{w_{s,i}\}_{i=0}^{N_s}, \boldsymbol{T}^{1+}_{s,0,N_s}|0)$.

Similarly to CYK algorithm, our approach fol-

lows bottom-up direction and goes through all possible subsequences for a sentence (sequence of words). At the beginning, the probabilities for subsequences of length 1 (i.e. single words) are calculated as $P^{1+}(\{w_{s,a}\}, \boldsymbol{T}^{1+}_{s,a,a}|k) = P(w_{s,a}|r_{s,a} = k)$. Once it has considered subsequences of length 1, it goes on to subsequences of length 2, and so on.

Thanks to mutual independence of roles under the same parent, we can find the most probable subtree with the root role k and with at least two root children according to

$$P^{2+}(\{w_{s,i}\}_{i=a}^{c}, \boldsymbol{T}^{2+}_{s,a,c}|k) = \max_{b:a<b<c} [P^{1+}(\{w_{s,i}\}_{i=a}^{b}, \boldsymbol{T}^{1+}_{s,a,b}|k) \times P^{1+}(\{w_{s,i}\}_{i=b+1}^{c}, \boldsymbol{T}^{1+}_{s,b+1,c}|k)]. \quad (11)$$

It means we merge two neighboring subtrees with the same root role k. This is the reason why the new subtree has at least two root children. This formula is visualized on Figure 2a. Unfortunately, this does not cover all subtree cases. We find the most probable tree with only root child as follows

$$P^{1}(\{w_{s,i}\}_{i=a}^{c}, \boldsymbol{T}^{1}_{s,a,c}|k) = \max_{b,m:a\le b\le c, 1\le m\le K} [P(w_{s,b}|r_{s,b} = m) \times P(r_{s,b} = m|k) \times P^{1+}(\{w_{s,i}\}_{i=a}^{b-1}, \boldsymbol{T}^{1+}_{s,a,b-1}|m) \times P^{1+}(\{w_{s,i}\}_{i=b+1}^{c}, \boldsymbol{T}^{1+}_{s,b+1,c}|m)]. \quad (12)$$

This formula is visualized on Figure 2b.

To find the most probable subtree no matter how many children the root has, we need to take the maximum from both mentioned equations $P^{1+} = max(P^{2+}, P^{1})$.

The algorithm has complexity $\mathcal{O}(N_s^3 K^2)$, i.e. it has cubic dependence on the length of the sentence N_s.

5 Side-dependent LTLM

Until now, we presented LTLM in its simplified version. In role-by-role probabilities (role conditioned on its parent role) we did not distinguish whether the role is on the left side or the right side of the parent. However, this position keeps important information about the syntax of words (and their roles).

We assume separate multinomial distributions $\dot{\boldsymbol{\theta}}$ for roles that are on the left and $\ddot{\boldsymbol{\theta}}$ for roles on the right. Each of them has its own Dirichlet prior with hyper-parameters $\dot{\boldsymbol{\alpha}}$ and $\ddot{\boldsymbol{\alpha}}$, respectively. The process of estimating LTLM parameters is almost the same. The only difference is that we need to redefine the predictive distribution for the role-by-role distribution (Equation 6) to include only counts of roles on the appropriate side. Also, every time the role-by-role probability is used we need to distinguish sides:

$$P(r_{s,i}|r_{s,h_s(i)}) = \begin{cases} \dot{\theta}^{(r_{s,h_s(i)})}_{r_{s,i}} & \text{for } i < h_s(i)) \\ \ddot{\theta}^{(r_{s,h_s(i)})}_{r_{s,i}} & \text{for } i > h_s(i)) \end{cases}. \quad (13)$$

In the following text we always assume the side-dependent LTLM.

6 Experimental Results and Discussion

In this section we present experiments with LTLM on two languages, English (EN) and Czech (CS).

As a training corpus we use CzEng 1.0 (Bojar et al., 2012) of the sentence-parallel Czech-English corpus. We choose this corpus because it contains multiple domains, it is of reasonable length, and it is parallel so we can easily provide comparison between both languages. The corpus is divided into 100 similarly-sized sections. We use parts 0–97 for training, the part 98 as a development set, and the last part 99 for testing.

We have removed all sentences longer than 30 words. The reason was that the complexity of the learning phase and the process of searching most probable trees depends on the length of sentences. It has led to removing approximately a quarter of all sentences. The corpus is available in a tokenized form so the only preprocessing step we use is lowercasing. We keep the vocabulary of 100,000 most frequent words in the corpus for both languages. The less frequent words were replaced by the symbol <unk>. Statistics for the final corpora are shown in Table 1.

We measure the quality of LTLM by *perplexity* that is the standard measure used for LMs. Perplexity is a measure of uncertainty. The lower perplexity means the better predictive ability of the LM.

Corpora	Sentences	Tokens	OOV rate
EN train	11,530,604	138,034,779	1.30%
EN develop.	117,735	1,407,210	1.28%
EN test	117,360	1,405,106	1.33%
CS train	11,832,388	133,022,572	3.98%
CS develop.	120,754	1,353,015	4.00%
CS test	120,573	1,357,717	4.03%

Table 1: Corpora statistics. *OOV rate* denotes the out-of-vocabulary rate.

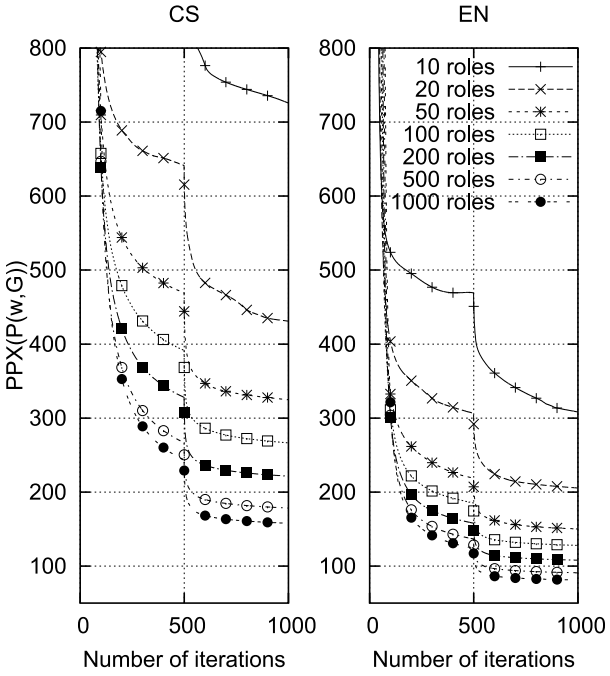


Figure 3: Learning curves of LTLM for both English and Czech. The points in the graphs represent the perplexities in every 100th iteration.

During the process of parameter estimation we measure the perplexity of joint probability of sentences and their trees defined as $\mathrm{PPX}(P(\boldsymbol{w}, \boldsymbol{G})) = \sqrt[N]{\frac{1}{P(\boldsymbol{w},\boldsymbol{G})}}$, where N is the number of all words in the training data $\boldsymbol{w}$.

As we describe in Section 3, there are two approaches for the parameter estimation of LTLM. During our experiments, we found that the per-position strategy of training has the ability to converge faster, but to a worse solution compared to the per-sentence strategy which converges slower, but to a better solution.

We train LTLM by 500 iterations of the per-position sampling followed by another 500 iterations of the per-sentence sampling. This proves to be efficient in both aspects, the reasonable speed of convergence and the satisfactory predictive ability of the model. The learning curves are showed on Figure 3. We present the models with 10, 20, 50, 100, 200, 500, and 1000 roles. The higher role cardinality models were not possible to create because of the very high computational requirements. Similarly to the training of LTLM, the non-deterministic inference uses 100 iterations of per-position sampling followed by 100 iterations of per-sentence sampling.

Model	EN	CS
2-gram MKN	165.9	272.0
3-gram MKN	67.7	99.3
4-gram MKN	**46.2**	73.5
300n RNNLM	51.2	**69.4**
4-gram LWLM	52.7	81.5
PoS STLM	455.7	747.3
1000r STLM	113.7	211.0
1000r det. LTLM	54.2	111.1
4-gram MKN + 300n RNNLM	36.8 (-20.4%)	49.5 (-32.7%)
4-gram MKN + 4-gram LWLM	41.5 (-10.2%)	62.4 (-15.1%)
4-gram MKN + PoS STLM	42.9 (-7.1%)	63.3 (-13.9%)
4-gram MKN + 1000r STLM	33.6 (-27.3%)	50.1 (-31.8%)
4-gram MKN + 1000r det. LTLM	**24.9 (-43.1%)**	**37.2 (-49.4%)**

Table 2: Perplexity results on the test data. The numbers in brackets are the relative improvements compared with standalone 4-gram MKN LM.

In the following experiments we measure how well LTLM generalizes the learned patterns, i.e. how well it works on the previously unseen data. Again, we measure the perplexity, but of probability $P(\boldsymbol{w})$ for mutual comparison with different LMs that are based on different architectures ($\mathrm{PPX}(P(\boldsymbol{w})) = \sqrt[N]{\frac{1}{P(\boldsymbol{w})}}$).

To show the strengths of LTLM we compare it with several state-of-the-art LMs. We experiment with Modified Kneser-Ney (MKN) interpolation (Chen and Goodman, 1998), with Recurrent Neural Network LM (RNNLM) (Mikolov et al., 2010; Mikolov et al., 2011)[3], and with LWLM (Deschacht et al., 2012)[4]. We have also created syntactic dependency tree based LM (denoted as STLM). Syntactic dependency trees for both languages are provided within CzEng corpus and are based on

[3]Implementation is available at `http://rnnlm.org/`. Size of the hidden layer was set to 300 in our experiments. It was computationally intractable to use more neurons.

[4]Implementation is available at `http://liir.cs.kuleuven.be/software.php`.

	EN							CS						
Model\roles	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**10**	**20**	**50**	**100**	**200**	**500**	**1000**
STLM	408.5	335.2	261.7	212.6	178.9	137.8	113.7	992.7	764.2	556.4	451.0	365.9	265.7	211.0
non-det. LTLM	329.5	215.1	160.4	126.5	105.6	86.7	78.4	851.0	536.6	367.4	292.6	235.2	186.1	157.6
det. LTLM	252.4	166.4	115.3	92.0	75.4	60.9	54.2	708.5	390.2	267.8	213.2	167.9	133.5	111.1
4-gram MKN + STLM	42.7	41.6	39.9	37.9	36.3	34.9	33.6	67.5	65.1	61.4	58.3	55.5	52.4	50.1
4-gram MKN + non-det. LTLM	41.1	38.0	35.2	32.7	30.7	28.9	27.8	65.8	59.4	55.1	51.1	47.5	43.7	41.3
4-gram MKN + det. LTLM	39.9	36.4	32.8	30.3	28.1	26.0	**24.9**	64.4	56.1	51.5	47.3	43.4	39.9	**37.2**

Table 3: Perplexity results on the test data for LTLMs and STLMs with different number of roles. Deterministic inference is denoted as *det.* and non-deterministic inference as *non-det.*

MST parser (McDonald et al., 2005). We use the same architecture as for LTLM and experiment with two approaches to represent the roles. Firstly, the roles are given by the part-of-speech tag (denoted as PoS STLM). No training is required, all information come from CzEng corpus. Secondly, we learn the roles using the same algorithm as for LTLM. The only difference is that the trees are kept unchanged. Note that both deterministic and non-deterministic inference perform almost the same in this model so we do not distinguish between them.

We combine baseline 4-gram MKN model with other models via linear combination (in the tables denoted by the symbol $+$) that is simple but very efficient technique to combine LMs. Final probability is then expressed as

$$P(\boldsymbol{w}) = \prod_{s=1}^{S} \prod_{i=1}^{N_s} \left[\lambda P^{\text{LM1}} + (\lambda - 1)\, P^{\text{LM2}}\right]. \quad (14)$$

In the case of MKN the probability P^{MKN} is the probability of a word $w_{s,i}$ conditioned by 3 previous words with MKN smoothing. For LTLM or STLM this probability is defined as

$$P^{\text{LTLM}}(w_{s,i}|r_{s,h_s(i)}) = \sum_{k=1}^{K} P(w_{s,i}|r_{s,i}=k) P(r_{s,i}=k|r_{s,h_s(i)}). \quad (15)$$

We use the *expectation maximization* algorithm (Dempster et al., 1977) for the maximum likelihood estimate of λ parameter on the development part of the corpus. The influence of the number of roles on the perplexity is shown in Table 3 and the final

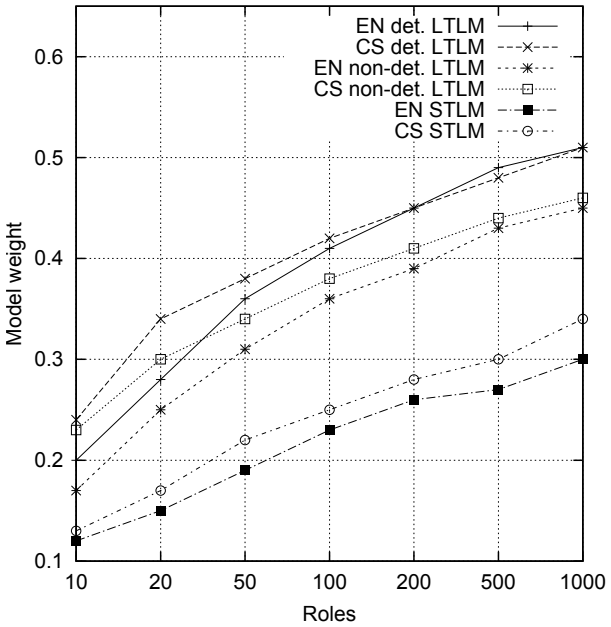


Figure 4: Model weights optimized on development data when interpolated with 4-gram MKN LM.

results are shown in Table 2. Note that these perplexities are not comparable with those on Figure 3 ($\text{PPX}(P(\boldsymbol{w}))$ vs. $\text{PPX}(P(\boldsymbol{w}, \boldsymbol{G}))$). Weights of LTLM and STLM when interpolated with MKN LM are shown on Figure 4.

From the tables we can see several important findings. Standalone LTLM performs worse than MKN on both languages, however their combination leads to dramatic improvements compared with other LMs. Best results are achieved by 4-gram MKN interpolated with 1000 roles LTLM and the deterministic inference. The perplexity was improved by approximately 46% on English and 49% on Czech compared with standalone MKN. The deterministic inference outperformed the non-deterministic one in all cases. LTLM also signifi-

everything	**has**	**beauty**	**,**	**but**	**not**	**everyone**	**sees**	**it**	**.**
it	's	one	,	but	was	he	saw	him	.
that	is	thing	;	course	it	i	made	it	!
let	was	life	–	though	not	she	found	her	...
there	knows	name	-	or	this	they	took	them	'
something	really	father	...	perhaps	that	that	gave	his	what
nothing	says	mother	:	and	the	it	told	me	"
everything	comes	way		maybe	now	who	felt	a	how
here	does	wife	(	although	had	you	thought	out	why
someone	gets	place	?	yet	<unk>	someone	knew	that	–
god	has	idea	naught	except	all	which	heard	himself	-

Table 4: Ten most probable word substitutions on each position in the sentence "*Everything has beauty, but not everyone sees it.*" produced by 1000 roles LTLM with the deterministic inference.

cantly outperformed STLM where the syntactic dependency trees were provided as a prior knowledge. The joint learning of syntax and semantics of a sentence proved to be more suitable for predicting the words.

An in-depth analysis of semantic and syntactic properties of LTLM is beyond the scope of this paper. For better insight into the behavior of LTLM, we show the most probable word substitutions for one selected sentence (see Table 4). We can see that the original words are often on the front positions. Also it seems that LTLM is more syntactically oriented, which confirms claims from (Levy and Goldberg, 2014; Padó and Lapata, 2007), but to draw such conclusions a deeper analysis is required. The properties of the model strongly depends on the number of distinct roles. We experimented with maximally 1000 roles. To catch the meaning of various words in natural language, more roles may be needed. However, with our current implementation, it was intractable to train LTLM with more roles in a reasonable time. Training 1000 roles LTLM took up to two weeks on a powerful computational unit.

7 Conclusion and Future Work

In this paper we introduced the Latent Tree Language Model. Our model discovers the latent tree structures hidden in natural text and uses them to predict the words in a sentence. Our experiments with English and Czech corpora showed dramatic improvements in the predictive ability compared with standalone Modified Kneser-Ney LM. Our Java implementation is available for research purposes at `https://github.com/brychcin/LTLM`.

It was beyond the scope of this paper to explicitly test the semantic and syntactic properties of the model. As the main direction for future work we plan to investigate these properties for example by comparison with human-assigned judgments. Also, we want to test our model in different NLP tasks (e.g. speech recognition, machine translation, etc.).

We think that the role-by-role distribution should depend on the distance between the parent and the child, but our preliminary experiments were not met with success. We plan to elaborate on this assumption. Another idea we want to explore is to use different distributions as a prior to multinomials. For example, Blei and Lafferty (2006) showed that the logistic-normal distribution works well for topic modeling because it captures the correlations between topics. The same idea might work for roles.

Acknowledgments

This publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures". Lastly, we would like to thank the anonymous reviewers for their insightful feedback.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic lan-

guage model. *Journal of Machine Learning Research*, 3:1137–1155, March.

David M. Blei and John D. Lafferty. 2006. Correlated topic models. In *In Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120. MIT Press.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. 2012. The joy of parallelism with czeng 1.0. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Tomáš Brychcín and Miloslav Konopík. 2014. Semantic spaces for improving language modeling. *Computer Speech & Language*, 28(1):192–209.

Tomáš Brychcín and Miloslav Konopík. 2015. Latent semantics in language models. *Computer Speech & Language*, 33(1):88–108.

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2009. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems 21*, pages 1–8.

Arthur P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38.

Koen Deschacht, Jan De Belder, and Marie-Francine Moens. 2012. The latent words language model. *Computer Speech & Language*, 26(5):384–409.

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

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June. Association for Computational Linguistics.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127.

Martin Lange and Hans Leiß. 2009. To cnf or not to cnf? an efficient yet presentable version of the cyk algorithm. *Informatica Didactica*, 8.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5528–5531, Prague Congress Center, Prague, Czech Republic.

Thomas P. Minka. 2003. Estimating a dirichlet distribution. Technical report.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, June.

Martin Popel and David Mareček. 2010. Perplexity of n-gram and dependency language models. In *Proceedings of the 13th International Conference on Text, Speech and Dialogue*, TSD'10, pages 173–180, Berlin, Heidelberg. Springer-Verlag.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010. Viterbi training

improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden, July. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Edward W. D. Whittaker and Philip C. Woodland. 2003. Language modelling for russian and english using words and classes. *Computer Speech & Language*, 17(1):87–104.

Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics

Douwe Kiela, Anita L. Verő and Stephen Clark
Computer Laboratory
University of Cambridge
douwe.kiela,alv34,stephen.clark@cl.cam.ac.uk

Abstract

Multi-modal distributional models learn grounded representations for improved performance in semantics. Deep visual representations, learned using convolutional neural networks, have been shown to achieve particularly high performance. In this study, we systematically compare deep visual representation learning techniques, experimenting with three well-known network architectures. In addition, we explore the various data sources that can be used for retrieving relevant images, showing that images from search engines perform as well as, or better than, those from manually crafted resources such as ImageNet. Furthermore, we explore the optimal number of images and the multi-lingual applicability of multi-modal semantics. We hope that these findings can serve as a guide for future research in the field.

1 Introduction

Multi-modal distributional semantics addresses the fact that text-based semantic models, which represent word meanings as a distribution over other words (Turney and Pantel, 2010; Clark, 2015), suffer from the grounding problem (Harnad, 1990). Recent work has shown that this theoretical motivation can be successfully exploited for practical gain. Indeed, multi-modal representation learning leads to improvements over language-only models in a range of tasks, including modelling semantic similarity and relatedness (Bruni et al., 2014; Silberer and Lapata, 2014; Kiela and Bottou, 2014; Lazaridou et al., 2015), improving lexical entailment (Kiela et al., 2015a), predicting compositionality (Roller and Schulte im Walde, 2013), bilingual lexicon induction (Bergsma and Van Durme, 2011), selectional preference prediction (Bergsma and Goebel, 2011), linguistic ambiguity resolution (Berzak et al., 2015), visual information retrieval (Bulat et al., 2016) and metaphor identification (Shutova et al., 2016).

Most multi-modal semantic models tend to rely on raw images as the source of perceptual input. Many data sources have been tried, ranging from image search engines to photo sharing websites to manually crafted resources. Images are retrieved for a given target word if they are ranked highly, have been tagged, or are otherwise associated with the target word(s) in the data source.

Traditionally, representations for images were learned through bag-of-visual words (Sivic and Zisserman, 2003), using SIFT-based local feature descriptors (Lowe, 2004). Kiela and Bottou (2014) showed that transferring representations from deep convolutional neural networks (ConvNets) yield much better performance than bag-of-visual-words in multi-modal semantics. ConvNets (LeCun et al., 1998) have become very popular in recent years: they are now the dominant approach for almost all recognition and detection tasks in the computer vision community (LeCun et al., 2015), approaching or even exceeding human performance in some cases (Weyand et al., 2016). The work by Alex Krizhevsky et al. (2012), which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) in 2012, has played an important role in bringing convolutional

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 447–456,
Austin, Texas, November 1-5, 2016.

	AlexNet	GoogLeNet	VGGNet
ILSVRC winner	2012	2014	2015
Number of layers	7	22	19
Number of parameters	~60 million	~6.7 million	~144 million
Receptive field size	11×11	3×3	$1 \times 1, 3 \times 3, 5 \times 5$
Fully connected layers	Yes	No	Yes

Table 1: Network architectures. Layer counts only include layers with parameters.

networks (back) to prominence. A similar network was used by Kiela and Bottou (2014) to obtain high quality image embeddings for semantics.

This work aims to provide a systematic comparison of such deep visual representation learning techniques and data sources; i.e. we aim to answer the following open questions in multi-modal semantics:

- Does the improved performance over bag-of-visual-words extend to different convolutional network architectures, or is it specific to Krizhevsky's AlexNet? Do others work even better?
- How important is the source of images? Is there a difference between search engines and manually annotated data sources? Does the number of images obtained for each word matter?
- Do these findings extend to different languages beyond English?

We evaluate semantic representation quality through examining how well a system's similarity scores correlate with human similarity and relatedness judgments. We examine both the visual representations themselves as well as the multi-modal representations that fuse visual representations with linguistic input, in this case using middle fusion (i.e., concatenation). To the best of our knowledge, this work is the first to systematically compare these aspects of visual representation learning.

2 Architectures

We use the MMFeat toolkit[1] (Kiela, 2016) to obtain image representations for three different convolutional network architectures: AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015) and VGGNet (Simonyan and Zisserman, 2014). Image representations are turned into an overall word-level visual representation by either taking the mean or the elementwise maximum of the relevant image representations. All three networks are trained to maximize the multinomial logistic regression objective using mini-batch gradient descent with momentum:

$$-\sum_{i=1}^{D}\sum_{k=1}^{K}\mathbf{1}\{y^{(i)}=k\}\log\frac{\exp(\theta^{(k)\top}x^{(i)})}{\sum_{j=1}^{K}\exp(\theta^{(j)\top}x^{(i)})}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, $x^{(i)}$ and $y^{(i)}$ are the input and output, respectively. D is the number of training examples and K is the number of classes. The networks are trained on the ImageNet classification task and we transfer layers from the pre-trained network. See Table 1 for an overview. In this section, we describe the network architectures and their properties.

AlexNet The network by Krizhevsky (2012) introduces the following network architecture: first, there are five convolutional layers, followed by two fully-connected layers, where the final layer is fed into a softmax which produces a distribution over the class labels. All layers apply rectified linear units (ReLUs) (Nair and Hinton, 2010) and use dropout for regularization (Hinton et al., 2012). This network won the ILSVRC 2012 ImageNet classification challenge. In our case, we actually use the CaffeNet reference model, which is a replication of AlexNet, with the difference that it is not trained with relighting data-augmentation, and that the order of pooling and normalization layers is switched (in CaffeNet, pooling is done before normalization,

[1] https://github.com/douwekiela/mmfeat

(a) ImageNet

(b) ESP Game dataset

(c) Bing

(d) Google

(e) Flickr

Figure 1: Example images for *dog* and *golden retriever* from the various data sources. ImageNet has no images for dog, with images only at nodes lower in the hierarchy. ESP does not have images for the golden retriever tag.

instead of the other way around). While it uses an almost identical architecture, performance of CaffeNet is slightly better than the original AlexNet.

GoogLeNet The ILSVRC 2014 challenge winning GoogLeNet (Szegedy et al., 2015) uses "inception modules" as a network-in-network method (Lin et al., 2013) for enhancing model discriminability for local patches within the receptive field. It uses much smaller receptive fields and explicitly focuses on efficiency: while it is much deeper than AlexNet, it has fewer parameters. Its architecture consists of two convolutional layers, followed by inception layers that culminate into an average pooling layer that feeds into the softmax decision (so it has no fully connected layers). Dropout is only applied on the final layer. All connections use rectifiers.

VGGNet The ILSVRC 2015 ImageNet classification challenge was won by VGGNet (Simonyan and Zisserman, 2014). Like GoogLeNet, it is much deeper than AlexNet and uses smaller receptive fields. It has many more parameters than the other networks. It consists of a series of convolutional layers followed by the fully connected ones. All layers are rectified and dropout is applied to the first two fully connected layers.

These networks were selected because they are very well-known in the computer vision community. They exhibit interesting qualitative differences in terms of their depth (i.e., the number of layers), the number of parameters, regularization methods and the use of fully connected layers. They have all been winning network architectures in the ILSVRC ImageNet classification challenges.

3 Sources of Image Data

Some systematic studies of parameters for text-based distributional methods have found that the source corpus has a large impact on representational quality (Bullinaria and Levy, 2007; Kiela and Clark, 2014). The same is likely to hold in the case of

	Google	Bing	Flickr	ImageNet	ESP Game
Type	Search engine	Search engine	Photo sharing	Image database	Game
Annotation	Automatic	Automatic	Human	Human	Human
Coverage	Unlimited	Unlimited	Unlimited	Limited	Limited
Multi-lingual	Yes	Yes	No	No	No
Sorted	Yes	Yes	Yes	No	No
Tag specificity	Unknown	Unknown	Loose	Specific	Loose

Table 2: Sources of image data.

visual representations. Various sources of image data have been used in multi-modal semantics, but there have not been many comparisons: Bergsma and Goebel (2011) compare Google and Flickr, and Kiela and Bottou (2014) compare ImageNet (Deng et al., 2009) and the ESP Game dataset (von Ahn and Dabbish, 2004), but most works use a single data source. In this study, one of our objectives is to asses the quality of various sources of image data. Table 2 provides an overview of the data sources, and Figure 1 shows some example images. We examine the following corpora:

Google Images Google's image search[2] results have been found to be comparable to hand-crafted image datasets (Fergus et al., 2005).

Bing Images An alternative image search engine is Bing Images[3]. It uses different underlying technology from Google Images, but offers the same functionality as an image search engine.

Flickr Although Bergsma and Goebel (2011) have found that Google Images works better in one experiment, the photo sharing service Flickr[4] is an interesting data source because its images are tagged by human annotators.

ImageNet ImageNet (Deng et al., 2009) is a large ontology of images developed for a variety of computer vision applications. It serves as a benchmarking standard for various image processing and computer vision tasks. ImageNet is constructed along the same hierarchical structure as WordNet (Miller, 1995), by attaching images to the corresponding synset (synonym set).

	MEN (3000)	SimLex (999)
Google	3000	999
Bing	3000	999
Flickr	3000	999
ImageNet	1326	373
ESPGame	2927	833
Common subset	1310	360

Table 3: Coverage on MEN and SimLex for our data sources.

ESP Game The ESP Game dataset (von Ahn and Dabbish, 2004) was constructed through a so-called "game with a purpose". Players were matched online and had to agree on an appropriate word label for a randomly selected image within a time limit. Once a word has been mentioned a certain number of times, that word becomes a taboo word and can no longer be used as a label.

These data sources have interesting qualitative differences. Online services return images for almost any query, with much better coverage than the fixed-size ImageNet and ESP Game datasets. Search engines annotate automatically, while the others are human-annotated, either through a strict annotation procedure in the case of ImageNet, or by letting users tag images, as in the case of Flickr and ESP. Automatic systems sort images by relevance, while the others are unsorted. The relevance ranking method is not accessible, however, and so has to be treated as a black box. Search results can be

[2]https://images.google.com/
[3]https://www.bing.com/images
[4]https://www.flickr.com

	Arch.	AlexNet				GoogLeNet				VGGNet			
	Agg.	Mean		Max		Mean		Max		Mean		Max	
Source	Type/Eval	SL	MEN	SL	MEN	SL	MEN	SL	MEN	SL	MEN	SL	MEN
Wikipedia	Text	.310	.682	.310	.682	.310	.682	.310	.682	.310	.682	.310	.682
Google	Visual	.340	.503	.334	.513	.358	.495	.367	.501	.342	.512	.332	.494
	MM	.380	.711	.370	.719	.379	.711	.365	.716	.380	.714	.365	.716
Bing	Visual	.325	.567	.316	.554	.310	.526	.303	.520	.304	.551	.289	.507
	MM	.373	.727	.360	.725	.364	.723	.350	.724	.361	.727	.349	.719
Flickr	Visual	.234	.483	.224	.441	.238	.407	.236	.385	.243	.460	.226	.385
	MM	.350	.715	.343	.711	.347	.689	.344	.703	.354	.702	.339	.696
ImageNet	Visual	.313	.561	.313	.561	.341	.540	.411	.603	.404	.584	.401	.578
	MM	.362	.713	.362	.713	.373	.719	.401	.731	.427	.727	.412	.723
ESPGame	Visual	.018	.448	.026	.376	.063	.487	.050	.434	.125	.506	.106	.451
	MM	.208	.686	.187	.672	.243	.700	.246	.696	.269	.708	.260	.698

Table 4: Performance on maximally covered datasets.

language-specific, while the human annotated data sources are restricted to English. Google and Bing will return images that were ranked highly, while Flickr contains photos rather than just any kind of image. ImageNet contains high-quality images descriptive of a given synset, meaning that the tagged object is likely to be centered in the image, while the ESP Game and Flickr images may have tags describing events happening in the background also.

3.1 Selecting and processing images

Selecting images for Google, Bing and Flickr is straightforward: using their respective APIs, the desired word is given as the search query and we obtain the top N returned images (unless otherwise indicated, we use N=10). In the case of ImageNet and ESP, images are not ranked and vary greatly in number: for some words there is only a single image, while others have thousands. With ImageNet, we are faced with the additional problem that images tend to be associated only with leaf nodes in the hierarchy. For example, *dog* has no directly associated images, while its hyponyms (e.g. *golden retriever*, *labrador*) have many. If a word has no associated images in its subtree, we try going up one level and seeing if the parent node's tree yields any images. We subsequently randomly sample 100 images associated with the word and obtain semi-ranked results by selecting the 10 images closest to the median representation as the sampled image representations. We use the same method for the ESP Game dataset. In all cases, images are resized and center-cropped to ensure that they are the correct size input.

4 Evaluation

Representation quality in semantics is usually evaluated using intrinsic datasets of human similarity and relatedness judgments. Model performance is assessed through the Spearman ρ_s rank correlation between the system's similarity scores for a given pair of words, together with human judgments. Here, we evaluate on two well-known similarity and relatedness judgment datasets: MEN (Bruni et al., 2012) and SimLex-999 (Hill et al., 2015). MEN focuses explicitly on relatedness (i.e. *coffee-tea* and *coffee-mug* get high scores, while *bakery-zebra* gets a low score), while SimLex-999 focuses on what it calls "genuine" similarity (i.e., *coffee-tea* gets a high score, while both *coffee-mug* and *bakery-zebra* get low scores). They are standard evaluations for evaluating representational quality in semantics.

In each experiment, we examine performance of the visual representations compared to text-based representations, as well as performance of the multimodal representation that fuses the two. In this

	Arch.	AlexNet				GoogLeNet				VGGNet			
	Agg.	Mean		Max		Mean		Max		Mean		Max	
Source	Type/Eval	SL	MEN	SL	MEN	SL	MEN	SL	MEN	SL	MEN	SL	MEN
Wikipedia	Text	.248	.654	.248	.654	.248	.654	.248	.654	.248	.654	.248	.654
Google	Visual	.406	.549	.402	.552	.420	.570	.434	.579	.430	.576	.406	.560
	MM	.366	.691	.344	.693	.366	.701	.342	.699	.378	.701	.341	.693
Bing	Visual	.431	.613	.425	.601	.410	.612	.414	.603	.400	.611	.398	.569
	MM	.384	.715	.355	.708	.374	.725	.343	.712	.363	.720	.340	.705
Flickr	Visual	.382	.577	.371	.544	.378	.547	.354	.518	.378	.567	.340	.511
	MM	.372	.725	.344	.712	.367	.728	.336	.716	.370	.726	.330	.711
ImageNet	Visual	.316	.560	.316	.560	.347	.538	.423	.600	.412	.581	.413	.574
	MM	.348	.711	.348	.711	.364	.717	.394	.729	.418	.724	.405	.721
ESPGame	Visual	.037	.431	.039	.347	.104	.501	.125	.438	.188	.514	.125	.460
	MM	.179	.666	.147	.651	.224	.692	.226	.683	.268	.697	.222	.688

Table 5: Performance on common coverage subsets of the datasets (MEN* and SimLex*).

case, we apply mid-level fusion, concatenating the L2-normalized representations (Bruni et al., 2014). Middle fusion is a popular technique in multi-modal semantics that has several benefits: 1) it allows for drawing from different data sources for each modality, that is, it does not require joint data; 2) concatenation is less susceptible to noise, since it preserves the information in the individual modalities; and 3) it is straightforward to apply and computationally inexpensive. Linguistic representations are 300-dimensional and are obtained by applying skip-gram with negative sampling (Mikolov et al., 2013) to a recent dump of Wikipedia. The normalization step that is performed before applying fusion ensures that both modalities contribute equally to the overall multi-modal representation.

5 Results

As Table 3 shows, the data sources vary in coverage: it would be unfair to compare data sources on the different subsets of the evaluation datasets that they have coverage for. That is, when comparing data sources we want to make sure we evaluate on images for the exact same word pairs. When comparing network architectures, however, we are less interested in the relative coverage between datasets and more interested in overall performance, in such a way that it can be compared to other work that was evaluated on the fully covered datasets. Hence, we report results on the maximally covered subsets per data source, which we refer to as MEN and SimLex, as well as for the overlapping common subset of word pairs that have images in each of the sources, which we refer to as MEN* and SimLex*.

5.1 Maximum coverage comparison

Table 4 shows the results on the maximally covered datasets. This means we cannot directly compare between data sources, because they have different coverage, but we can look at absolute performance and compare network architectures. The first row reports results for the text-based linguistic representations that were obtained from Wikipedia (repeated across columns for convenience). For each of the three architectures, we evaluate on SimLex (SL) and MEN, using either the mean (Mean) or elementwise maximum (Max) method for aggregating image representations into visual ones (see Section 2). For each data source, we report results for the visual representations, as well as for the multi-modal representations that fuse the visual and textual ones together. Performance across architectures is remarkably stable: we have had to report results up to three decimal points to show the difference in performance in some cases.

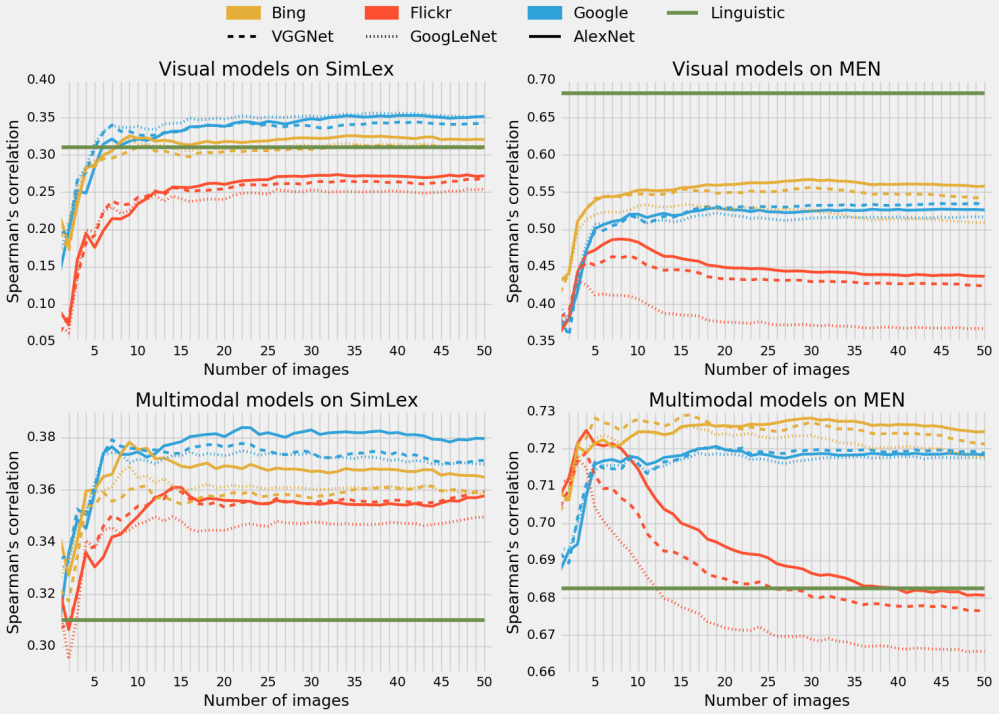


Figure 2: The effect of the number of images on representation quality.

For each of the network architectures, we see a marked improvement of multi-modal representations over uni-modal linguistic representations. In many cases, we also see visual representations outperforming linguistic ones, especially on SimLex. This is interesting, because e.g. Google and Bing have full coverage over the datasets, so their visual representations include highly abstract words, which does not appear to have an adverse impact on the method's performance. For the ESP Game dataset (on which performance is quite low) and ImageNet, we observe an increase in performance as we move to the right in the table. Interestingly, VGGNet on ImageNet scores very highly, which seems to indicate that VGGNet is somehow more "specialized" on ImageNet than the others. The difference between mean and max aggregation is relatively small, although the former seems to work better for SimLex while the latter does slightly better for MEN.

5.2 Common subset comparison

Table 5 shows the results on the common subset of the evaluation datasets, where all word pairs have images in each of the data sources. First, note the same patterns as before: multi-modal representations perform better than linguistic ones. Even for the poorly performing ESP Game dataset, the VGGNet representations perform better on both SimLex and MEN (bottom right of the table). Visual representations from Google, Bing, Flickr and ImageNet all perform much better than ESP Game on this common covered subset. In a sense, the full-coverage datasets were "punished" for their ability to return images for abstract words in the previous experiment: on this subset, which is more concrete, the search engines do much better. To a certain extent, including linguistic information is actually detrimental to performance, with multi-modal performing worse than purely visual. Again, we see the marked improvement with VGGNet for ImageNet, while Google, Bing and Flickr all do very well, regardless of the architecture.

These numbers indicate the robustness of the approach: we find that multi-modal representation learning yields better performance across the board: for different network architectures, different data sources and different aggregation methods. If computational efficiency or memory usage are issues, then GoogLeNet or AlexNet are the best choices. The ESP Game dataset does not appear to work very

well, and is best avoided. If we have the right coverage, then ImageNet gives good results, especially if we can use VGGNet. However, coverage is often the main issue, in which case Google and Bing yield images that are comparable or even better than images from the carefully annotated ImageNet.

5.3 Number of images

Another question is the number of images we want to use: does performance increase with more images? Is it always better to have seen 100 cats instead of only 10, or do we have enough information after having seen one or two already? There is an obvious trade-off here, since downloading and processing images takes time (and may incur financial costs). This experiment only applies to relevance-sorted data sources: the image selection procedure for ImageNet and ESPGame is more about removing outliers than about finding the best possible images.

As Figure 2 shows, it turns out that the optimal number of images stabilizes surprisingly quickly: around 10-20 images appears to be enough, and in some cases already too many. Performance across networks does not vary dramatically when using more images, but in the case of Flickr images on the MEN dataset, performance drops significantly as the number of images increases.

5.4 Multi- and cross-lingual applicability

Although there are some indicators that visual representation learning extends to other languages, particularly in the case of bilingual lexicon learning (Bergsma and Van Durme, 2011; Kiela et al., 2015b; Vulić et al., 2016), this has not been shown directly on the same set of human similarity and relatedness judgments. In order to examine the multi-lingual applicability of our findings, we train linguistic representations on recent dumps of the English and Italian Wikipedia. We then search for 10 images per word on Google and Bing, while setting the language to English or Italian. We compare the results on the original SimLex, and the Italian version from Leviant and Reichart (2015).

Similarly, we examine a cross-lingual scenario, where we translate Italian words into English using Google Translate. We then obtain images for the translated words and extract visual representations. These cross-lingual visual representations are subsequently evaluated on the Italian version of SimLex. Since we know that performance across architectures is similar, we use AlexNet representations.

		SimLex		
		EN	IT (M)	IT (C)
Wikipedia	Linguistic	.310	.179	.179
Google	Visual	.340	.231	.238
	Multi-modal	.380	.231	.227
Bing	Visual	.325	.212	.194
	Multi-modal	.373	.227	.207

Table 6: Performance on English and Italian SimLex, either in the multi-lingual setting (M) or the cross-lingual settting (C) where we first map to English.

The results can be found in Table 6. We find the same pattern: in all cases, visual and multi-modal representations outperform linguistic ones. The Italian version of SimLex appears to be more difficult than the English version. Google performs better than Bing, especially on the Italian evaluations. For Google, the cross-lingual scenario works better, while Bing yields better results in the multi-lingual setting where we use the language itself instead of mapping to English. Although somewhat preliminary, these results clearly indicate that multi-modal semantics can fruitfully be applied to languages other than English.

6 Conclusion and future work

The objective of this study has been to systematically compare network architectures and data sources for multi-modal systems. In particular, we focused on the capabilities of deep visual representations in capturing semantics, as measured by correlation with human similarity and relatedness judgments. Our findings can be summarized as follows:

- We examined AlexNet, GoogLeNet and VGGNet, all three recent winners of the ILSVRC ImageNet classification challenge (Russakovsky et al., 2015), and found that they perform very similarly. If efficiency or memory are issues, AlexNet or GoogLeNet are the most suitable architectures. For overall

best performance, AlexNet and VGGNet are the best choices.

- The choice of data sources appeared to have a bigger impact: Google, Bing, Flickr and ImageNet were much better than the ESP Game dataset. Google, Flickr and Bing have the advantage that they have potentially unlimited coverage. Google and Bing are particularly suited to full-coverage experiments, even when these include abstract words.

- We found that the number of images has an impact on performance, but that it stabilizes at around 10-20 images, indicating that it is usually not necessary to obtain more than 10 images per word. For Flickr, obtaining more images is detrimental to performance.

- Lastly, we established that these findings extend to other languages beyond English, obtaining the same findings on an Italian version of SimLex using the Italian Wikipedia. We examined both the multi-lingual setting where we obtain search results using the Italian language and a cross-lingual setting where we mapped Italian words to English and retrieved images for those.

This work answers several open questions in multi-modal semantics and we hope that it will serve as a guide for future research in the field. It is important to note that the multi-modal results only apply to the mid-level fusion method of concatenating normalized vectors: although these findings are indicative of performance for other fusion methods, different architectures or data sources may be more suitable for different fusion methods.

In future work, downstream tasks should be addressed: it is good that multi-modal semantics improves performance on intrinsic evaluations, but it is important to show its practical benefits in more applied tasks as well. Understanding what it is that makes these representations perform so well is another important and yet unanswered question. We hope that this work may be used as a reference in determining some of the choices that can be made when developing multi-modal models.

Acknowledgments

Anita Verő is supported by the Nuance Foundation Grant: Learning Type-Driven Distributed Representations of Language. Stephen Clark is supported by the ERC Starting Grant: DisCoTex (306920).

References

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *Proceedings of RANLP*, pages 399–405.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *IJCAI*, pages 1764–1769.

Yevgeni Berzak, Andrei Barbu, Daniel Harari, Boris Katz, and Shimon Ullman. 2015. Do you see what i mean? visual resolution of linguistic ambiguities. In *Proceedings of EMNLP*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, pages 136–145.

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

Luana Bulat, Douwe Kiela, and Stephen Clark. 2016. Vision and Feature Norms: Improving automatic feature norm learning through cross-modal maps. In *Proceedings of NAACL-HLT 2016*, San Diego, CA.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting Semantic Representations from Word Co-occurrence Statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Stephen Clark. 2015. Vector Space Models of Lexical Meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantic Theory*, chapter 16. Wiley-Blackwell, Oxford.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of CVPR*, pages 248–255.

Robert Fergus, Fei-Fei Li, Pietro Perona, and Andrew Zisserman. 2005. Learning object categories from Google's image search. In *Proceedings of ICCV*, pages 1816–1823.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D*, 42:335–346.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*, pages 36–45.

Douwe Kiela and Stephen Clark. 2014. A Systematic Study of Semantic Vector Space Model Parameters. In *Proceedings of EACL 2014, Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015a. Exploiting image generality for lexical entailment detection. In *Proceedings of ACL*, pages 119–124, Beijing, China, July. Association for Computational Linguistics.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015b. Visual bilingual lexicon induction with transferred convnet features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 148–158, Lisbon, Portugal, September. Association for Computational Linguistics.

Douwe Kiela. 2016. Mmfeat: A toolkit for extracting multi-modal features. In *Proceedings of ACL 2016*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1106–1114.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skipgram model. In *Proceedings of NAACL*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436—444.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *arXiv preprint arXiv:1508.00106*.

Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *CoRR*, abs/1312.4400.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, Scottsdale, Arizona, USA.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal LDA model integrating textual, cognitive and visual modalities. In *Proceedings of EMNLP*, pages 1146–1157.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of NAACL-HTL 2016*, San Diego. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of ACL*, pages 721–732.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Josef Sivic and Andrew Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *Proceedings of ICCV*, pages 1470–1477.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: vector space models of semantics. *Journal of Artifical Intelligence Research*, 37(1):141–188, January.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *CHI*, pages 319–326.

Ivan Vulić, Douwe Kiela, Marie-Francine Moens, and Stephen Clark. 2016. Multi-modal representations for improved bilingual lexicon learning. In *Proceedings of ACL*, Berlin, Germany. Association for Computational Linguistics.

Tobias Weyand, Ilya Kostrikov, and James Philbin. 2016. Planet - photo geolocation with convolutional neural networks. *CoRR*, abs/1602.05314.

Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding

Akira Fukui*[1,2] **Dong Huk Park***[1] **Daylen Yang***[1]
Anna Rohrbach*[1,3] **Trevor Darrell**[1] **Marcus Rohrbach**[1]
[1]UC Berkeley EECS, CA, United States
[2]Sony Corp., Tokyo, Japan
[3]Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

Modeling textual or visual information with vector representations trained from large language or visual datasets has been successfully explored in recent years. However, tasks such as visual question answering require combining these vector representations with each other. Approaches to multimodal pooling include element-wise product or sum, as well as concatenation of the visual and textual representations. We hypothesize that these methods are not as expressive as an outer product of the visual and textual vectors. As the outer product is typically infeasible due to its high dimensionality, we instead propose utilizing Multimodal Compact Bilinear pooling (MCB) to efficiently and expressively combine multimodal features. We extensively evaluate MCB on the visual question answering and grounding tasks. We consistently show the benefit of MCB over ablations without MCB. For visual question answering, we present an architecture which uses MCB twice, once for predicting attention over spatial features and again to combine the attended representation with the question representation. This model outperforms the state-of-the-art on the Visual7W dataset and the VQA challenge.

1 Introduction

Representation learning for text and images has been extensively studied in recent years. Recurrent neural networks (RNNs) are often used to represent sentences or phrases (Sutskever et al., 2014; Kiros et al., 2015), and convolutional neural networks (CNNs) have shown to work best to represent images (Donahue et al., 2013; He et al., 2015). For tasks such as visual question answering (VQA) and visual grounding, most approaches require joining the representation of both modalities. For combining the two vector representations (multimodal pooling), current approaches in VQA or grounding rely on concatenating vectors or applying element-wise sum or product. While this generates a joint representation, it might not be expressive enough to fully capture the complex associations between the two different modalities.

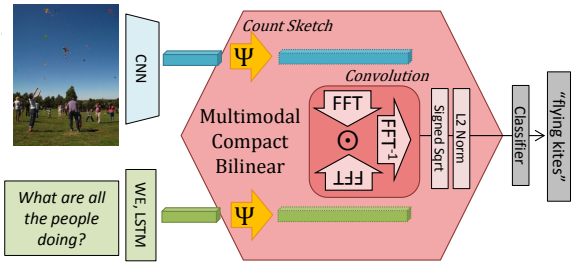


Figure 1: Multimodal Compact Bilinear Pooling for visual question answering.

In this paper, we propose to rely on Multimodal Compact Bilinear pooling (MCB) to get a joint representation. Bilinear pooling computes the outer product between two vectors, which allows, in contrast to element-wise product, a multiplicative interaction between all elements of both vectors. Bilinear pooling models (Tenenbaum and Freeman, 2000) have recently been shown to be beneficial for fine-grained classification for vision only tasks (Lin et al., 2015). However, given their high dimensionality (n^2), bilinear pooling has so far not been widely used. In

* indicates equal contribution

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 457–468,
Austin, Texas, November 1-5, 2016.

this paper, we adopt the idea from Gao et al. (2016) which shows how to efficiently compress bilinear pooling for a single modality. In this work, we discuss and extensively evaluate the extension to the multimodal case for text and visual modalities. As shown in Figure 1, Multimodal Compact Bilinear pooling (MCB) is approximated by randomly projecting the image and text representations to a higher dimensional space (using Count Sketch (Charikar et al., 2002)) and then convolving both vectors efficiently by using element-wise product in Fast Fourier Transform (FFT) space. We use MCB to predict answers for the VQA task and locations for the visual grounding task. For open-ended question answering, we present an architecture for VQA which uses MCB twice, once to predict spatial attention and the second time to predict the answer. For multiple-choice question answering we introduce a third MCB to relate the encoded answer to the question-image space. Additionally, we discuss the benefit of attention maps and additional training data for the VQA task. To summarize, MCB is evaluated on two tasks, four datasets, and with a diverse set of ablations and comparisons to the state-of-the-art.

2 Related Work

Multimodal pooling. Current approaches to multimodal pooling involve element-wise operations or vector concatenation. In the visual question answering domain, a number of models have been proposed. Simpler models such as iBOWIMG baseline (Zhou et al., 2015) use concatenation and fully connected layers to combine the image and question modalities. Stacked Attention Networks (Yang et al., 2015) and Spatial Memory Networks (Xu et al., 2015) use LSTMs or extract soft-attention on the image features, but ultimately use element-wise product or element-wise sum to merge modalities. D-NMN (Andreas et al., 2016a) introduced REINFORCE to dynamically create a network and use element-wise product to join attentions and element-wise sum predict answers. Dynamic Memory Networks (DMN) (Xiong et al., 2016) pool the image and question with element-wise product and sum, attending to part of the image and question with an Episodic Memory Module (Kumar et al., 2016). DPPnet (Noh et al., 2015) creates a Parameter Prediction Network which learns to predict the parameters of the second to last visual recognition layer dynamically from the question. Similar to this work, DPPnet allows multiplicative interactions between the visual and question encodings. Lu et al. (2016) recently proposed a model that extracts multiple co-attentions on the image and question and combines the co-attentions in a hierarchical manner using element-wise sum, concatenation, and fully connected layers.

For the visual grounding task, Rohrbach et al. (2016) propose an approach where the language phrase embedding is concatenated with the visual features in order to predict the attention weights over multiple bounding box proposals. Similarly, Hu et al. (2016a) concatenate phrase embeddings with visual features at different spatial locations to obtain a segmentation.

Bilinear pooling. Bilinear pooling has been applied to the fine-grained visual recognition task. Lin et al. (2015) use two CNNs to extract features from an image and combine the resulting vectors using an outer product, which is fully connected to an output layer. Gao et al. (2016) address the space and time complexity of bilinear features by viewing the bilinear transformation as a polynomial kernel. Pham and Pagh (2013) describe a method to approximate the polynomial kernel using Count Sketches and convolutions.

Joint multimodal embeddings. In order to model similarities between two modalities, many prior works have learned joint multimodal spaces, or embeddings. Some of such embeddings are based on Canonical Correlation Analysis (Hardoon et al., 2004) e.g. (Gong et al., 2014; Klein et al., 2015; Plummer et al., 2015), linear models with ranking loss (Frome et al., 2013; Karpathy and Fei-Fei, 2015; Socher et al., 2014; Weston et al., 2011) or non-linear deep learning models (Kiros et al., 2014; Mao et al., 2015; Ngiam et al., 2011). Our multimodal compact bilinear pooling can be seen as a complementary operation that allows us to capture different interactions between two modalities more expressively than e.g. concatenation. Consequently, many embedding learning approaches could benefit from incorporating such interactions.

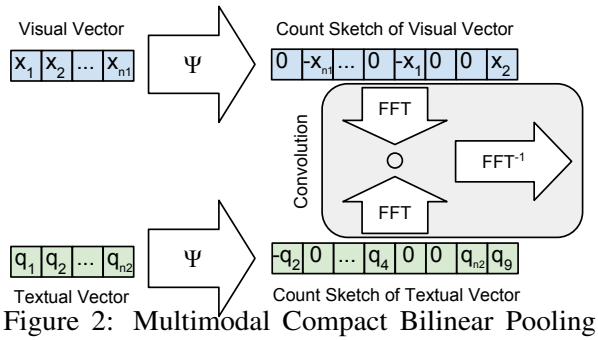


Figure 2: Multimodal Compact Bilinear Pooling (MCB)

3 Multimodal Compact Bilinear Pooling for Visual and Textual Embeddings

For the task of visual question answering (VQA) or visual grounding, we have to predict the most likely answer or location $\hat{a}$ for a given image $\mathbf{x}$ and question or phrase $\mathbf{q}$. This can be formulated as

$$\hat{a} = \underset{a \in A}{\operatorname{argmax}}\, p(a|\mathbf{x}, \mathbf{q}; \theta) \tag{1}$$

with parameters θ and the set of answers or locations A. For an image embedding $x = \Xi(\mathbf{x})$ (i.e. a CNN) and question embedding $q = \Omega(\mathbf{q})$ (i.e. an LSTM), we are interested in getting a good joint representation by pooling both representations. With a multimodal pooling $\Phi(x, q)$ that encodes the relationship between x and q well, it becomes easier to learn a classifier for Equation (1).

In this section, we first discuss our multimodal pooling Φ for combining representations from different modalities into a single representation (Sec. 3.1) and then detail our architectures for VQA (Sec. 3.2) and visual grounding (Sec. 3.3), further explaining how we predict $\hat{a}$ with the given image representation Ξ and text representation Ω.

3.1 Multimodal Compact Bilinear Pooling (MCB)

Bilinear models (Tenenbaum and Freeman, 2000) take the outer product of two vectors $x \in \mathbb{R}^{n_1}$ and $q \in \mathbb{R}^{n_2}$ and learn a model W (here linear), i.e. $z = W\,[x \otimes q]$, where $\otimes$ denotes the outer product (xq^T) and $[\,]$ denotes linearizing the matrix in a vector. As discussed in the introduction, bilinear pooling is interesting because it allows all elements of both vectors to interact with each other in a multiplicative way. However, the high dimensional representation (i.e. when n_1 and n_2 are large) leads to an infeasible number of parameters to learn in W. For example, we use $n_1 = n_2 = 2048$ and $z \in \mathbb{R}^{3000}$ for VQA. W thus would have 12.5 billion parameters, which leads to very high memory consumption and high computation times.

Algorithm 1 Multimodal Compact Bilinear

1: input: $v_1 \in \mathbb{R}^{n_1}, v_2 \in \mathbb{R}^{n_2}$
2: output: $\Phi(v_1, v_2) \in \mathbb{R}^d$
3: **procedure** MCB(v_1, v_2, n_1, n_2, d)
4: **for** $k \leftarrow 1 \ldots 2$ **do**
5: **if** h_k, s_k not initialized **then**
6: **for** $i \leftarrow 1 \ldots n_k$ **do**
7: sample $h_k[i]$ from $\{1, \ldots, d\}$
8: sample $s_k[i]$ from $\{-1, 1\}$
9: $v'_k = \Psi(v_k, h_k, s_k, n_k)$
10: $\Phi = \text{FFT}^{-1}(\text{FFT}(v'_1) \odot \text{FFT}(v'_2))$
11: **return** Φ
12: **procedure** $\Psi(v, h, s, n)$
13: $y = [0, \ldots, 0]$
14: **for** $i \leftarrow 1 \ldots n$ **do**
15: $y[h[i]] = y[h[i]] + s[i] \cdot v[i]$
16: **return** y

We thus need a method that projects the outer product to a lower dimensional space and also avoids computing the outer product directly. As suggested by Gao et al. (2016) for a single modality, we rely on the Count Sketch projection function Ψ (Charikar et al., 2002), which projects a vector $v \in \mathbb{R}^n$ to $y \in \mathbb{R}^d$. We initialize two vectors $s \in \{-1, 1\}^n$ and $h \in \{1, ..., d\}^n$: s contains either 1 or -1 for each index, and h maps each index i in the input v to an index j in the output y. Both s and h are initialized randomly from a uniform distribution and remain constant for future invocations of count sketch. y is initialized as a zero vector. For every element $v[i]$ its destination index $j = h[i]$ is looked up using h, and $s[i] \cdot v[i]$ is added to $y[j]$. See lines 1-9 and 12-16 in Algorithm 1.

This allows us to project the outer product to a lower dimensional space, which reduces the number of parameters in W. To avoid computing the outer product explicitly, Pham and Pagh (2013) showed that the count sketch of the outer product of two vectors can be expressed as convolution of both count sketches: $\Psi(x \otimes q, h, s) = \Psi(x, h, s) * \Psi(q, h, s)$,

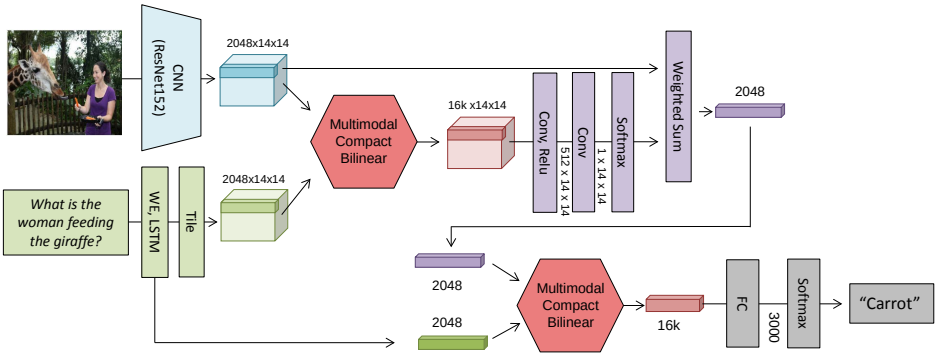


Figure 3: Our architecture for VQA: Multimodal Compact Bilinear (MCB) with Attention. Conv implies convolutional layers and FC implies fully connected layers. For details see Sec. 3.2.

where $*$ is the convolution operator. Additionally, the convolution theorem states that convolution in the time domain is equivalent to element-wise product in the frequency domain. The convolution $x' * q'$ can be rewritten as $\text{FFT}^{-1}(\text{FFT}(x') \odot \text{FFT}(q'))$, where $\odot$ refers to element-wise product. These ideas are summarized in Figure 2 and formalized in Algorithm 1, which is based on the Tensor Sketch algorithm of Pham and Pagh (2013). We invoke the algorithm with $v_1 = x$ and $v_2 = q$. We note that this easily extends and remains efficient for more than two multi-modal inputs as the combination happens as element-wise product.

3.2 Architectures for VQA

In VQA, the input to the model is an image and a question, and the goal is to answer the question. Our model extracts representations for the image and the question, pools the vectors using MCB, and arrives at the answer by treating the problem as a multi-class classification problem with 3,000 possible classes.

We extract image features using a 152-layer Residual Network (He et al., 2015) that is pretrained on ImageNet data (Deng et al., 2009). Images are resized to 448×448, and we use the output of the layer ("pool5") before the 1000-way classifier. We then perform L_2 normalization on the 2048-D vector.

Input questions are first tokenized into words, and the words are one-hot encoded and passed through a learned embedding layer. The tanh nonlinearity is used after the embedding. The embedding layer is followed by a 2-layer LSTM with 1024 units in each layer. The outputs of each LSTM layer are concatenated to form a 2048-D vector.

The two vectors are then passed through MCB. The MCB is followed by an element-wise signed square-root and L_2 normalization. After MCB pooling, a fully connected layer connects the resulting 16,000-D multimodal representation to the 3,000 top answers.

Attention. To incorporate spatial information, we use soft attention on our MCB pooling method. Explored by (Xu et al., 2015) for image captioning and by (Xu and Saenko, 2016) and (Yang et al., 2015) for VQA, the soft attention mechanism can be easily integrated in our model.

For each spatial grid location in the visual representation (i.e. last convolutional layer of ResNet [res5c], last convolutional layer of VGG [conv5]), we use MCB pooling to merge the slice of the visual feature with the language representation. As depicted in Figure 3, after the pooling we use two convolutional layers to predict the attention weight for each grid location. We apply softmax to produce a normalized soft attention map. We then take a weighted sum of the spatial vectors using the attention map to create the attended visual representation. We also experiment with generating multiple attention maps to allow the model to make multiple "glimpses" which are concatenated before being merged with the language representation through another MCB pooling for prediction. Predicting attention maps with MCB pooling allows the model to effectively learn how to attend to salient locations based on both the visual and language representations.

Answer Encoding. For VQA with multiple choices, we can additionally embed the answers. We

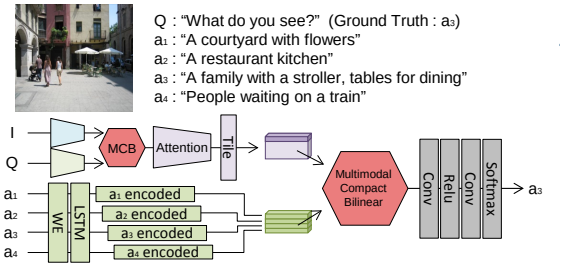


Figure 4: Our architecture for VQA: MCB with Attention and Answer Encoding

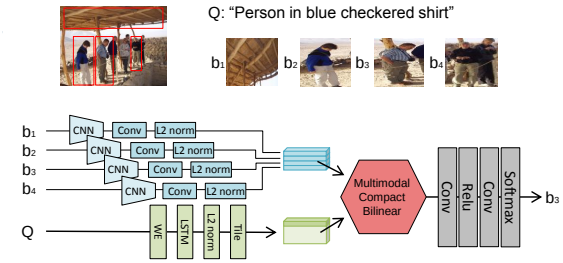


Figure 5: Our Architecture for Grounding with MCB (Sec. 3.3)

base our approach on the proposed MCB with attention. As can be seen from Figure 4, to deal with multiple variable-length answer choices, each choice is encoded using a word embedding and LSTM layers whose weights are shared across the candidates. In addition to using MCB with attention, we use an additional MCB pooling to merge the encoded answer choices with the multimodal representation of the original pipeline. The resulting embedding is projected to a classification vector with a dimension equal to the number of answers.

3.3 Architecture for Visual Grounding

We base our grounding approach on the fully-supervised version of GroundeR (Rohrbach et al., 2016). The overview of our model is shown in Figure 5. The input to the model is a query natural language phrase and an image along with multiple proposal bounding boxes. The goal is to predict a bounding box which corresponds to the query phrase. We replace the concatenation of the visual representation and the encoded phrase in GroundeR with MCB to combine both modalities. In contrast to Rohrbach et al. (2016), we include a linear embedding of the visual representation and L_2 normalization of both input modalities, instead of batch normalization (Ioffe and Szegedy, 2015), which we found to be beneficial when using MCB for the grounding task.

4 Evaluation on Visual Question Answering

We evaluate the benefit of MCB with a diverse set of ablations on two visual question answering datasets.

4.1 Datasets

The **Visual Question Answering (VQA)** real-image dataset (Antol et al., 2015) consists of approximately 200,000 MSCOCO images (Lin et al., 2014), with 3 questions per image and 10 answers per question. There are 3 data splits: train (80K images), validation (40K images), and test (80K images). Additionally, there is a 25% subset of test named test-dev. Accuracies for ablation experiments in this paper are reported on the test-dev data split. We use the VQA tool provided by Antol et al. (2015) for evaluation. We conducted most of our experiments on the open-ended real-image task. In Table 4, we also report our multiple-choice real-image scores.

The **Visual Genome** dataset (Krishna et al., 2016) uses 108,249 images from the intersection of YFCC100M (Thomee et al., 2015) and MSCOCO. For each image, an average of 17 question-answer pairs are collected. There are 1.7 million QA pairs of the 6W question types (*what, where, when, who, why*, and *how*). Compared to the VQA dataset, Visual Genome represents a more balanced distribution of the 6W question types. Moreover, the average question and answer lengths for Visual Genome are larger than the VQA dataset. To leverage the Visual Genome dataset as additional training data, we remove all the unnecessary words such as "a", "the", and "it is" from the answers to decrease the length of the answers and extract QA pairs whose answers are single-worded. The extracted data is filtered again based on the answer vocabulary space created from the VQA dataset, leaving us with additional 1M image-QA triplets.

The **Visual7W** dataset (Zhu et al., 2016) is a part of the Visual Genome. Visual7W adds a 7th *which* question category to accommodate visual answers,

Method	Accuracy
Element-wise Sum	56.50
Concatenation	57.49
Concatenation + FC	58.40
Concatenation + FC + FC	57.10
Element-wise Product	58.57
Element-wise Product + FC	56.44
Element-wise Product + FC + FC	57.88
MCB ($2048 \times 2048 \rightarrow 16$K)	**59.83**
Full Bilinear ($128 \times 128 \rightarrow 16$K)	58.46
MCB ($128 \times 128 \rightarrow 4$K)	58.69
Element-wise Product with VGG-19	55.97
MCB ($d = 16$K) with VGG-19	**57.05**
Concatenation + FC with Attention	58.36
MCB ($d = 16$K) with Attention	**62.50**

Table 1: Comparison of multimodal pooling methods. Models are trained on the VQA train split and tested on test-dev.

Compact Bilinear d	Accuracy
1024	58.38
2048	58.80
4096	59.42
8192	59.69
16000	**59.83**
32000	59.71

Table 2: Accuracies for different values of d, the dimension of the compact bilinear feature. Models are trained on the VQA train split and tested on test-dev. Details in Sec. 4.3.

Method	What	Where	When	Who	Why	How	Avg
Zhu et al.	51.5	57.0	75.0	59.5	55.5	49.8	54.3
Concat+Att.	47.8	56.9	74.1	62.3	52.7	**51.2**	52.8
MCB+Att.	**60.3**	**70.4**	**79.5**	**69.2**	**58.2**	51.1	**62.2**

Table 3: Multiple-choice QA tasks accuracy (%) on Visual7W test set.

but we only evaluate the models on the Telling task which involves 6W questions. The natural language answers in Visual7W are in a multiple-choice format and each question comes with four answer candidates, with only one being the correct answer. Visual7W is composed of 47,300 images from MSCOCO and there are a total of 139,868 QA pairs.

4.2 Experimental Setup

We use the Adam solver with $\epsilon = 0.0007$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We use dropout after the LSTM layers and in fully connected layers. For the experiments in Table 1 and 2, we train on the VQA train split, validate on the VQA validation split, and report results on the VQA test-dev split. We use early stopping: if the validation score does not improve for 50,000 iterations, we stop training and evaluate the best iteration on test-dev.

For the Visual7W task, we use the same hyperparameters and training settings as in the VQA experiments. We use the splits from (Zhu et al., 2016) to train, validate, and test our models. We also compute accuracies on this data using their evaluation code.

For VQA multiple choice, we train the open-ended models and take the argmax over the multiple choice answers at test time. For Visual7W, we use the answer encoding as described in Sec. 3.2.

4.3 Ablation Results

We compare the performance of non-bilinear and bilinear pooling methods in Table 1. We see that MCB pooling outperforms all non-bilinear pooling methods, such as eltwise sum, concatenation, and eltwise product.

One could argue that the compact bilinear method simply has more parameters than the non-bilinear pooling methods, which contributes to its performance. We compensated for this by stacking fully connected layers (with 4096 units per layer, ReLU activation, and dropout) after the non-bilinear pooling methods to increase their number of parameters. However, even with similar parameter budgets, non-bilinear methods could not achieve the same accuracy as the MCB method. For example, the "Concatenation + FC + FC" pooling method has approximately $4096^2 + 4096^2 + 4096 \times 3000 \approx 46$ million parameters, which matches the 48 million parameters available in MCB with $d = 16000$. However, the performance of the "Concatenation + FC + FC" method is only 57.10% compared to MCB's 59.83%.

Section 2 in Table 1 also shows that compact bi-

	Test-dev					Test-standard				
	Open Ended				MC	Open Ended				MC
	Y/N	No.	Other	All	All	Y/N	No.	Other	All	All
MCB	81.2	35.1	49.3	60.8	65.4	-	-	-	-	-
MCB + Genome	81.7	36.6	51.5	62.3	66.4	-	-	-	-	-
MCB + Att.	82.2	37.7	54.8	64.2	68.6	-	-	-	-	-
MCB + Att. + GloVe	82.5	37.6	55.6	64.7	69.1	-	-	-	-	-
MCB + Att. + Genome	81.7	38.2	57.0	65.1	69.5	-	-	-	-	-
MCB + Att. + GloVe + Genome	82.3	37.2	57.4	65.4	69.9	-	-	-	-	-
Ensemble of 7 Att. models	**83.4**	**39.8**	**58.5**	**66.7**	**70.2**	**83.2**	**39.5**	**58.0**	**66.5**	**70.1**
Naver Labs (challenge 2nd)	83.5	39.8	54.8	64.9	69.4	83.3	38.7	54.6	64.8	69.3
HieCoAtt (Lu et al., 2016)	79.7	38.7	51.7	61.8	65.8	-	-	-	62.1	66.1
DMN+ (Xiong et al., 2016)	80.5	36.8	48.3	60.3	-	-	-	-	60.4	-
FDA (Ilievski et al., 2016)	81.1	36.2	45.8	59.2	-	-	-	-	59.5	-
D-NMN (Andreas et al., 2016a)	81.1	38.6	45.5	59.4	-	-	-	-	59.4	-
AMA (Wu et al., 2016)	81.0	38.4	45.2	59.2	-	81.1	37.1	45.8	59.4	-
SAN (Yang et al., 2015)	79.3	36.6	46.1	58.7	-	-	-	-	58.9	-
NMN (Andreas et al., 2016b)	81.2	38.0	44.0	58.6	-	81.2	37.7	44.0	58.7	-
AYN (Malinowski et al., 2016)	78.4	36.4	46.3	58.4	-	78.2	36.3	46.3	58.4	-
SMem (Xu and Saenko, 2016)	80.9	37.3	43.1	58.0	-	80.9	37.5	43.5	58.2	-
VQA team (Antol et al., 2015)	80.5	36.8	43.1	57.8	62.7	80.6	36.5	43.7	58.2	63.1
DPPnet (Noh et al., 2015)	80.7	37.2	41.7	57.2	-	80.3	36.9	42.2	57.4	-
iBOWIMG (Zhou et al., 2015)	76.5	35.0	42.6	55.7	-	76.8	35.0	42.6	55.9	62.0

Table 4: Open-ended and multiple-choice (MC) results on VQA test set (trained on train+val set) compared with state-of-the-art: accuracy in %. See Sec. 4.4.

linear pooling has no impact on accuracy compared to full bilinear pooling. Section 3 in Table 1 demonstrates that the MCB brings improvements regardless of the image CNN used. We primarily use ResNet-152 in this paper, but MCB also improves performance if VGG-19 is used. Section 4 in Table 1 shows that our soft attention model works best with MCB pooling. In fact, attending to the Concatenation + FC layer has the same performance as not using attention at all, while attending to the MCB layer improves performance by 2.67 points.

Table 2 compares different values of d, the output dimensionality of the multimodal compact bilinear feature. Approximating the bilinear feature with a 16,000-D vector yields the highest accuracy.

We also evaluated models with multiple attention maps or channels. One attenion map achieves 64.67%, two 65.08% and four 64.24% accuracy (trained on train+val). Visual inspection of the generated attention maps reveals that an ensembling or smoothing effect occurs when using multiple maps.

Table 3 presents results for the Visual7W multiple-choice QA task. The MCB with attention model outperforms the previous state-of-the-art by 7.9 points overall and performs better in almost every category.

4.4 Comparison to State-of-the-Art

Table 4 compares our approach with the state-of-the-art on VQA test set. Our best single model uses MCB pooling with two attention maps. Additionally, we augment our training data with images and QA pairs from the Visual Genome dataset. We also concatenate the learned word embedding with pretrained GloVe vectors (Pennington et al., 2014).

Each model in our ensemble of 7 models uses MCB with attention. Some of the models were trained with data from Visual Genome, and some were trained with two attention maps. This ensem-

Method	Accuracy, %
Plummer et al. (2015)	27.42
Hu et al. (2016b)	27.80
Plummer et al. (2016)[1]	43.84
Wang et al. (2016)	43.89
Rohrbach et al. (2016)	47.81
Concatenation	46.50
Element-wise Product	47.41
Element-wise Product + Conv	47.86
MCB	**48.69**

Table 5: Grounding accuracy on Flickr30k Entities dataset.

Method	Accuracy, %
Hu et al. (2016b)	17.93
Rohrbach et al. (2016)	26.93
Concatenation	25.48
Element-wise Product	27.80
Element-wise Product + Conv	27.98
MCB	**28.91**

Table 6: Grounding accuracy on ReferItGame dataset.

ble is 1.8 points above the next best approach on the VQA open-ended task and 0.8 points above the next best approach on the multiple-choice task (on Test-dev). Even without ensembles, our "MCB + Genome + Att. + GloVe" model still outperforms the next best result by 0.5 points, with an accuracy of 65.4% versus 64.9% on the open-ended task (on Test-dev).

5 Evaluation on Visual Grounding

5.1 Datasets

We evaluate our visual grounding approach on two datasets. The first is Flickr30k Entities (Plummer et al., 2015) which consists of 31K images from Flickr30k dataset (Hodosh et al., 2014) with 244K phrases localized with bounding boxes. We follow the experimental setup of Rohrbach et al. (2016), e.g. we use the same Selective Search (Uijlings et al., 2013) object proposals and the Fast R-CNN (Girshick, 2015) fine-tuned VGG16 features (Simonyan and Zisserman, 2014). The second dataset is ReferItGame (Kazemzadeh et al., 2014), which contains 20K images from IAPR TC-12 dataset (Grubinger et al., 2006) with segmented regions from SAIAPR-12 dataset (Escalante et al., 2010) and 120K associated natural language referring expressions. For ReferItGame we follow the experimental setup of Hu et al. (2016b) and rely on their ground-truth bounding boxes extracted around the segmentation masks. We use the Edge Box (Zitnick and Dollár, 2014) object proposals and visual features (VGG16 combined with the spatial features, which encode bounding box relative position) from Hu et al. (2016b).

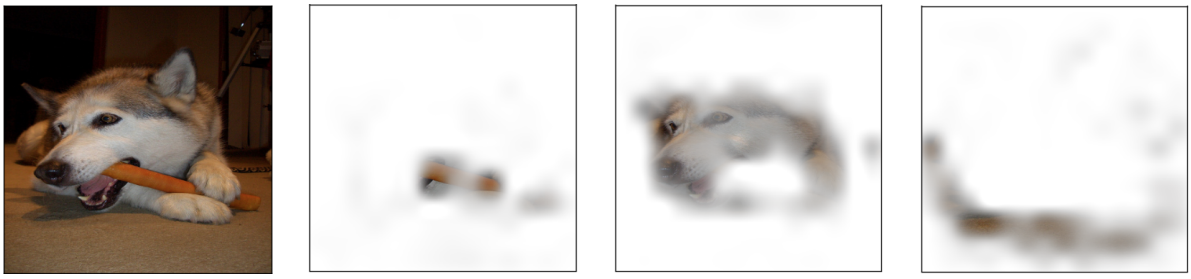
5.2 Experimental Setup

In all experiments we use Adam solver (Kingma and Ba, 2014) with learning rate $\epsilon = 0.0001$. The embedding size is 500 both for visual and language embeddings. We use $d = 2048$ in the MCB pooling, which we found to work best for the visual grounding task. The accuracy is measured as percentage of query phrases which have been localized correctly. The phrase is localized correctly if the predicted bounding box overlaps with the ground-truth bounding box by more than 50% intersection over union (IOU).

5.3 Results

Tables 5 and 6 summarize our results in the visual grounding task. We present multiple ablations of our proposed architecture. First, we replace the MCB with simple concatenation of the embedded visual feature and the embedded phrase, resulting in 46.5% on the Flickr30k Entities and 25.48% on the ReferItGame datasets. The results can be improved by replacing the concatenation with the element-wise product of both embedded features (47.41% and 27.80%). We can further slightly increase the performance by introducing additional 2048-D convolution after the element-wise product (47.86% and 27.98%). However, even with fewer parameters, our MCB pooling significantly improves over this baseline on both datasets, reaching state-of-the-art accuracy of 48.69% on Flickr30k Entities and 28.91% on ReferItGame dataset. Figure 6 (bottom) shows examples of improved phrase localization.

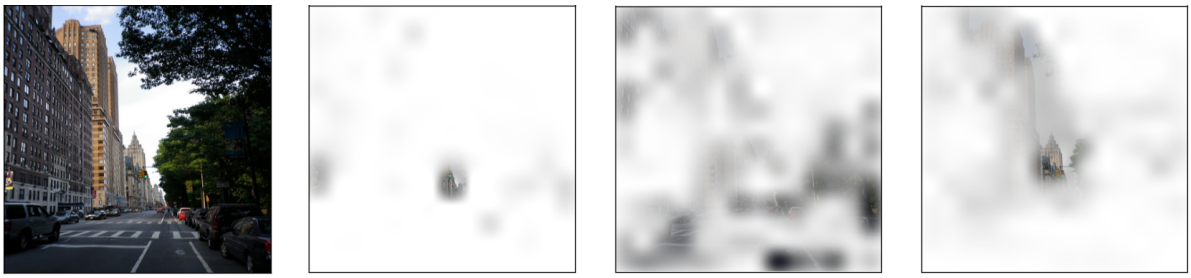
[1]Plummer et al. (2016) achieve higher accuracy of 50.89% when taking into account box size and color. We believe our approach would also benefit from such additional features.



What vegetable is the dog chewing on?
MCB: carrot
GT: carrot

What kind of dog is this?
MCB: husky
GT: husky

What kind of flooring does the room have?
MCB: carpet
GT: carpet



What color is the traffic light?
MCB: green
GT: green

Is this an urban area?
MCB: yes
GT: yes

Where are the buildings?
MCB: in background
GT: on left

MCB | Eltwise Product + Conv | MCB | Eltwise Product + Conv

A tattooed woman with a green dress and yellow backpack holding a water bottle is walking across the street.

A dog distracts his owner from working at her computer.

Figure 6: Top: predicted answers and attention maps from MCB model on VQA images. Bottom: predicted grounding from MCB model (left) and Eltwise Product + Conv model (right) on Flickr30k Entities images.

6 Conclusion

We propose the Multimodal Compact Bilinear Pooling (MCB) to combine visual and text representations. For visual question answering, our architecture with attention and multiple MCBs gives significant improvements on two VQA datasets compared to state-of-the-art. In the visual grounding task, introducing MCB pooling leads to improved phrase localization accuracy, indicating better interaction between query phrase representations and visual representations of proposal bounding boxes. The code to replicate our experiments is available at `https://github.com/akirafukui/vqa-mcb`.

Acknowledgments

We would like to thank Yang Gao and Oscar Beijbom for helpful discussions about Compact Bilinear Pooling. This work was supported by DARPA, AFRL, DoD MURI award N000141110688, NSF awards IIS-1427425 and IIS-1212798, and the Berkeley Artificial Intelligence Research (BAIR) Lab.

References

[Andreas et al.2016a] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[Andreas et al.2016b] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Antol et al.2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Charikar et al.2002] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *Automata, languages and programming*, pages 693–703. Springer.

[Deng et al.2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Donahue et al.2013] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2013. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Escalante et al.2010] Hugo Jair Escalante, Carlos A Hernández, Jesus A Gonzalez, Aurelio López-López, Manuel Montes, Eduardo F Morales, L Enrique Sucar, Luis Villaseñor, and Michael Grubinger. 2010. The segmented and annotated iapr tc-12 benchmark. *Computer Vision and Image Understanding*, 114(4):419–428.

[Frome et al.2013] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NIPS)*.

[Gao et al.2016] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Girshick2015] Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Gong et al.2014] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Grubinger et al.2006] Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International Workshop OntoImage*, volume 5, page 10.

[Hardoon et al.2004] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Hodosh et al.2014] Peter Hodosh, Alice Young, Micah Lai, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In *Transactions of the Association for Computational Linguistics (TACL)*.

[Hu et al.2016a] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. 2016a. Segmentation from natural language expressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Hu et al.2016b] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016b. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Ilievski et al.2016] Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. 2016. A focused dynamic attention model for visual question answering. *arXiv:1604.01485*.

[Ioffe and Szegedy2015] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Karpathy and Fei-Fei2015] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kazemzadeh et al.2014] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. 2014. Referit game: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Kiros et al.2014] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 595–603.

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems (NIPS)*.

[Klein et al.2015] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2015. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Krishna et al.2016] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv:1602.07332*.

[Kumar et al.2016] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Lin et al.2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Lin et al.2015] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. 2015. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Lu et al.2016] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical Co-Attention for Visual Question Answering. In *Advances in Neural Information Processing Systems (NIPS)*.

[Malinowski et al.2016] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2016. Ask Your Neurons: A Deep Learning Approach to Visual Question Answering. *arXiv: 1605.02697*.

[Mao et al.2015] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Ngiam et al.2011] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 689–696.

[Noh et al.2015] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2015. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Pham and Pagh2013] Ninh Pham and Rasmus Pagh. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 239–247, New York, NY, USA. ACM.

[Plummer et al.2015] Bryan Plummer, Liwei Wang, Chris Cervantes, Juan Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Plummer et al.2016] Bryan Plummer, Liwei Wang, Chris Cervantes, Juan Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2016. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *arXiv:1505.04870v3*.

[Rohrbach et al.2016] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Simonyan and Zisserman2014] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Socher et al.2014] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[Tenenbaum and Freeman2000] Joshua B Tenenbaum and William T Freeman. 2000. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283.

[Thomee et al.2015] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2015. The

new data and new challenges in multimedia research. *CoRR*, abs/1503.01817.

[Uijlings et al.2013] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. 2013. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 104(2).

[Wang et al.2016] Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Weston et al.2011] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[Wu et al.2016] Qi Wu, Peng Wang, Chunhua Shen, Anton van den Hengel, and Anthony Dick. 2016. Ask Me Anything: Free-form Visual Question Answering Based on Knowledge from External Sources. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*.

[Xiong et al.2016] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Xu and Saenko2016] Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the International Conference on Machine Learning (ICML)*.

[Yang et al.2015] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. *arXiv:1511.02274*.

[Zhou et al.2015] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv:1512.02167*.

[Zhu et al.2016] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded Question Answering in Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Zitnick and Dollár2014] C Lawrence Zitnick and Piotr Dollár. 2014. Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–405. Springer.

The Structured Weighted Violations Perceptron Algorithm

Rotem Dror and **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
{rtmdrr@campus|roiri@ie}.technion.ac.il

Abstract

We present the *Structured Weighted Violations Perceptron (SWVP)* algorithm, a new structured prediction algorithm that generalizes the Collins Structured Perceptron (CSP, (Collins, 2002)). Unlike CSP, the update rule of SWVP explicitly exploits the internal structure of the predicted labels. We prove the convergence of SWVP for linearly separable training sets, provide mistake and generalization bounds, and show that in the general case these bounds are tighter than those of the CSP special case. In synthetic data experiments with data drawn from an HMM, various variants of SWVP substantially outperform its CSP special case. SWVP also provides encouraging initial dependency parsing results.

1 Introduction

The structured perceptron ((Collins, 2002), henceforth denoted CSP) is a prominent training algorithm for structured prediction models in NLP, due to its effective parameter estimation and simple implementation. It has been utilized in numerous NLP applications including word segmentation and POS tagging (Zhang and Clark, 2008), dependency parsing (Koo and Collins, 2010; Goldberg and Elhadad, 2010; Martins et al., 2013), semantic parsing (Zettlemoyer and Collins, 2007) and information extraction (Hoffmann et al., 2011; Reichart and Barzilay, 2012), if to name just a few.

Like some training algorithms in structured prediction (e.g. structured SVM (Taskar et al., 2004; Tsochantaridis et al., 2005), MIRA (Crammer and Singer, 2003) and LaSo (Daumé III and Marcu, 2005)), CSP considers in its update rule the difference between *complete* predicted and gold standard labels (Sec. 2). Unlike others (e.g. factored MIRA (McDonald et al., 2005b; McDonald et al., 2005a) and dual-loss based methods (Meshi et al., 2010)) it does not exploit the structure of the predicted label. This may result in valuable information being lost.

Consider, for example, the gold and predicted dependency trees of Figure 1. The substantial difference between the trees may be mostly due to the difference in roots (*are* and *worse*, respectively). Parameter update w.r.t this mistake may thus be more useful than an update w.r.t the complete trees.

In this work we present a new perceptron algorithm with an update rule that exploits the structure of a predicted label when it differs from the gold label (Section 3). Our algorithm is called *The Structured Weighted Violations Perceptron (SWVP)* as its update rule is based on a weighted sum of updates w.r.t *violating assignments* and *non-violating assignments*: assignments to the input example, derived from the predicted label, that score higher (for violations) and lower (for non-violations) than the gold standard label according to the current model.

Our concept of *violating assignment* is based on Huang et al. (2012) that presented a variant of the CSP algorithm where the argmax inference problem is replaced with a violation finding function. Their update rule, however, is identical to that of the CSP algorithm. Importantly, although CSP and the above variant do not exploit the internal structure of the predicted label, they are special cases of SWVP.

In Section 4 we prove that for a linearly separable training set, SWVP converges to a linear separator of

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 469–478,
Austin, Texas, November 1-5, 2016.

the data under certain conditions on the parameters of the algorithm, that are respected by the CSP special case. We further prove mistake and generalization bounds for SWVP, and show that in the general case the SWVP bounds are tighter than the CSP's.

In Section 5 we show that SWVP allows *aggressive* updates, that exploit only violating assignments derived from the predicted label, and more *balanced* updates, that exploit both violating and non-violating assignments. In experiments with synthetic data generated by an HMM, we demonstrate that various SWVP variants substantially outperform CSP training. We also provide initial encouraging dependency parsing results, indicating the potential of SWVP for real world NLP applications.

2 The Collins Structured Perceptron

In structured prediction the task is to find a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $y \in \mathcal{Y}$ is a structured object rather than a scalar, and a feature mapping $\phi(x, y) : \mathcal{X} \times \mathcal{Y}(x) \rightarrow \mathbb{R}^d$ is given. In this work we denote $\mathcal{Y}(x) = \{y' | y' \in {D_Y}^{L_x}\}$, where L_x, a scalar, is the size of the allowed output sequence for an input x and D_Y is the domain of y'_i for every $i \in \{1, \ldots L_x\}$. [1] Our results, however, hold for the general case of an output space with variable size vectors as well.

The CSP algorithm (Algorithm 1) aims to learn a parameter (or weight) vector $\mathbf{w} \in \mathbb{R}^d$, that separates the training data, i.e. for each training example (x, y) it holds that: $y = \arg\max_{y' \in \mathcal{Y}(x)} \mathbf{w} \cdot \phi(x, y')$. To find such a vector the algorithm iterates over the training set examples and solves the above inference ($argmax$) problem. If the inferred label y^* differs from the gold label y the update $\mathbf{w} = \mathbf{w} + \Delta\phi(x, y, y^*)$ is performed. For linearly separable training data (see definition 4), CSP is proved to converge to a vector $\mathbf{w}$ separating the training data.

Collins and Roark (2004) and Huang et al. (2012) expanded the CSP algorithm by proposing various alternatives to the argmax *inference* problem which is often intractable in structured prediction problems (e.g. in high-order graph-based dependency parsing (McDonald and Pereira, 2006)). The basic idea is replacing the argmax problem with the search for a *violation*: an output label that the model scores higher than the gold standard label. The update rule in these CSP variants is, however, identical to the CSP's. We, in contrast, propose a novel update rule that exploits the internal structure of the model's prediction regardless of the way this prediction is generated.

[1] In the general case L_x is a set of output sizes, which may be finite or infinite (as in constituency parsing (Collins, 1997)).

Algorithm 1 The Structured Perceptron (CSP)

Input: data $D = \{x^i, y^i\}_{i=1}^n$, feature mapping ϕ
Output: parameter vector $\mathbf{w} \in \mathbb{R}^d$
Define: $\Delta\phi(x, y, z) \triangleq \phi(x, y) - \phi(x, z)$
Initialize $\mathbf{w} = 0$.
repeat
  for each $(x^i, y^i) \in D$ **do**
    $y^* = \arg\max_{y' \in \mathcal{Y}(x^i)} \mathbf{w} \cdot \phi(x^i, y')$
    if $y^* \neq y^i$ **then**
      $\mathbf{w} = \mathbf{w} + \Delta\phi(x^i, y^i, y^*)$
    end if
  end for
until Convergence

3 The Structured Weighted Violations Perceptron (SWVP)

SWVP exploits the internal structure of a predicted label $y^* \neq y$ for a training example $(x, y) \in D$, by updating the weight vector with respect to substructures of y^*. We start by presenting the fundamental concepts at the basis of our algorithm.

3.1 Basic Concepts

Sub-structure Sets We start with two fundamental definitions: **(1)** An individual *sub-structure* of a structured object (or label) $y \in {D_Y}^{L_x}$, denoted with J, is defined to be a subset of indexes $J \subseteq [L_x]$;[2] and **(2)** A *set of substructures* for a training example (x, y), denoted with JJ_x, is defined as $JJ_x \subseteq 2^{[L_x]}$.

Mixed Assignment We next define the concept of a *mixed assignment*:

Definition 1. *For a training pair (x, y) and a predicted label $y^* \in \mathcal{Y}(x)$, $y^* \neq y$, a **mixed assignment** (MA) vector denoted as $m^J(y^*, y)$ is defined with respect to $J \in JJ_x$ as follows:*

$$m_k^J(y^*, y) = \begin{cases} y_k^* & k \in J \\ y_k & else \end{cases}$$

That is, a mixed assignment is a new label, derived from the predicted label y^*, that is identical to y^* in all indexes in J and to y otherwise. For simplicity we denote $m^J(y^*, y) = m^J$ when the reference y^* and y labels are clear from the context.

[2] We use the notation $[n] = \{1, 2, \ldots n\}$.

Consider, for example, the trees of Figure 1, assuming that the top tree is y, the middle tree is y^* and $J = [2, 5]$.[3] In the $m^J(y^*, y)$ (bottom) tree the heads of all the words are identical to those of the top tree, except for the heads of *mistakes* and of *then*.

Violation The next central concept is that of a violation, originally presented by Huang et al. (2012):

Definition 2. *A triple (x, y, y^*) is said to be a **violation** with respect to a training example (x, y) and a parameter vector **w** if for $y^* \in \mathcal{Y}(x)$ it holds that $y^* \neq y$ and $\mathbf{w} \cdot \Delta\phi(x, y, y^*) \leq 0$.*

The SWVP algorithm distinguishes between MAs that are violations, and ones that are not. For a triplet (x, y, y^*) and a set of substructures $JJ_x \subseteq 2^{[L_x]}$ we provide the following notations:

$$I(y^*, y, JJ_x)^v = \{J \in JJ_x | m^J \neq y, \mathbf{w} \cdot \Delta\phi(x, y, m^J) \leq 0\}$$

$$I(y^*, y, JJ_x)^{nv} = \{J \in JJ_x | m^J \neq y, \mathbf{w} \cdot \Delta\phi(x, y, m^J) > 0\}$$

This notation divides the set of substructures into two subsets, one consisting of the substructures that yield violating MAs and one consisting of the substructures that yield non-violating MAs. Here again when the reference label y^* and the set JJ_x are known we denote: $I(y^*, y, JJ_x)^v = I^v$, $I(y^*, y, JJ_x)^{nv} = I^{nv}$ and $I = I^v \cup I^{nv}$.

Weighted Violations The key idea of SWVP is the exploitation of the internal structure of the predicted label in the update rule. For this aim at each iteration we define the set of substructures, JJ_x, and then, for each $J \in JJ_x$, update the parameter vector, **w**, with respect to the mixed assignments, MA^J's. This is a more flexible setup compared to CSP, as we can update with respect to the predicted output (if it is a violation, as is promised if inference is performed via argmax), if we wish to do so, as well as with respect to other mixed assignments.

Naturally, not all mixed assignments are equally important for the update rule. Hence, we weigh the different updates using a weight vector γ. This paper therefore extends the observation of Huang et al. (2012) that perceptron parameter update can be performed w.r.t violations (Section 2), by showing that **w** can actually be updated w.r.t linear combinations of mixed assignments, under certain conditions on the selected weights.

[3] We index the dependency tree words from 1 onwards.

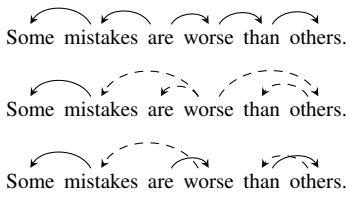


Figure 1: Example parse trees: gold tree (y, top), predicted tree (y^*, middle) with arcs differing from the gold's marked with a dashed line, and $m^J(y^*, y)$ for $J = [2, 5]$ (bottom tree).

3.2 Algorithm

With these definitions we can present the SWVP algorithm (Algorithm 2). SWVP is in fact a family of algorithms differing with respect to two decisions that can be made at each pass over each training example (x, y): the choice of the set JJ_x and the implementation of the SETGAMMA function.

SWVP is very similar to CSP except for in the update rule. Like in CSP, the algorithm iterates over the training data examples and for each example it first predicts a label according to the current parameter vector **w** (inference is discussed in Section 4.2, property 2). The main difference from CSP is in the update rule (lines 6-12). Here, for each substructure in the substructure set, $J \in JJ_x$, the algorithm generates a mixed assignment m^J (lines 7-9). Then, **w** is updated with a weighted sum of the mixed assignments (line 11), unlike in CSP where the update is held w.r.t the predicted assignment only.

The $\gamma(m^J)$ weights assigned to each of the $\Delta\phi(x, y, m^J)$ updates are defined by a SETGAMMA function (line 10). Intuitively, a $\gamma(m^J)$ weight should be higher the more the mixed assignment is assumed to convey useful information that can guide the update of **w** in the right direction. In Section 4 we detail the conditions on SETGAMMA under which SWVP converges, and in Section 5 we describe various SETGAMMA implementations.

Going back to the example of Figure 1, one would assume (Sec. 1) that the head word prediction for *worse* is pivotal to the substantial difference between the two top trees (UAS of 0.2). CSP does not directly exploit this observation as it only updates its parameter vector with respect to the differences between complete assignments: $\mathbf{w} = \mathbf{w} + \Delta\phi(x, y, z)$.

In contrast, SWVP can exploit this observation in various ways. For example, it can generate a mixed

assignment for each of the erroneous arcs where all other words are assigned their correct arc (according to the gold tree) except for that specific arc which is kept as in the bottom tree. Then, higher weights can be assigned to errors that seem more central than others. We elaborate on this in the next two sections.

Algorithm 2 The Structured Weighted Violations Perceptron

Input: data $D = \{x^i, y^i\}_{i=1}^n$, feature mapping ϕ
Output: parameter vector $\mathbf{w} \in \mathbb{R}^d$
Define: $\Delta\phi(x, y, z) \triangleq \phi(x, y) - \phi(x, z)$
1: Initialize $\mathbf{w} = 0$.
2: **repeat**
3: **for** each $(x^i, y^i) \in D$ **do**
4: $y^* = \arg\max_{y' \in \mathcal{Y}(x^i)} \mathbf{w} \cdot \phi(x^i, y')$
5: **if** $y^* \neq y^i$ **then**
6: **Define:** $JJ_{x^i} \subseteq 2^{[L_{x^i}]}$
7: **for** $J \in JJ_{x^i}$ **do**
8: **Define:** m^J s.t. $m_k^J = \begin{cases} y_k^* & k \in J \\ y_k^i & else \end{cases}$
9: **end for**
10: $\gamma = \text{SETGAMMA}()$
11: $\mathbf{w} = \mathbf{w} + \sum_{J \in I^v \cup I^{nv}} \gamma(m^J)\Delta\phi(x^i, y^i, m^J)$
12: **end if**
13: **end for**
14: **until** Convergence

4 Theory

We start this section with the convergence conditions on the γ vector which weighs the mixed assignment updates in the SWVP update rule (line 11). Then, using these conditions, we describe the relation between the SWVP and the CSP algorithms. After that, we prove the convergence of SWVP and analyse the derived properties of the algorithm.

γ Selection Conditions Our main observation in this section is that SWVP converges under two conditions: **(a)** the training set D is linearly separable; and **(b)** for any parameter vector $\mathbf{w}$ achievable by the algorithm, there exists $(x, y) \in D$ with $JJ_x \subseteq 2^{[L_x]}$, such that for the predicted output $y^* \neq y$, SETGAMMA returns a γ weight vector that respects the γ selection conditions defined as follows:

Definition 3. *The γ **selection conditions** for the SWVP algorithm are ($I = I^v \cup I^{nv}$):*

$$(1) \sum_{J \in I} \gamma(m^J) = 1. \quad \gamma(m^J) \geq 0, \quad \forall J \in I.$$

$$(2) \quad \boldsymbol{w} \cdot \sum_{J \in I} \gamma(m^J)\Delta\phi(x^i, y^i, m^J) \leq 0.$$

With this definition we are ready to prove the following property.

SWVP Generalizes the CSP Algorithm We now show that the CSP algorithm is a special case of SWVP. CSP can be derived from SWVP when taking: $JJ_x = \{[L_x]\}$, and $\gamma(m^{[L_x]}) = 1$ for every $(x, y) \in D$. With these parameters, the γ selection conditions hold for every $\mathbf{w}$ and y^*. Condition (1) holds trivially as there is only one γ coefficient and it is equal to 1. Condition (2) holds as $y^* = m^{[L_x]}$ and hence $I = \{[L_x]\}$ and $w \cdot \sum_{J \in I} \Delta\phi(x, y, m^J) \leq 0$.

4.1 Convergence for Linearly Separable Data

Here we give the theorem regarding the convergence of the SWVP in the separable case. We first define:

Definition 4. *A data set $D = \{x^i, y^i\}_{i=1}^n$ is **linearly separable with margin** $\delta > 0$ if there exists some vector $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_2 = 1$ such that for all i:*

$$\boldsymbol{u} \cdot \Delta\phi(x^i, y^i, z) \geq \delta, \forall z \in \mathcal{Y}(x^i).$$

Definition 5. *The **radius** of a data set $D = \{x^i, y^i\}_{i=1}^n$ is the minimal scalar R s.t for all i:*

$$\|\Delta\phi(x^i, y^i, z)\| \leq R, \forall z \in \mathcal{Y}(x^i).$$

We next extend these definitions:

Definition 6. *Given a data set $D = \{x^i, y^i\}_{i=1}^n$ and a set $JJ = \{JJ_{x^i} \subseteq 2^{[L_{x^i}]} | (x^i, y^i) \in D\}$, D is **linearly separable w.r.t JJ, with margin** $\delta^{JJ} > 0$ if there exists a vector $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_2 = 1$ such that: $\boldsymbol{u} \cdot \Delta\phi(x^i, y^i, m^J(z, y^i)) \geq \delta^{JJ}$ for all $i, z \in \mathcal{Y}(x^i), J \in JJ_{x^i}$.*

Definition 7. *The **mixed assignment radius w.r.t** JJ of a data set $D = \{x^i, y^i\}_{i=1}^n$ is a constant R^{JJ} s.t for all i it holds that:*

$$\|\Delta\phi(x^i, y^i, m^J(z, y^i))\| \leq R^{JJ}, \forall z \in \mathcal{Y}(x^i), J \in JJ_{x^i}.$$

With these definitions we can make the following observation (proof in A):

Observation 1. For linearly separable data D and a set JJ, every unit vector $\mathbf{u}$ that separates the data with margin δ, also separates the data with respect to mixed assignments with JJ, with margin $\delta^{JJ} \geq \delta$. Likewise, it holds that $R^{JJ} \leq R$.

We can now state our convergence theorem. While the proof of this theorem resembles that of the CSP (Collins, 2002), unlike the CSP proof the SWVP proof relies on the γ *selection conditions* presented above and on the *Jensen inequality*.

Theorem 1. *For any dataset D, linearly separable with respect to JJ with margin $\delta^{JJ} > 0$, the SWVP algorithm terminates after $t \leq \frac{(R^{JJ})^2}{(\delta^{JJ})^2}$ steps, where R^{JJ} is the mixed assignment radius of D w.r.t. JJ.*

Proof. Let $\mathbf{w}^t$ be the weight vector before the t^{th} update, thus $\mathbf{w}^1 = 0$. Suppose the t^{th} update occurs on example (x, y), i.e. for the predicted output y^* it holds that $y^* \neq y$. We will bound $\|\mathbf{w}^{t+1}\|^2$ from both sides.
First, it follows from the update rule of the algorithm that: $\mathbf{w}^{t+1} = \mathbf{w}^t + \sum_{J \in I^v \cup I^{nv}} \gamma(m^J) \Delta\phi(x, y, m^J)$. For simplicity, in this proof we will use the notation $I^v \cup I^{nv} = I$. Hence, multiplying each side of the equation by $\mathbf{u}$ yields:

$$\begin{aligned}
\mathbf{u} \cdot \mathbf{w}^{t+1} &= \mathbf{u} \cdot \mathbf{w}^t + \mathbf{u} \cdot \sum_{J \in I} \gamma(m^J) \Delta\phi(x, y, m^J) \\
&= \mathbf{u} \cdot \mathbf{w}^t + \sum_{J \in I} \gamma(m^J) \mathbf{u} \cdot \Delta\phi(x, y, m^J) \\
&\geq \mathbf{u} \cdot \mathbf{w}^t + \sum_{J \in I} \gamma(m^J) \delta^{JJ} \quad \text{(margin property)} \\
&\geq \mathbf{u} \cdot \mathbf{w}^t + \delta^{JJ} \geq \ldots \geq t\delta^{JJ}.
\end{aligned}$$

The last inequality holds because $\sum_{J \in I} \gamma(m^J) = 1$. From this we get that $\|w^{t+1}\|^2 \geq (\delta^{JJ})^2 t^2$ since $\|u\|$=1. Second,

$$\begin{aligned}
\|\mathbf{w}^{t+1}\|^2 &= \|\mathbf{w}^t + \sum_{J \in I} \gamma(m^J) \Delta\phi(x, y, m^J)\|^2 \\
&= \|\mathbf{w}^t\|^2 + \|\sum_{J \in I} \gamma(m^J) \Delta\Phi(x, y, m^J)\|^2 \\
&\quad + 2\mathbf{w}^t \cdot \sum_{J \in I} \gamma(m^J) \Delta\Phi(x, y, m^J).
\end{aligned}$$

From γ selection condition (2) we get that:

$$\begin{aligned}
\|\mathbf{w}^{t+1}\|^2 &\leq \|\mathbf{w}^t\|^2 + \|\sum_{J \in I} \gamma(m^J) \Delta\Phi(x, y, m^J)\|^2 \\
&\leq \|\mathbf{w}^t\|^2 + \sum_{J \in I} \gamma(m^J) \|\Delta\Phi(x, y, m^J)\|^2 \\
&\leq \|\mathbf{w}^t\|^2 + (R^{JJ})^2. \quad \text{(radius property)}
\end{aligned}$$

The inequality one before the last results from the *Jensen inequality* which holds due to **(a)** γ selection condition (1); and **(b)** the squared norm function being convex. From this we finally get:

$$\|\mathbf{w}^{t+1}\|^2 \leq \|\mathbf{w}^t\|^2 + (R^{JJ})^2 \leq \ldots \leq t(R^{JJ})^2.$$

Combining the two steps we get:

$$(\delta^{JJ})^2 t^2 \leq \|\mathbf{w}^{t+1}\|^2 \leq t(R^{JJ})^2.$$

From this it is easy to derive the upper bound in the theorem: $t \leq \frac{(R^{JJ})^2}{(\delta^{JJ})^2}$. □

4.2 Convergence Properties

We next point on three properties of the SWVP algorithm, derived from its convergence proof:

Property 1 (tighter iterations bound) The convergence proof of CSP (Collins, 2002) is given for a vector $\mathbf{u}$ that linearly separates the data, with margin δ and for a data radius R. Following observation 1, it holds that in our case, $\mathbf{u}$ also linearly separates the data with respect to mixed assignments with a set JJ and with margin $\delta^{JJ} \geq \delta$. Together with the definition of $R^{JJ} \leq R$ we get that: $\frac{(R^{JJ})^2}{(\delta^{JJ})^2} \leq \frac{R^2}{\delta^2}$. This means that the bound on the number of updates made by SWVP is tighter than the bound of CSP.

Property 2 (inference) From the γ selection conditions it holds that any label from which at least one violating MA can be derived through JJ_x is suitable for an update. This is because in such a case we can choose, for example, a SETGAMMA function that assigns the weight of 1 to that MA, and the weight of 0 to all other MAs.

Algorithm 2 employs the $argmax$ inference function, following the basic reasoning that it is a good choice to base the parameter update on. Importantly, if the inference function is argmax and the algorithm performs an update ($y^* \neq y$), this means that y^*, the output of the argmax function, is a violating MA by definition. However, it is obvious that solving the inference problem and the optimal γ assignment problems jointly may result in more informed parameter ($\mathbf{w}$) updates. We leave a deeper investigation of this issue to future research.

Property 3 (dynamic updates) The γ selection conditions paragraph states two conditions ((a) and (b)) under which the convergence proof holds. While it is trivial for SETGAMMA to generate a γ vector that respects condition (a), if there is a parameter vector **w'** achievable by the algorithm for which SETGAMMA cannot generate γ that respects condition (b), SWVP gets stuck when reaching **w'**.

This problem can be solved with *dynamic updates*. A deep look into the convergence proof reveals that the set JJ_x and the SETGAMMA function can actually differ between iterations. While this will change the bound on the number of iterations, it will not change the fact that the algorithm converges if the data is linearly separable. This makes SWVP highly flexible as it can always

back off to the CSP setup of $JJ_x = \{[L_x]\}$, and $\forall(x,y) \in D : \gamma(m^{[L_x]}) = 1$, update its parameters and continue with its original JJ and SETGAMMA when this option becomes feasible. If this does not happen, the algorithm can continue till convergence with the CSP setup.

4.3 Mistake and Generalization Bounds

The following bounds are proved: the number of updates in the separable case (see Theorem 1); the number of mistakes in the non-separable case (see Appendix B); and the probability to misclassify an unseen example (see supplementary material). It can be shown that in the general case these bounds are tighter than those of the CSP special case. We next discuss variants of SWVP.

5 Passive Aggressive SWVP

Here we present types of update rules that can be implemented within SWVP. Such rule types are defined by: (a) the selection of γ, which should respect the γ *selection conditions* (see Definition 3) and (b) the selection of $JJ = \{JJ_x \subseteq 2^{[L_x]} | (x,y) \in D\}$, the substructure sets for the training examples.

γ Selection A first approach we consider is the *aggressive approach*[4] where only mixed assignments that are violations $\{m^J : J \in I^v\}$ are exploited (i.e. for all $J \in I^{nv}, \gamma(m^J) = 0$). Note, that in this case condition (2) of the γ *selection conditions* trivially holds as: $\mathbf{w} \cdot \sum_{J \in I^v} \gamma(m^J)\Delta\phi(x,y,m^J) \leq 0$. The only remaining requirement is that condition (1) also holds, i.e. that $\sum_{J \in I^v} \gamma(m^J) = 1$.

The opposite, *passive approach*, exploits only non-violating MA's $\{m^J : J \in I^{nv}\}$. However, such γ assignments do not respect γ selection condition (2), as they yield: $\mathbf{w} \cdot \sum_{J \in I^{nv}} \gamma(m^J)\Delta\phi(x,y,m^J) \leq 0$ which holds if and only if for every $J \in I^{nv}$, $\gamma(m^J) = 0$ that in turn contradicts condition (1).

Finally, we can take a *balanced approach* which gives a positive γ coefficient for at least one violating MA and at least one positive γ coefficient for a non-violating MA. This approach is allowed by SWVP as long as both γ selection conditions hold.

[4]We borrow the term *passive-aggressive* from (Crammer et al., 2006), despite the substantial difference between the works.

We implemented two weighting methods, both based on the concept of margin:

(1) Weighted Margin (WM): $\gamma(m^J) = \frac{|\mathbf{w}\cdot\Delta\phi(x,y,m^J)|^\beta}{\sum_{J' \in JJ_x} |\mathbf{w}\cdot\Delta\phi(x,y,m^{J'})|^\beta}$

(2) Weighted Margin Rank (WMR): $\gamma(m^J) = \left(\frac{|JJ_x|-r}{|JJ_x|}\right)^\beta$. where r is the rank of $|\mathbf{w} \cdot \Delta\phi(x,y,m^J(y^*,y))|$ among the $|\mathbf{w} \cdot \Delta\phi(x,y,m^{J'}(y^*,y))|$ values for $J' \in JJ_x$.

Both schemes were implemented twice, within a balanced approach (denoted as B) and an aggressive approach (denoted as A).[5] The aggressive schemes respect both γ selection conditions. The balanced schemes, however, respect the first condition but not necessarily the second. Since all models that employ the balanced weighting schemes converged after at most 10 iterations, we did not impose this condition (which we could do by, e.g., excluding terms for $J \in I^{nv}$ till condition (2) holds).

JJ Selection Another choice that strongly affects the updates made by SWVP is that of JJ. A choice of $JJ_x = 2^{[L_x]}$, for every $(x,y) \in D$ results in an update rule which considers all possible mixing assignments derived from the predicted label y^* and the gold label y. Such an update rule, however, requires computing a sum over an exponential number of terms (2^{L_x}) and is therefore highly inefficient.

Among the wide range of alternative approaches, in this paper we exploit *single difference* mixed assignments. In this approach we define: $JJ = \{JJ_x = \{\{1\},\{2\},\ldots\{L_x\}\} | (x,y) \in D\}$. For a training pair $(x,y) \in D$, a predicted label y^* and $J = \{j\} \in JJ_x$, we will have:

$$m^J_k(y^*,y) = \begin{cases} y_k & k \neq j \\ y^*_k & k = j \end{cases}$$

Under this approach for the pair $(x,y) \in D$ only L_x terms are summed in the SWVP update rule. We leave a further investigation of JJ selection approaches to future research.

6 Experiments

Synthetic Data We experiment with synthetic data generated by a linear-chain, first-

[5]For the aggressive approach the equations for schemes (1) and (2) are changed such that JJ_x is replaced with $I(y^*,y,JJ_x)^v$.

order Hidden Markov Model (HMM, (Rabiner and Juang, 1986)). Our learning algorithm is a liner-chain conditional random field (CRF, (Lafferty et al., 2001)): $P(y|x) = \frac{1}{Z(x)} \prod_{i=1:L_x} exp(w \cdot \phi(y_{i-1}, y_i, x))$ (where $Z(x)$ is a normalization factor) with binary indicator features $\{x_i, y_i, y_{i-1}, (x_i, y_i), (y_i, y_{i-1}), (x_i, y_i, y_{i-1})\}$ for the triplet (y_i, y_{i-1}, x).

A dataset is generated by iteratively sampling K items, each is sampled as follows. We first sample a hidden state, y_1, from a uniform prior distribution. Then, iteratively, for $i = 1, 2, \ldots, L_x$ we sample an observed state from the emission probability and (for $i < L_x$) a hidden state from the transition probability. We experimented in 3 setups. In each setup we generated 10 datasets that were subsequently divided to a 7000 items training set, a 2000 items development set and a 1000 items test set. In all datasets, for each item, we set $L_x = 8$. We experiment in three conditions: (1) simple(++), learnable(+++), (2) simple(++), learnable(++) and (3) simple(+), learnable(+).[6]

For each dataset (3 setups, 10 datasets per setup) we train variants of the SWVP algorithm differing in the γ selection strategy (WM or WMR, Section 5), being aggressive (A) or passive (B), and in their β parameter ($\beta = \{0.5, 1, \ldots, 5\}$). Training is done on the training subset and the best performing variant on the development subset is applied to the test subset. For CSP no development set is employed as there is no hyper-parameter to tune. We report averaged accuracy (fraction of observed states for which the model successfully predicts the hidden state value) across the test sets, together with the standard deviation.

Dependency Parsing We also report initial dependency parsing results. We implemented our algorithms within the TurboParser (Martins et al., 2013). That is, every other aspect of the parser: feature set, probabilistic pruning algorithm, inference algorithm etc., is kept fixed but training is performed with SWVP. We compare our results to the parser performance with CSP training (which comes with the standard implementation of the parser).

We experiment with the datasets of the CoNLL 2007 shared task on multilingual dependency parsing (Nilsson et al., 2007), for a total of 9 languages. We followed the standard train/test split of these dataset. For SWVP, we randomly sampled 1000 sentences from each training set to serve as development sets and tuned the parameters as in the synthetic data experiments. CSP is trained on the training set and applied to the test set without any development set involved. We report the Unlabeled Attachment Score (UAS) for each language and model.

7 Results

Synthetic Data Table 1 presents our results. In all three setups an SWVP algorithm is superior. Averaged accuracy differences between the best performing algorithms and CSP are: 3.72 (B-WMR, (simple(++), learnable(+++))), 5.29 (B-WM, (simple(++), learnable(++))) and 5.18 (A-WM, (simple(+), learnable(+))). In all setups SWVP outperforms CSP in terms of averaged performance (except from B-WMR for (simple(+), learnable(+))). Moreover, the weighted models are more stable than CSP, as indicated by the lower standard deviation of their accuracy scores. Finally, for the more simple and learnable datasets the SWVP models outperform CSP in the majority of cases (7-10/10).

We measure generalization from development to test data in two ways. First, for each SWVP algorithm we count the number of times its β parameter results in an algorithm that outperforms the CSP on the development set but not on the test set (not shown in the table). Of the 120 comparisons reported in the table (4 SWVP models, 3 setups, 10 comparisons per model/setup combination) this happened once (A-MV, (simple(++), learnable(+++)).

Second, we count the number of times the best development set value of the β hyper-parameter is also the best value on the test set, or the test set accuracy with the best development set β is at most 0.5% lower than that with the best test set β. The *Gener-*

[6] Denoting $D_x = [C_x]$, $D_y = [C_y]$, and a permutation of a vector v with $perm(v)$, the parameters of the different setups are: (1) simple(++), learnable(+++): $C_x = 5$, $C_y = 3$, $P(y'|y) = perm(0.7, 0.2, 0.1)$, $P(x|y) = perm(0.75, 0.1, 0.05, 0.05, 0.05)$. (2) simple(++), learnable(++): $C_x = 5$, $C_y = 3$, $P(y'|y) = perm(0.5, 0.3, 0.2)$, $P(x|y) = perm(0.6, 0.15, 0.1, 0.1, 0.05)$. (3) simple(+), learnable(+): $C_x = 20$, $C_y = 7$, $P(y'|y) = perm(0.7, 0.2, 0.1, 0, \ldots, 0))$, $P(x|y) = perm(0.4, 0.2, 0.1, 0.1, 0.1, 0, \ldots, 0)$.

Model	simple(++), learnable(+++)			simple(++), learnable(++)			simple(+), learnable(+)		
	Acc. (std)	# Wins	Gener.	Acc. (std)	# Wins	Gener.	Acc. (std)	# Wins	Gener.
B-WM	75.47(3.05)	**9/10**	10/10	**63.18 (1.32)**	**9/10**	10/10	28.48 (1.9)	5/10	10/10
B-WMR	**75.96 (2.42)**	8/10	10/10	63.02 (2.49)	9/10	10/10	24.31 (5.2)	4/10	10/10
A-WM	74.18 (2.16)	7/10	10/10	61.65 (2.30)	**9/10**	10/10	**30.45 (1.0)**	**6/10**	10/10
A-WMR	75.17 (3.07)	7/10	10/10	61.02 (1.93)	8/10	10/10	25.8 (3.18)	2/10	10/10
CSP	72.24 (3.45)	NA	NA	57.89 (2.85)	NA	NA	25.27(8.55)	NA	NA

Table 1: Overall Synthetic Data Results. *A-* and *B-* denote an aggressive and a balanced approaches, respectively. Acc. (std) is the average and the standard deviation of the accuracy across 10 test sets. # Wins is the number of test sets on which the SWVP algorithm outperforms CSP. Gener. is the number of times the best β hyper-parameter value on the development set is also the best value on the test set, or the test set accuracy with the best development set β is at most 0.5% lower than that with the best test set β.

Language	First Order				Second Order			
	CSP	B-WM	Top B-WM	Test B-WM	CSP	B-WM	Top B-WM	Test B-WM
English	86.34	86.4	**86.7**	86.7	**88.02**	87.82	87.82	87.92
Chinese	84.60	84.5	**85.04**	85.05	86.82	86.69	**86.83**	87.02
Arabic	79.09	79.17	**79.21**	79.21	76.07	75.94	**76.09**	76.09
Greek	**80.41**	80.20	80.28	80.28	80.31	**80.40**	**80.40**	80.61
Italian	84.63	84.64	**84.74**	84.70	84.03	84.08	**84.15**	84.28
Turkish	**83.05**	82.89	82.89	82.89	83.02	**83.04**	**83.04**	83.31
Basque	79.47	**79.54**	**79.54**	79.54	80.52	80.57	**80.63**	80.64
Catalan	**88.51**	88.46	88.50	88.5	88.71	**88.81**	**88.81**	88.82
Hungarian	**80.17**	80.07	80.07	80.21	**80.61**	80.45	80.45	80.55
Average	83.69	83.65	**83.77**	83.79	83.12	83.08	**83.13**	83.35

Table 2: First and second order dependency parsing UAS results for CSP trained models, as well as for models trained with SWVP with a balanced γ selection (B) and with a weighted margin (WM) strategy. For explanation of the B-WM, Top B-WM, and Test B-WM see text. For each language and parsing order we highlight the best result in bold font, but this do not include results from Test B-WM as it is provided only as an upper bound on the performance of SWVP.

alization column of the table shows that this has not happened in all of the 120 runs of SWVP.

Dependency Parsing Results are given in Table 2. For the SWVP trained models we report three numbers: (a) B-WM is the standard setup where the β hyper parameter is tuned on the development data; (b) For Top B-WM we first selected the models with a UAS score within 0.1% of the best development data result, and of these we report the UAS of the model that performs best on the test set; and (c) Test B-WM reports results when β is tuned on the test set. This measure provides an upper bound on SWVP with our simplistic JJ (Section 5).

Our results indicate the potential of SWVP. Despite our simple JJ set, Top B-WM and Test B-WM improve over CSP in 5/9 and 6/9 cases in first order parsing, respectively, and in 7/9 cases in second order parsing. In the latter case, Test B-WM improves the UAS over CSP in 0.22% on average across languages. Unfortunately, SWVP still does not generalize well from train to test data as indicated, e.g., by the modest improvements B-WM achieves over CSP in only 5 of 9 languages in second order parsing.

8 Conclusions

We presented the Structured Weighted Violations Perceptron (SWVP) algorithm, a generalization of the Structured Perceptron (CSP) algorithm that explicitly exploits the internal structure of the predicted label in its update rule. We proved the convergence of the algorithm for linearly separable training sets under certain conditions on its parameters, and provided generalization and mistake bounds.

In experiments we explored only very simple configurations of the SWVP parameters - γ and JJ. Nevertheless, several of our SWVP variants outperformed the CSP special case in synthetic data experiments. In dependency parsing experiments, SWVP demonstrated some improvements over CSP, but these do not generalize well. While we find these results somewhat encouraging, they emphasize the need to explore the much more flexible γ and JJ selection strategies allowed by SWVP (Sec. 4.2). In future work we will hence develop γ and JJ selection algorithms, where selection is ideally performed jointly with inference (property 2, Sec. 4.2), to make SWVP practically useful in NLP applications.

A Proof Observation 1.

Proof. For every training example $(x, y) \in D$, it holds that: $\cup_{z \in \mathcal{Y}(x)} m^J(z, y) \subseteq \mathcal{Y}(x)$. As $\mathbf{u}$ separates the data with margin δ, it holds that:

$$\mathbf{u} \cdot \Delta\phi(x, y, m^J(z, y)) \geq \delta^{JJ_x}, \quad \forall z \in \mathcal{Y}(x), J \in JJ_x.$$
$$\mathbf{u} \cdot \Delta\phi(x, y, z) \geq \delta, \quad \forall z \in \mathcal{Y}(x).$$

Therefore also $\delta^{JJ_x} \geq \delta$. As the last inequality holds for every $(x, y) \in D$ we get that $\delta^{JJ} = \min_{(x,y)\in D} \delta^{JJ_x} \geq \delta$.
From the same considerations it holds that $R^{JJ} \leq R$. This is because R^{JJ} is the radius of a subset of the dataset with radius R (proper subset if $\exists (x, y) \in D, [L_x] \notin JJ_x$, non-proper subset otherwise). □

B Mistake Bound - Non Separable Case

Here we provide a mistake bound for the algorithm in the non-separable case. We start with the following definition and observation:

Definition 8. *Given an example $(x^i, y^i) \in D$, for a $\boldsymbol{u}, \delta$ pair define:*

$$r^i = \boldsymbol{u} \cdot \phi(x^i, y^i) - \max_{z \in \mathcal{Y}(x^i)} \boldsymbol{u} \cdot \phi(x^i, z)$$
$$\epsilon_i = \max\{0, \delta - r^i\}$$
$$r^{iJJ} = \boldsymbol{u} \cdot \phi(x^i, y^i) - \max_{z \in \mathcal{Y}(x^i), J \in JJ_{x^i}} \boldsymbol{u} \cdot \phi(x^i, m^J(z, y^i))$$

Finally define: $D_{\boldsymbol{u},\delta} = \sqrt{\sum_{i=1}^{n} \epsilon_i^2}$

Observation 2. For all i: $r^i \leq r^{iJJ}$.
Observation 2 easily follows from Definition 8. Following this observation we denote: $r^{diff} = \min_i\{r^{iJJ} - r^i\} \geq 0$ and present the next theorem:

Theorem 2. *For any training sequence D, for the **first** pass over the training set of the CSP and the SWVP algorithms respectively, it holds that:*

$$\#mistakes - CSP \leq \min_{\boldsymbol{u}:\|\boldsymbol{u}\|=1,\delta>0} \frac{(R + D_{\boldsymbol{u},\delta})^2}{\delta^2}.$$
$$\#mistakes - SWVP \leq \min_{\boldsymbol{u}:\|\boldsymbol{u}\|=1,\delta>0} \frac{(R^{JJ} + D_{\boldsymbol{u},\delta})^2}{(\delta + r^{diff})^2}.$$

As $R^{JJ} \leq R$ (Observation 1) and $r^{diff} \geq 0$, we get a tighter bound for SWVP. The proof for #mistakes-CSP is given at (Collins, 2002). The proof for #mistakes-SWVP is given below.

Proof. We transform the representation $\phi(x, y) \in \mathbb{R}^d$ into a new representation $\psi(x, y) \in \mathbb{R}^{d+n}$ as follows: for $i = 1, ..., d : \psi_i(x, y) = \phi_i(x, y)$, for $j = 1, ..., n : \psi_{d+j}(x, y) = \Delta$ if $(x, y) = (x^j, y^j)$ and 0 otherwise, where $\Delta > 0$ is a parameter.
Given a $\mathbf{u}, \delta$ pair define $\mathbf{v} \in \mathbb{R}^{d+n}$ as follows: for $i = 1, ..., d : \mathbf{v}_i = \mathbf{u}_i$, for $j = 1, ..., n : \mathbf{v}_{d+j} = \frac{\epsilon_j}{\Delta}$.
Under these definitions we have:

$$\mathbf{v} \cdot \psi(x^i, y^i) - \mathbf{v} \cdot \psi(x^i, z) \geq \delta, \quad \forall i, z \in \mathcal{Y}(x^i).$$

For every $i, z \in \mathcal{Y}(x^i), J \in JJ_{x^i}$:

$$\mathbf{v} \cdot \psi(x^i, y^i) - \mathbf{v} \cdot \psi(x^i, m^J(z, y^i)) \geq \delta + r^{diff}.$$
$$\|\psi(x^i, y^i) - \psi(x^i, m^J(z, y^i))\|^2 \leq (R^{JJ})^2 + \Delta^2.$$

Last, we have,

$$\|\mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \sum_{i=1}^{n} \frac{\epsilon_i^2}{\Delta^2} = 1 + \frac{D_{\mathbf{u},\delta}^2}{\Delta^2}.$$

We get that the vector $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ linearly separates the data with respect to single decision assignments with margin $\frac{\delta}{\sqrt{1+\frac{D_{U,\delta}^2}{\Delta^2}}}$. Likewise, $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ linearly separates the data with respect to mixed assignments with JJ, with margin $\frac{\delta + r^{diff}}{\sqrt{1+\frac{D_{\mathbf{u},\delta}}{\Delta^2}}}$. Notice that the **first** pass of SWVP with representation Ψ is identical to the first pass with representation Φ because the parameter weight for the additional features affects only a single example of the training data and do not affect the classification of test examples. By theorem 1 this means that the **first** pass of SWVP with representation Ψ makes at most $\frac{((R^{JJ})^2+\Delta^2)}{(\delta+r^{diff})^2} \cdot \left(1 + \frac{D_{\mathbf{u},\delta}^2}{\Delta^2}\right)$.
We minimize this w.r.t Δ, which gives: $\Delta = \sqrt{R^{JJ} D_{\mathbf{u},\delta}}$, and obtain the result guaranteed in the theorem. □

Acknowledgments

The second author was partly supported by a research grant from the GIF Young Scientists' Program (No. I-2388-407.6/2015): Syntactic Parsing in Context.

References

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL*, pages 16–23.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, pages 1–8.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of ICML*, pages 169–176.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proc. of NAACL-HLT 2010*, pages 742–750.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proc. of ACL*.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. of NAACL-HLT*, pages 142–151.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*, pages 1–11.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

André FT Martins, Miguel Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Prc. of ACL short papers*, pages 617–622.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*, pages 91–98.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP-HLT*, pages 523–530.

Ofer Meshi, David Sontag, Tommi Jaakkola, and Amir Globerson. 2010. Learning efficiently with approximate inference via dual losses. In *Proc. of ICML*.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn.

Lawrence Rabiner and Biing-Hwang Juang. 1986. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16.

Roi Reichart and Regina Barzilay. 2012. Multi event extraction guided by global constraints. In *Proc. of NAACL-HLT 2012*, pages 70–79.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Proc. of NIPS*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proc. of EMNLP-CoNLL*, pages 678–687.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *proc. of ACL*, pages 888–896.

How Transferable are Neural Networks in NLP Applications?

Lili Mou,[1] **Zhao Meng,**[1] **Rui Yan,**[2] **Ge Li,**[1,†] **Yan Xu,**[1,*] **Lu Zhang,**[1] **Zhi Jin**[1,†]
[1]Key Laboratory of High Confidence Software Technologies (Peking University), MoE, China
Institute of Software, Peking University, China †Corresponding authors
[2]Insitute of Computer Science and Technology of Peking University, China
{doublepower.mou,rui.yan.peking}@gmail.com,zhaomeng.pku@outlook.com
{lige,xuyan14,zhanglu,zhijin}@sei.pku.edu.cn

Abstract

Transfer learning is aimed to make use of valuable knowledge in a *source* domain to help model performance in a *target* domain. It is particularly important to neural networks, which are very likely to be overfitting. In some fields like image processing, many studies have shown the effectiveness of neural network-based transfer learning. For neural NLP, however, existing studies have only casually applied transfer learning, and conclusions are inconsistent. In this paper, we conduct systematic case studies and provide an illuminating picture on the transferability of neural networks in NLP.[1]

1 Introduction

Transfer learning, or sometimes known as domain adaptation,[2] plays an important role in various natural language processing (NLP) applications, especially when we do not have large enough datasets for the task of interest (called the *target* task $\mathcal{T}$). In such scenarios, we would like to transfer or adapt knowledge from other domains (called the *source* domains/tasks $\mathcal{S}$) so as to mitigate the problem of overfitting and to improve model performance in $\mathcal{T}$. For traditional feature-rich or kernel-based models, researchers have developed a variety of elegant methods for domain adaptation; examples include EasyAdapt (Daumé III, 2007; Daumé III et al., 2010), instance weighting (Jiang and Zhai, 2007; Foster et al., 2010), and structural correspondence learning (Blitzer et al., 2006; Prettenhofer and Stein, 2010).

Recently, deep neural networks are emerging as the prevailing technical solution to almost every field in NLP. Although capable of learning highly nonlinear features, deep neural networks are very prone to overfitting, compared with traditional methods. Transfer learning therefore becomes even more important. Fortunately, neural networks can be trained in a transferable way by their incremental learning nature: we can directly use trained (tuned) parameters from a source task to initialize the network in the target task; alternatively, we may also train two tasks simultaneously with some parameters shared. But their performance should be verified by empirical experiments.

Existing studies have already shown some evidence of the transferability of neural features. For example, in image processing, low-level neural layers closely resemble Gabor filters or color blobs (Zeiler and Fergus, 2014; Krizhevsky et al., 2012); they can be transferred well to different tasks. Donahue et al. (2014) suggest that high-level layers are also transferable in general visual recognition; Yosinski et al. (2014) further investigate the transferability of neural layers in different levels of abstraction.

Although transfer learning is promising in image processing, conclusions appear to be less clear in NLP applications. Image pixels are low-level signals, which are generally continuous and less related to semantics. By contrast, natural language tokens

*Yan Xu is currently a research scientist at Inveno Co., Ltd.

[1]Code released on https://sites.google.com/site/transfernlp/

[2]In this paper, we do not distinguish the conceptual difference between *transfer learning* and *domain adaptation*. *Domain*—in the sense we use throughout this paper—is defined by datasets.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 479–489,
Austin, Texas, November 1-5, 2016.

are discrete: each word well reflects the thought of humans, but neighboring words do not share as much information as pixels in images do. Previous neural NLP studies have casually applied transfer techniques, but their results are not consistent. Collobert and Weston (2008) apply multi-task learning to SRL, NER, POS, and CHK,[3] but obtain only 0.04–0.21% error reduction[4] (out of a base error rate of 16–18%). Bowman et al. (2015), on the contrary, improve a natural language inference task from an accuracy of 71.3% to 80.8% by initializing parameters with an additional dataset of 550,000 samples. Therefore, more systematic studies are needed to shed light on transferring neural networks in the field of NLP.

Our Contributions

In this paper, we investigate the question "*How transferable are neural networks in NLP applications?*"

We distinguish two scenarios of transfer: (1) transferring knowledge to a semantically similar/equivalent task but with a different dataset; (2) transferring knowledge to a task that is semantically different but shares the same neural topology/architecture so that neural parameters can indeed be transferred. We further distinguish two transfer methods: (1) using the parameters trained on $\mathcal{S}$ to initialize $\mathcal{T}$ (INIT), and (2) multi-task learning (MULT), i.e., training $\mathcal{S}$ and $\mathcal{T}$ simultaneously. (Please see Sections 2 and 4). Our study mainly focuses on the following research questions:

RQ1: How transferable are neural networks between two tasks with similar or different semantics in NLP applications?

RQ2: How transferable are different layers of NLP neural models?

RQ3: How transferable are INIT and MULT, respectively? What is the effect of combining these two methods?

[3]The acronyms refer to *semantic role labeling*, *named entity recognition*, *part-of-speech tagging*, and *chunking*, respectively.

[4]Here, we quote the accuracies obtained by using unsupervised pretraining of word embeddings. This is the highest performance in that paper; using pretrained word embeddings is also a common practice in the literature.

We conducted extensive experiments over six datasets on classifying sentences and sentence pairs. We leveraged the widely-used convolutional neural network (CNN) and long short term memory (LSTM)-based recurrent neural network (RNN) as our models.

Based on our experimental results, we have the following main observations, some of which are unexpected.

- Whether a neural network is transferable in NLP depends largely on how semantically similar the tasks are, which is different from the consensus in image processing.
- The output layer is mainly specific to the dataset and not transferable. Word embeddings are likely to be transferable to semantically different tasks.
- MULT and INIT appear to be generally comparable to each other; combining these two methods does not result in further gain in our study.

The rest of this paper is organized as follows. Section 2 introduces the datasets that neural models are transferred across; Section 3 details the neural architectures and experimental settings. We describe two approaches (INIT and MULT) to transfer learning in Section 4. We present experimental results in Sections 5–6 and have concluding remarks in Section 7.

2 Datasets

In our study, we conducted two series of experiments using six open datasets as follows.

- **Experiment I**: Sentence classification
  - `IMDB`. A large dataset for binary sentiment classification (`positive` vs. `negative`).[5]
  - `MR`. A small dataset for binary sentiment classification.[6]
  - `QC`. A (small) dataset for 6-way question classification (e.g., `location`, `time`, and `number`).[7]

[5]https://drive.google.com/file/d/0B8yp1gOBCztyN0JaMDVoeXhHWm8/

[6]https://www.cs.cornell.edu/people/pabo/movie-review-data/

[7]http://cogcomp.cs.illinois.edu/Data/QA/QC/

Statistics (# of Samples)						
	Experiment I			Experiment II		
	IMDB	MR	QC	SNLI	SICK	MSRP
#Train	550,000	8,500	4,800	550,152	4,439	3,575
#Val	50,000	1,100	600	10,000	495	501
#Test	2,000	1,100	500	10,000	4,906	1,725

Examples in Experiment I	
Sentiment Analysis (IMDB and MR)	
An idealistic love story that brings out the latent 15-year-old romantic in everyone.	+
Its mysteries are transparently obvious, and its too slowly paced to be a thriller.	−
Question Classification (QC)	
What is the temperature at the center of the earth?	number
What state did the Battle of Bighorn take place in?	location

Examples in Experiment II		
Natural Language Inference (SNLI and SICK)		
Premise	Two men on bicycles competing in a race.	
Hypothesis	People are riding bikes.	E
	Men are riding bicycles on the streets.	C
	A few people are catching fish.	N
Paraphrase Detection (MSRP)		
The DVD-CCA then appealed to the state Supreme Court.		Paraphrase
The DVD CCA appealed that decision to the U.S. Supreme Court.		
Earnings per share from recurring operations will be 13 cents to 14 cents.		Non-Paraphrase
That beat the company's April earnings forecast of 8 to 9 cents a share.		

Table 1: Statistics and examples of the datasets.

- **Experiment II**: Sentence-pair classification
  - SNLI. A large dataset for sentence entailment recognition. The classification objectives are entailment, contradiction, and neutral.[8]
  - SICK. A small dataset with exactly the same classification objective as SNLI.[9]
  - MSRP. A (small) dataset for paraphrase detection. The objective is binary classification: judging whether two sentences have the same meaning.[10]

In each experiment, the large dataset serves as the source domain and small ones are the target domains. Table 1 presents statistics of the above datasets.

We distinguish two scenarios of transfer regarding semantic similarity: (1) semantically equivalent transfer (IMDB→MR, SNLI→SICK), that is, the tasks of $\mathcal{S}$ and $\mathcal{T}$ are defined by the same meaning, and (2) semantically different transfer (IMDB→QC, SNLI→MSRP). Examples are also illustrated in Table 1 to demonstrate semantic relatedness.

It should be noticed that in image or speech processing (Yosinski et al., 2014; Wang and Zheng, 2015), the input of neural networks pretty much consists of raw signals; hence, low-level feature detectors are almost always transferable, even if Yosinski et al. (2014) manually distinguish artificial objects and natural ones in an image classification task.

Distinguishing semantic relatedness—which emerges from very low layers of either word embeddings or the successive hidden layer—is specific to NLP and also a new insight of our paper. As we shall see in Sections 5 and 6, the transferability of neural networks in NLP is more sensitive to semantics than in image processing.

3 Neural Models and Settings

In each group, we used a single neural model to solve three problems in a unified manner. That is to say, the neural architecture is the same among the three datasets, which makes it possible to investigate transfer learning regardless of whether the tasks are semantically equivalent. Concretely, the neural models are as follows.

- **Experiment I: LSTM-RNN**. To classify a sentence according to its sentiment or question type, we use a recurrent neural network (RNN, Figure 1a) with long short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). A softmax layer is added to the last word's hidden state for classification.

- **Experiment II: CNN-pair**. In this group, we use a "Siamese" architecture (Bromley et al., 1993) to classify the relation of two sentences. We first apply a convolutional neural network (CNN, Figure 1b) with a window size of 5 to model local context, and a max pooling layer gathers information to a fixed-size vector. Then the sentence vectors are concatenated and fed to a hidden layer before the softmax output.

In our experiments, embeddings were pretrained by word2vec (Mikolov et al., 2013); all embeddings and hidden layers were 100 dimensional. We

[8] http://nlp.stanford.edu/projects/snli/

[9] http://http://alt.qcri.org/semeval2014/task1/

[10] http://research.microsoft.com/en-us/downloads/

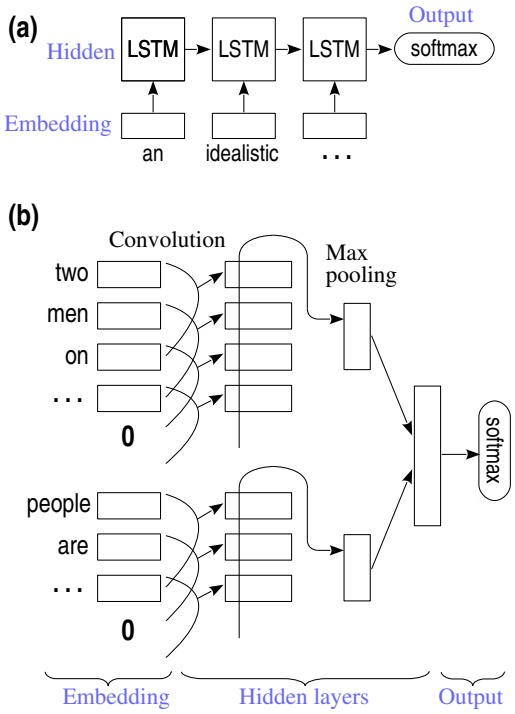


Figure 1: The models in our study. (a) Experiment I: RNNs with LSTM units for sentence classification. (b) Experiment II: CNN for sentence pair modeling.

applied stochastic gradient descent with a mini-batch size of 50 for optimization. In each setting, we tuned the hyperparameters as follows: learning rate from $\{3, 1, 0.3, 0.1, 0.03\}$, power decay of learning rate from {fast, moderate, low} (defined by how much, after one epoch, the learning rate residual is: 0.1x, 0.3x, 0.9x, resp). We regularized our network by dropout with a rate from $\{0, 0.1, 0.2, 0.3\}$. Note that we might not run nonsensical settings, e.g., a larger dropout rate if the network has already been underfitting (i.e., accuracy has decreased when the dropout rate increases). We report the test performance associated with the highest validation accuracy.

To setup a baseline, we trained our models without transfer 5 times by different random parameter initializations (Table 2). We have achieved reasonable performance that is comparable to similar models reported in the literature with all six datasets. Therefore, our implementation is fair and suitable for further study of transfer learning.

It should be mentioned that the goal of this paper is not to outperform state-of-the-art results; instead, we would like to conduct a fair comparison of different methods and settings for transfer learning in NLP.

	Dataset	Avg acc.±std.	Related model
Exp. I	IMDB	87.0	89.3 (Non-NN, Dong$^+$,2015)
	MR	75.1 ± 0.6	77.7 (RAE, Socher$^+$, 2013)
	QC	90.8 ± 0.9	90.2 (RNN, Zhao$^+$,2015)
Exp. II	SNLI	76.3	77.6 (RNN, Bowman$^+$,2015)
	SICK	70.9 ± 1.3	71.3 (RNN, Bowman$^+$,2015)
	MSRP	69.0 ± 0.5	69.6 (Arc-I CNN, Hu$^+$,2014)

Table 2: Accuracy (%) without transfer. We also include related models for comparison (Dong et al., 2015; Socher et al., 2011; Zhao et al., 2015; Bowman et al., 2015; Hu et al., 2014), showing that we have achieved comparable results, and thus are ready to investigate transfer learning. The models were run one only once in source domains, because we could only transfer a particular model instead of an average of several models.

4 Transfer Methods

Transfer learning aims to use knowledge in a source domain to aid the target domain. As neural networks are usually trained incrementally with gradient descent (or variants), it is straightforward to use gradient information in both source and target domains for optimization so as to accomplish knowledge transfer. Depending on how samples in source and target domains are scheduled, there are two main approaches to neural network-based transfer learning:

- Parameter initialization (INIT). The INIT approach first trains the network on $\mathcal{S}$, and then directly uses the tuned parameters to initialize the network for $\mathcal{T}$. After transfer, we may fix (🔒) the parameters in the target domain (Glorot et al., 2011), i.e., no training is performed on $\mathcal{T}$. But when labeled data are available in $\mathcal{T}$, it would be better to fine-tune (🔓) the parameters.

  INIT is also related to unsupervised pretraining such as word embedding learning (Mikolov et al., 2013) and autoencoders (Bengio et al., 2006). In these approaches, parameters that are (pre)trained in an unsupervised way are transferred to initialize the model for a supervised task (Plank and Moschitti, 2013). However, our paper focuses on "supervised pretraining," which means we transfer knowledge from a labeled source domain.

- Multi-task learning (MULT). MULT, on the other hand, simultaneously trains samples in both domains (Collobert and Weston, 2008; Liu et al., 2016). The overall cost function is given by

$$J = \lambda J_{\mathcal{T}} + (1 - \lambda) J_{\mathcal{S}} \quad (1)$$

  where $J_{\mathcal{T}}$ and $J_{\mathcal{S}}$ are the individual cost function of each domain. (Both $J_{\mathcal{T}}$ and $J_{\mathcal{S}}$ are normalized by the number of training samples.) $\lambda \in (0, 1)$ is a hyperparameter balancing the two domains.

  It is nontrivial to optimize Equation 1 in practice by gradient-based methods. One may take the partial derivative of J and thus λ goes to the learning rate (Liu et al., 2016), but the model is then vulnerable because it is likely to blow up with large learning rates (multiplied by λ or $1 - \lambda$) and be stuck in local optima with small ones.

  Collobert and Weston (2008) alternatively choose a data sample from either domain with a certain probability (controlled by λ) and take the derivative for the particular data sample. In this way, domain transfer is independent of learning rates, but we may not be able to fully use the entire dataset of $\mathcal{S}$ if λ is large. We adopted the latter approach in our experiment for simplicity. (More in-depth analysis may be needed in future work.) Formally, our multi-task learning strategy is as follows.

  1 Switch to $\mathcal{T}$ with prob. λ, or to $\mathcal{S}$ with prob. $1 - \lambda$.
  2 Compute the gradient of the next data sample in the particular domain.

Further, INIT and MULT can be combined straightforwardly, and we obtain the third setting:

- Combination (MULT+INIT). We first pretrain on the source domain $\mathcal{S}$ for parameter initialization, and then train $\mathcal{S}$ and $\mathcal{T}$ simultaneously.

From a theoretical perspective, INIT and MULT work in different ways. In the MULT approach, the source domain regularizes the model by "aliasing" the error surface of the target domain; hence the neural network is less prone to overfitting. In INIT, $\mathcal{T}$'s error surface remains intact. Before training on the target dataset, the parameters are initialized in such a meaningful way that they contain additional knowledge in the source domain. However, in an extreme case where $\mathcal{T}$'s error surface is convex, INIT is ineffective because the parameters can reach the global optimum regardless of their initialization. In practice, deep neural networks usually have highly complicated, non-convex error surfaces. By properly initializing parameters with the knowledge of $\mathcal{S}$, we can reasonably expect that the parameters are in a better "catchment basin," and that the INIT approach can transfer knowledge from $\mathcal{S}$ to $\mathcal{T}$.

5 Results of Transferring by INIT

We first analyze how INIT behaves in NLP-based transfer learning. In addition to two different transfer scenarios regarding semantic relatedness as described in Section 2, we further evaluated two settings: (1) fine-tuning parameters 🔓, and (2) freezing parameters after transfer 🔒. Existing evidence shows that frozen parameters would generally hurt the performance (Peng et al., 2015), but this setting provides a more direct understanding on how transferable the features are (because the factor of target domain optimization is ruled out). Therefore, we included it in our experiments. Moreover, we transferred parameters layer by layer to answer our second research question.

Through Subsections 5.1–5.3, we initialized the parameters of $\mathcal{T}$ with the ones corresponding to the highest validation accuracy of $\mathcal{S}$. In Subsection 5.4, we further investigated when the parameters are ready to be transferred during the training on $\mathcal{S}$.

5.1 Overall Performance

Table 3 shows the main results of INIT. A quick observation is that, in both groups, transfer learning of semantically equivalent tasks (IMDB→MR, SNLI→SICK) appears to be successful with an improvement of ∼6%. The results are not surprising and also reported in Bowman et al. (2015).

For IMDB→QC and SNLI→MSRP, however, there is no improvement of transferring hidden layers (embeddings excluded), namely LSTM-RNN units and CNN feature maps. The E🔓H🔓O□ setting yields a slight degradation of 0.2–0.4%, ∼.5x std. The incapability of transferring is also proved by locking embeddings and hidden layers

(E🔒H🔒O□). We see in this setting, the test performance is very low in QC or even worse than majority-class guess in MSRP. By further examining its training accuracy, which is 48.2% and 65.5%, respectively, we conclude that extracted features by LSTM-RNN and CNN models in $\mathcal{S}$ are almost irrelevant to the ultimate tasks $\mathcal{T}$ (QC and MSRP).

Although in previous studies, researchers have mainly drawn positive conclusions about transfer learning, we find a negative result similar to ours upon careful examination of Collobert and Weston (2008), and unfortunately, their results may be somewhat misinterpreted. In that paper, the authors report transferring NER, POS, CHK, and pretrained word embeddings improves the SRL task by 1.91–3.90% accuracy (out of 16.54–18.40% error rate), but their gain is mainly due to word embeddings. In the settings that use pretrained word embeddings (which is common in NLP), NER, POS, and CHK together improve the SRL accuracy by only 0.04–0.21%.

The above results are rather frustrating, indicating for RQ1 that neural networks may not be transferable to NLP tasks of different semantics. Transfer learning for NLP is more prone to semantics than the image processing domain, where even high-level feature detectors are almost always transferable (Donahue et al., 2014; Yosinski et al., 2014).

Experiment I		
Setting	IMDB→MR	IMDB→QC
Majority	50.0	22.9
E⊠ H□ O□	75.1	90.8
E🔒 H□ O□	78.2	93.2
E🔒 H🔒 O□	78.8	55.6
E🔒 H🔒 O🔒	73.6	–
E🔓 H□ O□	78.3	92.6
E🔓 H🔓 O□	81.4	90.4
E🔓 H🔓 O🔓	80.9	–
Experiment II		
Setting	SNLI→SICK	SNLI→MSRP
Majority	56.9	66.5
E⊠ H□ O□	70.9	69.0
E🔒 H□ O□	69.3	68.1
E🔒 H🔒 O□	70.0	66.4
E🔒 H🔒 O🔒	43.1	–
E🔓 H□ O□	71.0	69.9
E🔓 H🔓 O□	76.3	68.8
E🔓 H🔓 O🔓	77.6	–

Table 3: Main results of neural transfer learning by INIT. We report test accuracies (%) in this table. E: embedding layer; H: hidden layers; O: output layer. ⊠: Word embeddings are pretrained by word2vec; □: Parameters are randomly initialized); 🔒: Parameters are transferred but frozen; 🔓: Parameters are transferred and fine-tuned. Notice that the E🔒H🔒O🔒 and E🔓H🔓O🔓 settings are inapplicable to IMDB→QC and SNLI→MSRP, because the output targets do not share same meanings and numbers of target classes.

5.2 Layer-by-Layer Analysis

To answer RQ2, we next analyze the transferability of each layer. First, we freeze both embeddings and hidden layers (E🔒H🔒). Even in semantically equivalent settings, if we further freeze the output layer (O🔒), the performance in both IMDB→MR and SNLI→SICK drops, but by randomly initializing the output layer's parameters (O□), we can obtain a similar or higher result compared with the baseline (E⊠H□O□). The finding suggests that the output layer is mainly specific to a dataset. Transferring the output layer's parameters yields little (if any) gain.

Regarding embeddings and hidden layers (in the settings E🔓H🔓O□/E🔓H□O□ vs. E⊠H□O□), the IMDB→MR experiment suggests both of embeddings and the hidden layer play an important role, each improving the accuracy by 3%. In SNLI→SICK, however, the main improvement lies in the hidden layer. A plausible explanation is that in sentiment classification tasks (IMDB and MR), information emerges from raw input, i.e., sentiment lexicons and thus their embeddings, but natural language inference tasks (SNLI and SICK) address more on semantic compositionality and thus hidden layers are more important.

Moreover, for semantically different tasks (IMDB→QC and SNLI→MSRP), the embeddings are the only parameters that have been observed to be transferable, slightly benefiting the target task by 2.7x and 1.8x std, respectively.

5.3 How does learning rate affect transfer?

Bowman et al. (2015) suggest that after transferring, a large learning rate may damage the knowledge stored in the parameters; in their paper, they transfer the learning rate information (AdaDelta) from $\mathcal{S}$ to $\mathcal{T}$ in addition to the parameters.

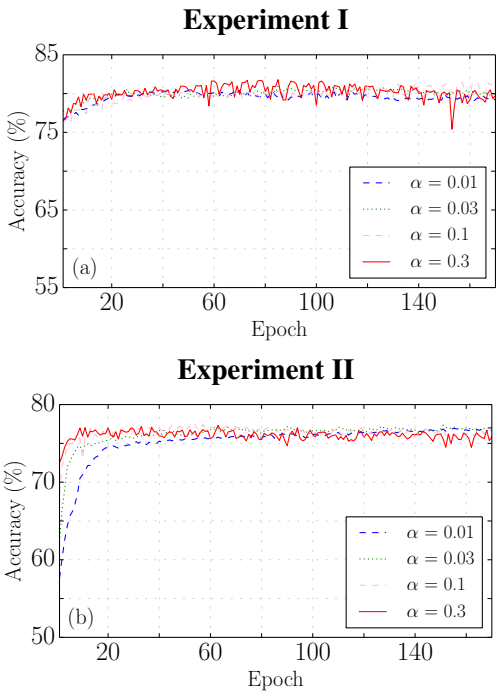


Figure 2: Learning curves of different learning rates (denoted as α). (a) Experiment I: IMDB→MR; (b) Experiment II: SNLI→SICK.

Although the rule of the thumb is to choose all hyperparameters—including the learning rate—by validation, we are curious whether the above conjecture holds. Estimating a rough range of sensible hyperparameters can ease the burden of model selection; it also provides evidence to better understand how transfer learning actually works.

We plot the learning curves of different learning rates α in Figure 2 (IMDB→MR and SNLI→SICK, E🔓H🔓O□). (In the figure, no learning rate decay is applied.) As we see, with a large learning rate like $\alpha = 0.3$, the accuracy increases fast and peaks at earlier epochs. Training with a small learning rate (e.g., $\alpha = 0.01$) is slow, but its peak performance is comparable to large learning rates when iterated by, say, 100 epochs. The learning curves in Figure 2 are similar to classic speed/variance trade-off, and we have the following additional discovery:

> In INIT, transferring learning rate information is not necessarily useful. A large learning rate does not damage the knowledge stored in the pretrained hyperparameters, but accelerates the training process to a large extent. In all, we may need to perform validation to choose the learning rate if computational resources are available.

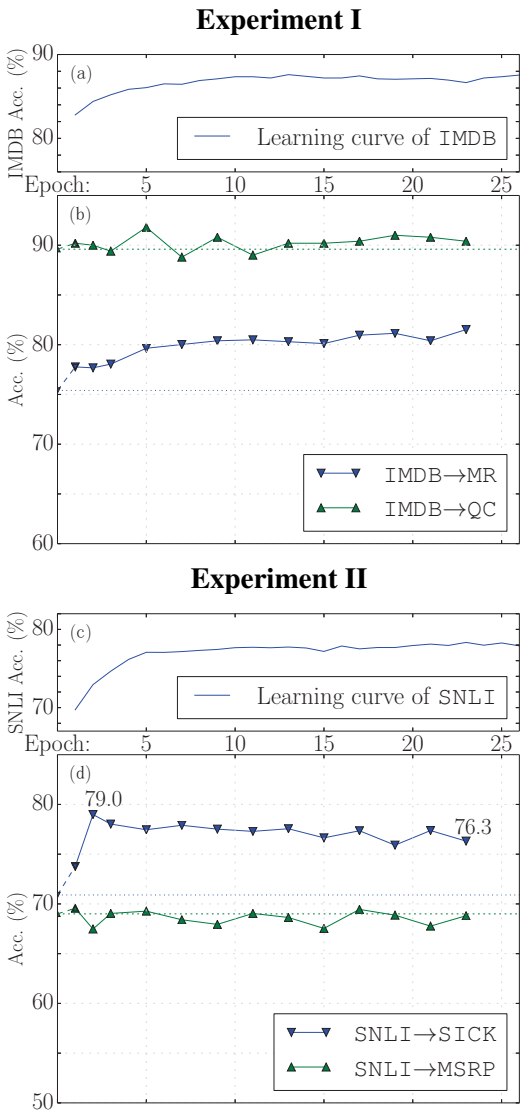


Figure 3: (a) and (c): Learning curves of $\mathcal{S}$. (b) and (d): Accuracies of $\mathcal{T}$ when parameters are transferred at a certain epoch during the training of $\mathcal{S}$. Dotted lines refer to non-transfer, which can be equivalently viewed as transferring before training on $\mathcal{S}$, i.e., epoch $= 0$. Note that the x-axis shares across different subplots.

5.4 When is it ready to transfer?

In the above experiments, we transfer the parameters when they achieve the highest validation performance on $\mathcal{S}$. This is a straightforward and intuitive practice.

However, we may imagine that the parameters well-tuned to the source dataset may be too specific to it, i.e., the model overfits $\mathcal{S}$ and thus may underfit $\mathcal{T}$. Another advantage of early transfer lies in com-

putational concerns. If we manage to transfer model parameters after one or a few epochs on $\mathcal{S}$, we can save much time especially when $\mathcal{S}$ is large.

We therefore made efforts in studying when the neural model is ready to be transferred. Figures 3a and 3c plot the learning curves of the source tasks. The accuracy increases sharply from epochs 1–5; later, it reaches a plateau but is still growing slowly.

We then transferred the parameters at different stages (epochs) of training to target tasks (also with the setting E🔓H🔓O□). Their accuracies are plotted in Figures 3b and 3d.

In IMDB→MR, the source performance and transferring performance align well. The SNLI→SICK experiment, however, produces interesting yet unexpected results. Using the second epoch of SNLI's training yields the highest transfer performance on SICK, i.e., 78.98%, when the SNLI performance itself is comparatively low (72.65% vs. 76.26% at epoch 23). Later, the transfer performance decreases gradually by ∼2.7%. The results in these two experiments are inconsistent and lack explanation.

6 MULT, and its Combination with INIT

To answer RQ3, we investigate how multi-task learning performs in transferring knowledge, as well as the effect of the combination of MULT and INIT. In this section, we applied the setting: sharing embeddings and hidden layers (denoted as E❤H❤O□), analogous to E🔓H🔓O□ in INIT. When combining MULT and INIT, we used the pretrained parameters of embeddings and hidden layers on $\mathcal{S}$ to initialize the multi-task training of $\mathcal{S}$ and $\mathcal{T}$, visually represented by E🔓❤H🔓❤O□.

In both MULT and MULT+INIT, we had a hyperparameter $\lambda \in (0, 1)$ balancing the source and target tasks (defined in Section 4). λ was tuned with a granularity of 0.1. As a friendly reminder, $\lambda = 1$ refers to using $\mathcal{T}$ only; $\lambda = 0$ refers to using $\mathcal{S}$ only. After finding that a small λ yields high performance of MULT in the IMDB+MR and SNLI+SICK experiments (thick blue lines in Figures 4a and 4c), we further tuned the λ from 0.01 to 0.09 with a fine-grained granularity of 0.02.

The results are shown in Figure 4. From the green curves in the 2nd and 4th subplots, we see MULT (with or without INIT) does not improve the accuracy of target tasks (QC and MSRP); the inability to transfer is cross-checked by the INIT method in Section 5. For MR and SICK, on the other hand, transferability of the neural model is also consistently positive (blue curves in Figures 4a and 4c), supporting our conclusion to RQ1 that neural trans-

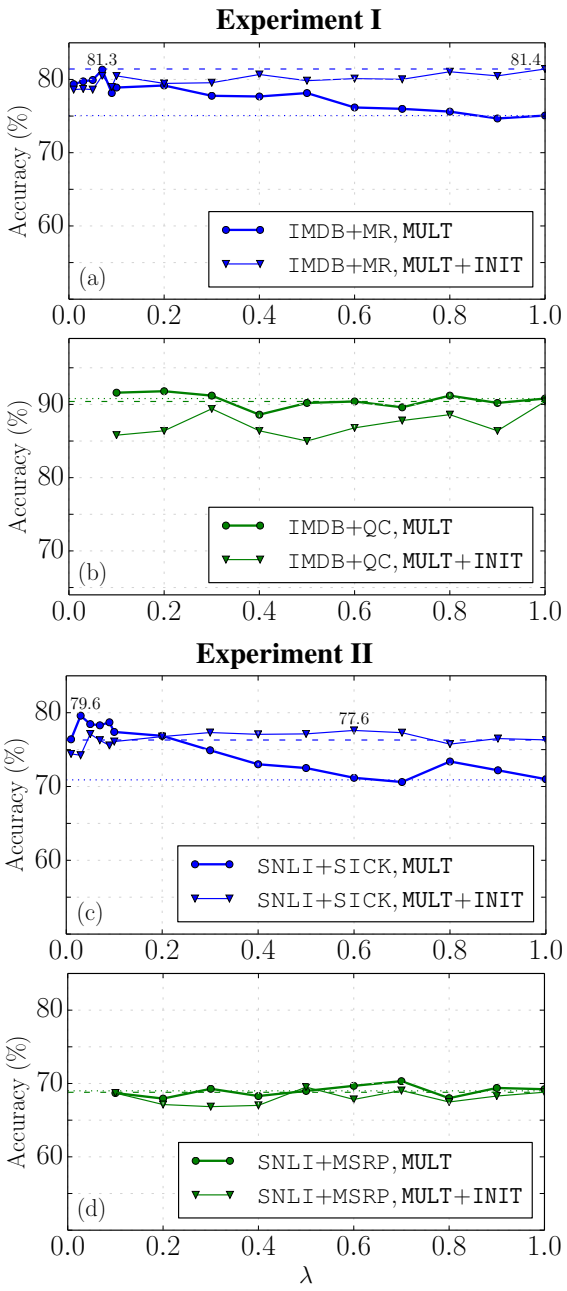


Figure 4: Results of MULT and MULT+INIT, where we share word embeddings and hidden layers. Dotted lines are the non-transfer setting; dashed lines are the INIT setting E🔓H🔓O□, transferred at the peak performance of IMDB and SNLI.

fer learning in NLP depends largely on how similar in semantics the source and target datasets are.

Moreover, we see that the peak performance of MULT is slightly lower than INIT in Experiment I (Figure 4a), but higher in Experiment II (Figure 4c); they are in the same ballpark.

In MULT+INIT (E❤H❤O□), the transfer performance of MULT+INIT remains high for different values of λ. Because the parameters given by INIT have already conveyed sufficient information about the source task, MULT+INIT consistently outperforms non-transferring by a large margin. Its peak performance, however, is not higher than MULT or INIT. In summary, we answer our RQ3 as follows: in our experiments, MULT and INIT are generally comparable; we do not obtain further gain by combining MULT and INIT.

7 Concluding Remarks

In this paper, we addressed the problem of transfer learning in neural network-based NLP applications. We conducted two series of experiments on six datasets, showing that the transferability of neural NLP models depends largely on the semantic relatedness of the source and target tasks, which is different from other domains like image processing. We analyzed the behavior of different neural layers. We also experimented with two transfer methods: parameter initialization (INIT) and multi-task learning (MULT). Besides, we reported two additional studies in Sections 5.3 and 5.4 (not repeated here). Our paper provides insight on the transferability of neural NLP models; the results also help to better understand neural features in general.

How transferable are the conclusions in this paper? We have to concede that empirical studies are subject to a variety of factors (e.g., models, tasks, datasets), and that conclusions may vary in different scenarios. In our paper, we have tested all results on two groups of experiments involving 6 datasets and 2 neural models (CNN and LSTM-RNN). Both models and tasks are widely studied in the literature, and not chosen deliberately. Results are mostly consistent (except Section 5.4). Along with analyzing our own experimental data, we have also collected related results in previous studies, serving as additional evidence in answering our research questions. Therefore, we think the generality of this work is fair and that the conclusions can be generalized to similar scenarios.

Future work. Our work also points out some future directions of research. For example, we would like to analyze the effect of different MULT strategies. More efforts are also needed in developing an effective yet robust method for multi-task learning.

Acknowledgments

We thank all reviewers for their constructive comments, Sam Bowman for helpful suggestion, and Vicky Li for discussion on the manuscript. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, 61225007, and 61502014.

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing:

Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning*, pages 647–655.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 451–459.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2750–2756.

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

Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2106–2111.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1225–1237.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of 13th European Conference on Computer Vision*, pages 818–833.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *International Joint Conference on Artificial Intelligence*, pages 4069–4076.

Morphological Priors for Probabilistic Neural Word Embeddings

Parminder Bhatia*
Yik Yak, Inc.
3525 Piedmont Rd NE, Building 6, Suite 500
Atlanta, GA
parminder@yikyakapp.com

Robert Guthrie* and **Jacob Eisenstein**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30312 USA
{rguthrie3 + jacobe}@gatech.edu

Abstract

Word embeddings allow natural language processing systems to share statistical information across related words. These embeddings are typically based on distributional statistics, making it difficult for them to generalize to rare or unseen words. We propose to improve word embeddings by incorporating morphological information, capturing shared sub-word features. Unlike previous work that constructs word embeddings directly from morphemes, we combine morphological and distributional information in a unified probabilistic framework, in which the word embedding is a latent variable. The morphological information provides a prior distribution on the latent word embeddings, which in turn condition a likelihood function over an observed corpus. This approach yields improvements on intrinsic word similarity evaluations, and also in the downstream task of part-of-speech tagging.

1 Introduction

Word embeddings have been shown to improve many natural language processing applications, from language models (Mikolov et al., 2010) to information extraction (Collobert and Weston, 2008), and from parsing (Chen and Manning, 2014) to machine translation (Cho et al., 2014). Word embeddings leverage a classical idea in natural language processing: use distributional statistics from large amounts of unlabeled data to learn representations that allow sharing across related words (Brown et al., 1992). While this approach is undeniably effective, the long-tail nature of linguistic data ensures that there will always be words that are not observed in even the largest corpus (Zipf, 1949). There will be many other words which are observed only a handful of times, making the distributional statistics too sparse to accurately estimate the 100- or 1000-dimensional dense vectors that are typically used for word embeddings. These problems are particularly acute in morphologically rich languages like German and Turkish, where each word may have dozens of possible inflections.

Recent work has proposed to address this issue by replacing word-level embeddings with embeddings based on subword units: morphemes (Luong et al., 2013; Botha and Blunsom, 2014) or individual characters (Santos and Zadrozny, 2014; Ling et al., 2015; Kim et al., 2016). Such models leverage the fact that word meaning is often *compositional*, arising from subword components. By learning representations of subword units, it is possible to generalize to rare and unseen words.

But while morphology and orthography are sometimes a signal of semantics, there are also many cases similar spellings do *not* imply similar meanings: *better-batter*, *melon-felon*, *dessert-desert*, etc. If each word's embedding is constrained to be a deterministic function of its characters, as in prior work, then it will be difficult to learn appropriately distinct embeddings for such pairs. Automated morphological analysis may be incorrect: for example, *really* may be segmented into *re+ally*, incorrectly suggesting a similarity to *revise* and *review*. Even correct morphological segmentation may be misleading. Consider

*The first two authors contributed equally. Code is available at https://github.com/rguthrie3/MorphologicalPriorsForWordEmbeddings.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 490–500,
Austin, Texas, November 1-5, 2016.

that *incredible* and *inflammable* share a prefix *in-*, which exerts the opposite effect in these two cases.[1] Overall, a word's observed internal structure gives evidence about its meaning, but it must be possible to override this evidence when the distributional facts point in another direction.

We formalize this idea using the machinery of probabilistic graphical models. We treat word embeddings as *latent variables* (Vilnis and McCallum, 2014), which are conditioned on a prior distribution that is based on word morphology. We then maximize a variational approximation to the expected likelihood of an observed corpus of text, fitting variational parameters over latent binary word embeddings. For common words, the expected word embeddings are largely determined by the expected corpus likelihood, and thus, by the distributional statistics. For rare words, the prior plays a larger role. Since the prior distribution is a function of the morphology, it is possible to impute embeddings for unseen words after training the model.

We model word embeddings as latent binary vectors. This choice is based on linguistic theories of lexical semantics and morphology. Morphemes are viewed as adding morphosyntactic *features* to words: for example, in English, *un-* adds a negation feature (*unbelievable*), *-s* adds a plural feature, and *-ed* adds a past tense feature (Halle and Marantz, 1993). Similarly, the lexicon is often viewed as organized in terms of features: for example, the word *bachelor* carries the features HUMAN, MALE, and UNMARRIED (Katz and Fodor, 1963). Each word's semantic role within a sentence can also be characterized in terms of binary features (Dowty, 1991; Reisinger et al., 2015). Our approach is more amenable to such theoretical models than traditional distributed word embeddings. However, we can also work with the *expected word embeddings*, which are vectors of probabilities, and can therefore be expected to hold the advantages of dense distributed representations (Bengio et al., 2013).

[1]The confusion is resolved by morphologically analyzing the second example as *(in+flame)+able*, but this requires hierarchical morphological parsing, not just segmentation.

2 Model

The modeling framework is illustrated in Figure 1, focusing on the word *sesquipedalianism*. This word is rare, but its morphology indicates several of its properties: the *-ism* suffix suggests that the word is a noun, likely describing some abstract property; the *sesqui-* prefix refers to one and a half, and so on. If the word is unknown, we must lean heavily on these intuitions, but if the word is well attested then we can rely instead on its examples in use.

It is this reasoning that our modeling framework aims to formalize. We treat word embeddings as latent variables in a joint probabilistic model. The prior distribution over a word's embedding is conditioned on its morphological structure. The embedding itself then participates, as a latent variable, in a neural sequence model over a corpus, contributing to the overall corpus likelihood. If the word appears frequently, then the corpus likelihood dominates the prior — which is equivalent to relying on the word's distributional properties. If the word appears rarely, then the prior distribution steps in, and gives a best guess as to the word's meaning.

Before describing these component pieces in detail, we first introduce some notation. The representation of word w is a latent binary vector $\boldsymbol{b}_w \in \{0,1\}^k$, where k is the size of each word embedding. As noted in the introduction, this binary representation is motivated by feature-based theories of lexical semantics (Katz and Fodor, 1963). Each word w is constructed from a set of M_w observed morphemes, $\mathcal{M}_w = (m_{w,1}, m_{w,2}, \ldots, m_{w,M_w})$. Each morpheme is in turn drawn from a finite vocabulary of size v_m, so that $m_{w,i} \in \{1, 2, \ldots, v_m\}$. Morphemes are obtained from an unsupervised morphological segmenter, which is treated as a black box. Finally, we are given a corpus, which is a sequence of words, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, where each word $x_t \in \{1, 2, \ldots, v_w\}$, with v_w equal to the size of the vocabulary, including the token $\langle \text{UNK} \rangle$ for unknown words.

2.1 Prior distribution

The key differentiating property of this model is that rather than estimating word embeddings directly, we treat them as a latent variable, with a prior distribution reflecting the word's morphological proper-

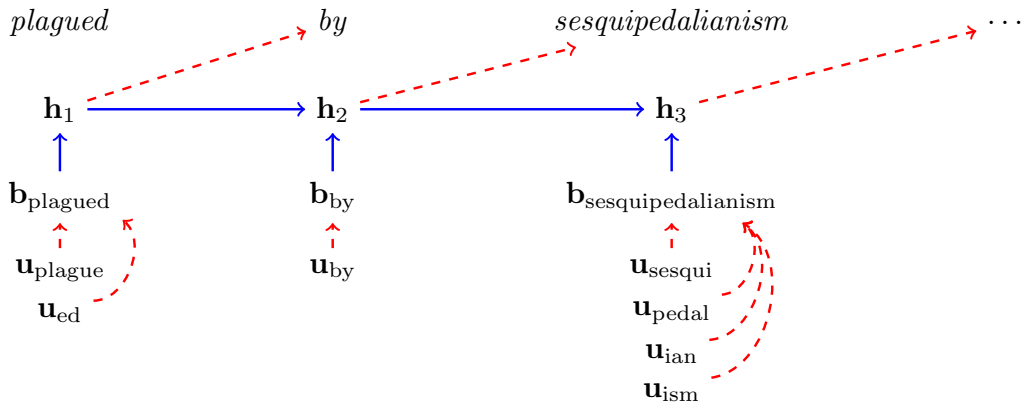


Figure 1: Model architecture, applied to the example sequence . . . *plagued by sesquipedalianism* . . . . Blue solid arrows indicate direct computation, red dashed arrows indicate probabilistic dependency. For simplicity, we present our models as recurrent neural networks rather than long short-term memories (LSTMs).

ties. To characterize this prior distribution, each morpheme m is associated with an embedding of its own, $\boldsymbol{u}_m \in \mathbb{R}^k$, where k is again the embedding size. Then for position i of the word embedding $\boldsymbol{b}_w$, we have the following prior,

$$b_{w,i} \sim \text{Bernoulli}\left(\sigma(\sum_{m \in \mathcal{M}_w} u_{m,i})\right), \quad (1)$$

where $\sigma(\cdot)$ indicates the sigmoid function. The prior log-likelihood for a set of word embeddings is,

$$\log P(\boldsymbol{b}; \mathcal{M}, \boldsymbol{u}) \quad (2)$$

$$= \sum_{w}^{V_w} \log P(\boldsymbol{b}_w; \mathcal{M}_w, \boldsymbol{u}) \quad (3)$$

$$= \sum_{w}^{V_w} \sum_{i}^{k} \log P(b_{w,i}; \mathcal{M}_w, \boldsymbol{u}) \quad (4)$$

$$= \sum_{w}^{V_w} \sum_{i}^{k} b_{w,i} \log \sigma\left(\sum_{m \in \mathcal{M}_w} u_{m,i}\right) \quad (5)$$

$$+ (1 - b_{w,i}) \log\left(1 - \sigma\left(\sum_{m \in \mathcal{M}_w} u_{m,i}\right)\right).$$

2.2 Expected likelihood

The corpus likelihood is computed via a recurrent neural network language model (Mikolov et al., 2010, RNNLM), which is a generative model of sequences of tokens. In the RNNLM, the probability of each word is conditioned on all preceding words through a recurrently updated state vector. This state vector in turn depends on the embeddings of the previous words, through the following update equations:

$$\boldsymbol{h}_t = f(\boldsymbol{b}_{x_t}, \boldsymbol{h}_{t-1}) \quad (6)$$

$$x_{t+1} \sim \text{Multinomial}\left(\text{Softmax}\left[\mathbf{V}\boldsymbol{h}_t\right]\right). \quad (7)$$

The function $f(\cdot)$ is a recurrent update equation; in the RNN, it corresponds to $\sigma(\Theta \boldsymbol{h}_{t-1} + \boldsymbol{b}_{x_t})$, where $\sigma(\cdot)$ is the elementwise sigmoid function. The matrix $\mathbf{V} \in \mathbb{R}^{v \times k}$ contains the "output embeddings" of each word in the vocabulary. We can then define the conditional log-likelihood of a corpus $\boldsymbol{x} = (x_1, x_2, \ldots x_N)$ as,

$$\log P(\boldsymbol{x} \mid \boldsymbol{b}) = \sum_{t}^{N} \log P(x_t \mid \boldsymbol{h}_{t-1}, \boldsymbol{b}). \quad (8)$$

Since $\boldsymbol{h}_{t-1}$ is deterministically computed from $\boldsymbol{x}_{1:t-1}$ (conditioned on $\boldsymbol{b}$), we can equivalently write the log-likelihood as,

$$\log P(\boldsymbol{x} \mid \boldsymbol{b}) = \sum_{t} \log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}). \quad (9)$$

This same notation can be applied to compute the likelihood under a long-short term memory (LSTM) language model (Sundermeyer et al., 2012). The only difference is that the recurrence function $f(\cdot)$ from Equation 6 now becomes more complex, including the input, output, and forget gates, and the recurrent state $\boldsymbol{h}_t$ now includes the memory cell. As the LSTM

update equations are well known, we focus on the more concise RNN notation, but we employ LSTMs in all experiments due to their better ability to capture long-range dependencies.

2.3 Variational approximation

Inference on the marginal likelihood $P(\boldsymbol{x}_{1:N}) = \int P(\boldsymbol{x}_{1:N}, \boldsymbol{b}) d\boldsymbol{b}$ is intractable. We address this issue by making a variational approximation,

$$\log P(\boldsymbol{x}) = \log \sum_{\boldsymbol{b}} P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b}) \tag{10}$$

$$= \log \sum_{\boldsymbol{b}} \frac{Q(\boldsymbol{b})}{Q(\boldsymbol{b})} P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b}) \tag{11}$$

$$= \log E_q \left[P(\boldsymbol{x} \mid \boldsymbol{b}) \frac{P(\boldsymbol{b})}{Q(\boldsymbol{b})} \right] \tag{12}$$

$$\geq E_q[\log P(\boldsymbol{x} \mid \boldsymbol{b}) + \log P(\boldsymbol{b}) - \log Q(\boldsymbol{b})] \tag{13}$$

The variational distribution $Q(\boldsymbol{b})$ is defined using a fully factorized mean field approximation,

$$Q(\boldsymbol{b}; \boldsymbol{\gamma}) = \prod_{w}^{v_w} \prod_{i}^{k} q(b_{w,i}; \gamma_{w,i}). \tag{14}$$

The variational distribution is a product of Bernoullis, with parameters $\gamma_{w,j} \in [0, 1]$. In the evaluations that follow, we use the expected word embeddings $q(\boldsymbol{b}_w)$, which are dense vectors in $[0, 1]^k$. We can then use $Q(\cdot)$ to place a variational lower bound on the expected conditional likelihood,

Even with this variational approximation, the expected log-likelihood is still intractable to compute. In recurrent neural network language models, each word x_t is conditioned on the entire prior history, $\boldsymbol{x}_{1:t-1}$ — indeed, this is one of the key advantages over fixed-length n-gram models. However, this means that the individual expected log probabilities involve not just the word embedding of x_t and its immediate predecessor, but rather, the embeddings of *all* words in the sequence $\boldsymbol{x}_{1:t}$:

$$E_q \left[\log P(\boldsymbol{x} \mid \boldsymbol{b}) \right] \tag{15}$$

$$= \sum_{t}^{N} E_q \left[\log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}) \right] \tag{16}$$

$$= \sum_{t}^{N} \sum_{\{\boldsymbol{b}_w : w \in \boldsymbol{x}_{1:t}\}} Q(\{\boldsymbol{b}_w : w \in \boldsymbol{x}_{1:t}\}) \times \log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}). \tag{17}$$

We therefore make a further approximation by taking a local expectation over the recurrent state,

$$E_q \left[\boldsymbol{h}_t \right] \approx f(E_q \left[\boldsymbol{b}_{x_t} \right], E_q \left[\boldsymbol{h}_{t-1} \right]) \tag{18}$$

$$E_q \left[\log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}) \right] \approx \log \text{Softmax} \left(\mathbf{V} E_q \left[\boldsymbol{h}_t \right] \right). \tag{19}$$

This approximation means that we do not propagate uncertainty about $\boldsymbol{h}_t$ through the recurrent update or through the likelihood function, but rather, we use local point estimates. Alternative methods such as variational autoencoders (Chung et al., 2015) or sequential Monte Carlo (de Freitas et al., 2000) might provide better and more principled approximations, but this direction is left for future work.

Variational bounds in the form of Equation 13 can generally be expressed as a difference between an expected log-likelihood term and a term for the Kullback-Leibler (KL) divergence between the prior distribution $P(\boldsymbol{b})$ and the variational distribution $Q(\boldsymbol{b})$ (Wainwright and Jordan, 2008). Incorporating the approximation in Equation 19, the resulting objective is,

$$\mathcal{L} = \sum_{t}^{N} \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}]) - D_{KL}(Q(\boldsymbol{b}) \parallel P(\boldsymbol{b})). \tag{20}$$

The KL divergence is equal to,

$$D_{KL}(Q(\boldsymbol{b}) \parallel P(\boldsymbol{b})) \tag{21}$$

$$= \sum_{w}^{v_w} \sum_{i}^{k} D_{KL}(q(b_{w,i}) \parallel P(b_{w,i})) \tag{22}$$

$$= \sum_{w}^{v_w} \sum_{i}^{k} \gamma_{w,i} \log \sigma(\sum_{m \in \mathcal{M}_w} u_{m,i}) + (1 - \gamma_{w,i}) \log(1 - \sigma(\sum_{m \in \mathcal{M}_w} u_{m,i})) - \gamma_{w,i} \log \gamma_{w,i} - (1 - \gamma_{w,i}) \log(1 - \gamma_{w,i}). \tag{23}$$

Each term in the variational bound can be easily constructed in a computation graph, enabling automatic differentiation and the application of standard stochastic optimization techniques.

3 Implementation

The objective function is given by the variational lower bound in Equation 20, using the approximation to the conditional likelihood described in Equation 19. This function is optimized in terms of several parameters:

- the morpheme embeddings, $\{\boldsymbol{u}_m\}_{m \in 1 \ldots v_m}$;
- the variational parameters on the word embeddings, $\{\boldsymbol{\gamma}\}_{w \in 1 \ldots v_w}$;
- the output word embeddings $\mathbf{V}$;
- the parameter of the recurrence function, Θ.

Each of these parameters is updated via the `RMSProp` online learning algorithm (Tieleman and Hinton, 2012). The model and baseline (described below) are implemented in `blocks` (van Merriënboer et al., 2015). In the remainder of the paper, we refer to our model as VAREMBED.

3.1 Data and preprocessing

All embeddings are trained on 22 million tokens from the the North American News Text (NANT) corpus (Graff, 1995). We use an initial vocabulary of 50,000 words, with a special ⟨UNK⟩ token for words that are not among the 50,000 most common. We then perform downcasing and convert all numeric tokens to a special ⟨NUM⟩ token. After these steps, the vocabulary size decreases to 48,986. Note that the method can impute word embeddings for out-of-vocabulary words under the prior distribution $P(\boldsymbol{b}; \mathcal{M}, \boldsymbol{u})$; however, it is still necessary to decide on a vocabulary size to determine the number of variational parameters $\boldsymbol{\gamma}$ and output embeddings to estimate.

Unsupervised morphological segmentation is performed using Morfessor (Creutz and Lagus, 2002), with a maximum of sixteen morphemes per word. This results in a total of 14,000 morphemes, which includes stems for monomorphemic words. We do not rely on any labeled information about morphological structure, although the incorporation of gold morphological analysis is a promising topic for future work.

3.2 Learning details

The LSTM parameters are initialized uniformly in the range $[-0.08, 0.08]$. The word embeddings are initialized using pre-trained `word2vec` embeddings. We train the model for 15 epochs, with an initial learning rate of 0.01, a decay of 0.97 per epoch, and minibatches of size 25. We clip the norm of the gradients (normalized by minibatch size) at 1, using the default settings in the `RMSprop` implementation in `blocks`. These choices are motivated by prior work (Zaremba et al., 2014). After each iteration, we compute the objective function on the development set; when the objective does not improve beyond a small threshold, we halve the learning rate.

Training takes roughly one hour per iteration using an NVIDIA 670 GTX, which is a commodity graphics processing unit (GPU) for gaming. This is nearly identical to the training time required for our reimplementation of the algorithm of Botha and Blunsom (2014), described below.

3.3 Baseline

The most comparable approach is that of Botha and Blunsom (2014). In their work, embeddings are estimated for each morpheme, as well as for each in-vocabulary word. The final embedding for a word is then the sum of these embeddings, e.g.,

$$\overline{\text{greenhouse}} = \overline{\textit{greenhouse}} + \overline{\textit{green}} + \overline{\textit{house}}, \tag{24}$$

where the italicized elements represent learned embeddings.

We build a baseline that is closely inspired by this approach, which we call SUMEMBED. The key difference is that while Botha and Blunsom (2014) build on the log-bilinear language model (Mnih and Hinton, 2007), we use the same LSTM-based architecture as in our own model implementation. This enables our evaluation to focus on the critical difference between the two approaches: the use of latent variables rather than summation to model the word embeddings. As with our method, we used pre-trained `word2vec` embeddings to initialize the model.

3.4 Number of parameters

The dominant terms in the overall number of parameters are the (expected) word embeddings themselves. The variational parameters of the input word embeddings, γ, are of size $k \times v_w$. The output word embeddings are of size $\#|\boldsymbol{h}| \times v_w$. The morpheme embeddings are of size $k \times v_m$, with $v_m \ll v_w$. In our main experiments, we set $v_w = 48,896$ (see above), $k = 128$, and $\#|\boldsymbol{h}| = 128$. After including the character embeddings and the parameters of the recurrent models, the total number of parameters is roughly 12.8 million. This is identical to number of parameters in the SUMEMBED baseline.

4 Evaluation

Our evaluation compares the following embeddings:

WORD2VEC We train the popular `word2vec` CBOW (continuous bag of words) model (Mikolov et al., 2013), using the `gensim` implementation.

SUMEMBED We compare against the baseline described in § 3.3, which can be viewed as a reimplementation of the compositional model of Botha and Blunsom (2014).

VAREMBED For our model, we take the *expected* embeddings $E_q[\boldsymbol{b}]$, and then pass them through an inverse sigmoid function to obtain values over the entire real line.

4.1 Word similarity

Our first evaluation is based on two classical word similarity datasets: Wordsim353 (Finkelstein et al., 2001) and the Stanford "rare words" (rw) dataset (Luong et al., 2013). We report Spearmann's ρ, a measure of rank correlation, evaluating on both the entire vocabulary as well as the subset of in-vocabulary words.

As shown in Table 1, VAREMBED consistently outperforms SUMEMBED on both datasets. On the subset of in-vocabulary words, WORD2VEC gives slightly better results on the wordsim words that are in the NANT vocabulary, but is not applicable to the complete dataset. On the rare words dataset, WORD2VEC performs considerably worse than both morphology-based models, matching the findings of Luong et al. (2013) and Botha and Blunsom (2014) regarding the importance of morphology for doing well on this dataset.

4.2 Alignment with lexical semantic features

Recent work questions whether these word similarity metrics are predictive of performance on downstream tasks (Faruqui et al., 2016). The QVEC statistic is another intrinsic evaluation method, which has been shown to be better correlated with downstream tasks (Tsvetkov et al., 2015). This metric measures the alignment between word embeddings and a set of lexical semantic features. Specifically, we use the semcor noun verb supersenses oracle provided at the qvec github repository.[2]

As shown in Table 2, VAREMBED outperforms SUMEMBED on the full lexicon, and gives similar performance to WORD2VEC on the set of in-vocabulary words. We also consider the morpheme embeddings alone. For SUMEMBED, this means that we construct the word embedding from the sum of the embeddings for its morphemes, *without* the additional embedding per word. For VAREMBED, we use the expected embedding under the prior distribution $E[\boldsymbol{b} \mid \boldsymbol{c}]$. The results for these representations are shown in the bottom half of Table 2, revealing that VAREMBED learns much more meaningful embeddings at the morpheme level, while much of the power of SUMEMBED seems to come from the word embeddings.

[2] `https://github.com/ytsvetko/qvec`

	WORD2VEC	SUMEMBED	VAREMBED
Wordsim353			
all words (353)	n/a	42.9	**48.8**
in-vocab (348)	**51.4**	45.9	51.3
rare words (rw)			
all words (2034)	n/a	23.0	**24.0**
in-vocab (715)	33.6	37.3	**44.1**

Table 1: Word similarity evaluation results, as measured by Spearmann's $\rho \times 100$. WORD2VEC cannot be evaluated on all words, because embeddings are not available for out-of-vocabulary words. The total number of words in each dataset is indicated in parentheses.

	all words (4199)	in vocab (3997)
WORD2VEC	n/a	**34.8**
SUMEMBED	32.8	33.5
VAREMBED	**33.6**	34.7
morphemes only		
SUMEMBED	24.7	25.1
VAREMBED	**30.2**	**31.0**

Table 2: Alignment with lexical semantic features, as measured by QVEC. Higher scores are better, with a maximum possible score of 100.

4.3 Part-of-speech tagging

Our final evaluation is on the downstream task of part-of-speech tagging, using the Penn Treebank. We build a simple classification-based tagger, using a feedforward neural network. (This is not intended as an alternative to state-of-the-art tagging algorithms, but as a comparison of the syntactic utility of the information encoded in the word embeddings.) The inputs to the network are the concatenated embeddings of the five word neighborhood $(x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2})$; as in all evaluations, 128-dimensional embeddings are used, so the total size of the input is 640. This input is fed into a network with two hidden layers of size 625, and a softmax output layer over all tags. We train using RMSProp (Tieleman and Hinton, 2012).

Results are shown in Table 3. Both morphologically-informed embeddings are significantly better to WORD2VEC ($p < .01$, two-tailed binomial test), but the difference between SUMEMBED and VAREMBED is not significant

	dev	test
WORD2VEC	92.42	92.40
SUMEMBED	**93.26**	**93.26**
VAREMBED	93.05	93.09

Table 3: Part-of-speech tagging accuracies

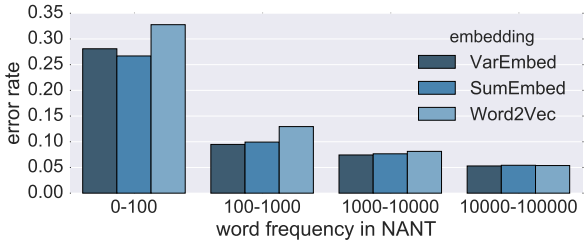


Figure 2: Error rates by word frequency.

at $p < .05$. Figure 2 breaks down the errors by word frequency. As shown in the figure, the tagger based on WORD2VEC performs poorly for rare words, which is expected because these embeddings are estimated from sparse distributional statistics. SUMEMBED is slightly better on the rarest words (the $0 - 100$ group accounts for roughly 10% of all tokens). In this case, it appears that this simple additive model is better, since the distributional statistics are too sparse to offer much improvement. The probabilistic VAREMBED embeddings are best for all other frequency groups, showing that it effectively combines morphology and distributional statistics.

5 Related work

Adding side information to word embeddings An alternative approach to incorporating additional

information into word embeddings is to constrain the embeddings of semantically-related words to be similar. Such work typically draws on existing lexical semantic resources such as WordNet. For example, Yu and Dredze (2014) define a joint training objective, in which the word embedding must predict not only neighboring word tokens in a corpus, but also related word *types* in a semantic resource; a similar approach is taken by Bian et al. (2014). Alternatively, Faruqui et al. (2015) propose to "retrofit" pre-trained word embeddings over a semantic network. Both retrofitting and our own approach treat the true word embeddings as latent variables, from which the pre-trained word embeddings are stochastically emitted. However, a key difference from our approach is that the underlying representation in these prior works is relational, and not generative. These methods can capture similarity between words in a relational lexicon such as WordNet, but they do not offer a generative account of how (approximate) meaning is constructed from orthography or morphology.

Word embeddings and morphology The SUMEMBED baseline is based on the work of Botha and Blunsom (2014), in which words are segmented into morphemes using MORFESSOR (Creutz and Lagus, 2002), and then word representations are computed through addition of morpheme representations. A key modeling difference from this prior work is that rather than computing word embeddings directly and deterministically from subcomponent embeddings (morphemes or characters, as in (Ling et al., 2015; Kim et al., 2016)), we use these subcomponents to define a prior distribution, which can be overridden by distributional statistics for common words. Other work exploits morphology by training word embeddings to optimize a joint objective over distributional statistics and rich, morphologically-augmented part of speech tags (Cotterell and Schütze, 2015). This can yield better word embeddings, but does not provide a way to compute embeddings for unseen words, as our approach does.

Recent work by Cotterell et al. (2016) extends the idea of retrofitting, which was based on semantic similarity, to a morphological framework. In this model, embeddings are learned for morphemes as well as for words, and each word embedding is conditioned on the sum of the morpheme embeddings, using a multivariate Gaussian. The covariance of this Gaussian prior is set to the inverse of the number of examples in the training corpus, which has the effect of letting the morphology play a larger role for rare or unseen words. Like all retrofitting approaches, this method is applied in a pipeline fashion after training word embeddings on a large corpus; in contrast, our approach is a joint model over the morphology and corpus. Another practical difference is that Cotterell et al. (2016) use gold morphological features, while we use an automated morphological segmentation.

Latent word embeddings Word embeddings are typically treated as a parameter, and are optimized through point estimation (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2010). Current models use word embeddings with hundreds or even thousands of parameters per word, yet many words are observed only a handful of times. It is therefore natural to consider whether it might be beneficial to model uncertainty over word embeddings. Vilnis and McCallum (2014) propose to model Gaussian densities over dense vector word embeddings. They estimate the parameters of the Gaussian directly, and, unlike our work, do not consider using orthographic information as a prior distribution. This is easy to do in the latent binary framework proposed here, which is also a better fit for some theoretical models of lexical semantics (Katz and Fodor, 1963; Reisinger et al., 2015). This view is shared by Kruszewski et al. (2015), who induce binary word representations using labeled data of lexical semantic entailment relations, and by Henderson and Popa (2016), who take a mean field approximation over binary representations of lexical semantic features to induce hyponymy relations.

More broadly, our work is inspired by recent efforts to combine directed graphical models with discriminatively trained "deep learning" architectures. The variational autoencoder (Kingma and Welling, 2014), neural variational inference (Mnih and Gregor, 2014; Miao et al., 2016), and black box variational inference (Ranganath et al., 2014) all propose to use a neural network to compute the variational approximation. These ideas are employed by Chung et al. (2015) in the variational recurrent neural network, which places a latent continuous variable at each time step. In contrast, we have a dictionary of latent vari-

ables — the word embeddings — which introduce uncertainty over the hidden state $\boldsymbol{h}_t$ in a standard recurrent neural network or LSTM. We train this model by employing a mean field approximation, but these more recent techniques for neural variational inference may also be applicable. We plan to explore this possibility in future work.

6 Conclusion and future work

We present a model that unifies compositional and distributional perspectives on lexical semantics, through the machinery of Bayesian latent variable models. In this framework, our prior expectations of word meaning are based on internal structure, but these expectations can be overridden by distributional statistics. The model is based on the very successful long-short term memory (LSTM) for sequence modeling, and while it employs a Bayesian justification, its inference and estimation are little more complicated than a standard LSTM. This demonstrates the advantages of reasoning about uncertainty even when working in a "neural" paradigm.

This work represents a first step, and we see many possibilities for improving performance by extending it. Clearly we would expect this model to be more effective in languages with richer morphological structure than English, and we plan to explore this possibility in future work. From a modeling perspective, our prior distribution merely sums the morpheme embeddings, but a more accurate model might account for sequential or combinatorial structure, through a recurrent (Ling et al., 2015), recursive (Luong et al., 2013), or convolutional architecture (Kim et al., 2016). There appears to be no technical obstacle to imposing such structure in the prior distribution. Furthermore, while we build the prior distribution from morphemes, it is natural to ask whether characters might be a better underlying representation: character-based models may generalize well to non-word tokens such as names and abbreviations, they do not require morphological segmentation, and they require a much smaller number of underlying embeddings. On the other hand, morphemes encode rich regularities across words, which may make a morphologically-informed prior easier to learn than a prior which works directly at the character level. It is possible that this tradeoff could be transcended by combining characters and morphemes in a single model.

Another advantage of latent variable models is that they admit partial supervision. If we follow Tsvetkov et al. (2015) in the argument that word embeddings should correspond to lexical semantic features, then an inventory of such features could be used as a source of partial supervision, thus locking dimensions of the word embeddings to specific semantic properties. This would complement the graph-based "retrofitting" supervision proposed by Faruqui et al. (2015), by instead placing supervision at the level of individual words.

Acknowledgments

Thanks to Erica Briscoe, Martin Hyatt, Yangfeng Ji, Bryan Leslie Lee, and Yi Yang for helpful discussion of this work. Thanks also the EMNLP reviewers for constructive feedback. This research is supported by the Defense Threat Reduction Agency under award HDTRA1-15-1-0019.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 740–750.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk,

and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Neural Information Processing Systems (NIPS)*, Montréal.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 160–167.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO, May.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin, August.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics.

João FG de Freitas, Mahesan Niranjan, Andrew H. Gee, and Arnaud Doucet. 2000. Sequential monte carlo methods to train neural network models. *Neural computation*, 12(4):955–993.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, pages 547–619.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO, May.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arxiv*, 1605.02276.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM.

David Graff. 1995. North american news text corpus.

Morris Halle and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. In Kenneth L. Hale and Samuel J. Keyser, editors, *The view from building 20*. MIT Press, Cambridge, MA.

James Henderson and Diana Nicoleta Popa. 2016. A vector space for distributional semantics for entailment. In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin, August.

Jerrold J Katz and Jerry A Fodor. 1963. The structure of a semantic theory. *Language*, pages 170–210.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

German Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, September.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1791–1799.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Rajesh Ranganath, Sean Gerrish, and David Blei. 2014. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1818–1826.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proceedings of INTERSPEECH*.

Tijman Tieleman and Geoffrey Hinton. 2012. Lecture 6.5: Rmsprop. Technical report, Coursera Neural Networks for Machine Learning.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, September.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR*, abs/1412.6623.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 545–550, Baltimore, MD.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort*. Addison-Wesley.

Automatic Cross-Lingual Similarization of Dependency Grammars for Tree-based Machine Translation

Wenbin Jiang [1] and **Wen Zhang** [1] and **Jinan Xu** [2] and **Rangjia Cai** [3]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, China
[3]Research Center of Tibetan Information, Qinghai Normal University, China
jiangwenbin@ict.ac.cn

Abstract

Structural isomorphism between languages benefits the performance of cross-lingual applications. We propose an automatic algorithm for cross-lingual similarization of dependency grammars, which automatically learns grammars with high cross-lingual similarity. The algorithm similarizes the annotation styles of the dependency grammars for two languages in the level of classification decisions, and gradually improves the cross-lingual similarity without losing linguistic knowledge resorting to iterative cross-lingual cooperative learning. The dependency grammars given by cross-lingual similarization have much higher cross-lingual similarity while maintaining non-triviality. As applications, the cross-lingually similarized grammars significantly improve the performance of dependency tree-based machine translation.

1 Introduction

Due to the inherent syntactic regularity of each language and the discrepancy between annotation guidelines of linguists, there is not necessarily structural isomorphism between grammars of different languages. For many cross-lingual scenarios such as information retrieval and machine translation, relationships between linguistic units are expected to be (at least roughly) consistent across languages (Hwa et al., 2002; Smith and Eisner, 2009). For cross-lingual applications, syntactic structures with high cross-lingual similarity facilitates knowledge extraction, feature representation and classification decision. The structural isomorphism between languages, therefore, is an important aspect for the performance of cross-lingual applications such as machine translation.

To achieve effective cross-lingual similarization for two grammars in different languages, an adequate algorithm should both improve the cross-lingual similarity between two grammars and maintain the non-triviality of each grammar, where non-triviality indicates that the resulted grammars should not give flat or single-branched outputs. Different from constituency structures, dependency structures are lexicalized without specialized hierarchical structures. Such concise structures depict the syntactic or semantic relationships between words, and thus have advantage on many cross-lingual scenarios. It is worth to perform cross-lingual similarization for dependency grammars, but the special property of dependency grammars makes it hard to directly adopt the conventional structure transformation methods resorting to hand-crafted rules or templates.

Both graph-based models (McDonald et al., 2005) and transition-based models (Nivre et al., 2006) factorize dependency parsing into fundamental classification decisions, that is, the relationships between words or the actions applied to current states. We assume that cross-lingual similarization can also be factorized into fundamental classification decisions, and propose an automatic cross-lingual similarization algorithm for dependency grammars according to this assumption. The algorithm conducts cross-lingual similarization on the level of classification decisions

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 501–510,
Austin, Texas, November 1-5, 2016.

with simple blending operations rather than on the level of syntactic structures with complicated hand-crafted rules or templates, and adopts iterative cross-lingual collaborative learning to gradually improve the cross-lingual similarity while maintaining the non-triviality of grammars.

We design an evaluation metric for the cross-lingual similarity of dependency grammars, which calculates the consistency degree of dependency relationships across languages. We also propose an effective method to measure the *real* performance of the cross-lingually similarized grammars based on the transfer learning methodology (Pan and Yang, 2010). We validate the method on the dependency grammar induction of Chinese and English, where significant increment of cross-lingual similarity is achieved without losing non-triviality of the grammars. As applications, the cross-lingually similarized grammars gain significant performance improvement for the dependency tree-based machine translation by simply replacing the parser of the translator.

2 Graph-based Dependency Parsing

Dependency parsing aims to link each word to its arguments so as to form a directed graph spanning the whole sentence. Normally the directed graph is restricted to a dependency tree where each word depends on exactly one parent, and all words find their parents. Given a sentence as a sequence n words:

$$x = x_1\ x_2\ ..\ x_n$$

dependency parsing finds a dependency tree y, where $(i, j) \in y$ is an edge from the head word x_i to the modifier word x_j. The root $r \in x$ in the tree y has no head word, and each of the other words, $j(j \in x \textbf{ and } j \neq r)$, depends on a head word $i(i \in x \textbf{ and } i \neq j)$.

Following the edge-based factorization method (Eisner, 1996), the score of a dependency tree can be factorized into the dependency edges in the tree. The graph-based method (McDonald et al., 2005) factorizes the score of the tree as the sum of the scores of all its edges, and the score of an edge is defined as the inner product of the feature vector and the weight vector. Given a sentence x, the parsing procedure searches for the candidate dependency tree with the maximum score:

$$\begin{aligned} y(x) &= \underset{y \in \mathbf{GEN}(x)}{\operatorname{argmax}} \mathbf{S}(y) \\ &= \underset{y \in \mathbf{GEN}(x)}{\operatorname{argmax}} \sum_{(i,j) \in y} \mathbf{S}(i, j) \end{aligned} \quad (1)$$

Here, the function $\mathbf{GEN}$ indicates the enumeration of candidate trees. The MIRA algorithm (Crammer et al., 2003) is used to train the parameter vector. A bottom-up dynamic programming algorithm is designed for projective parsing which gives projective parsing trees, and the Chu-Liu-Edmonds algorithm for non-projective parsing which gives non-projective parsing trees.

3 Cross-Lingual Similarization

Since structural analysis can be factorized into fundamental classification decisions, we assume that the adjustment of the analysis results can be factorized into the adjustment of the fundamental decisions. The classification decision for graph-based dependency parsing is to classify the dependency relationship between each pair of words, and we hope it works well to conduct cross-lingual similarization on the level of dependency relationship classification. In this work, we investigate the automatic cross-lingual similarization for dependency grammars on the level of fundamental classification decisions, to avoid the difficulty of using hand-crafted transformation rules or templates.

In this section, we first introduce the evaluation metric for cross-lingual similarity, then describe the automatic cross-lingual similarization algorithm, and finally give a method to measure the real performance of the cross-lingually similarized grammars.

3.1 Evaluation of Cross-Lingual Similarity

The cross-lingual similarity between two dependency structures can be automatically evaluated. Dependency parsing is conducted on sentences, so we take bilingual sentence pairs as the objects for evaluation. The calculation of cross-lingual similarity needs the lexical alignment information between two languages, which can be obtained by manual annotation or unsupervised algorithms.

Given a bilingual sentence pair x_α and x_β, their dependency structures y_α and y_β, and the word

alignment probabilities $\mathcal{A}$, the cross-lingual similarity can be calculated as below:

$$d(y_\alpha, y_\beta) = \frac{\sum_{(i,j)\in y_\alpha} \sum_{(i',j')\in y_\beta} \mathcal{A}_{i,i'}\mathcal{A}_{j,j'}}{\sum_{(i,j)\in y_\alpha} \sum_{i',j'\in x_\beta} \mathcal{A}_{i,i'}\mathcal{A}_{j,j'}} \quad (2)$$

The bracketed word pair indicates a dependency edge. The evaluation metric is a real number between 0 and 1, indicating the degree of cross-lingual consistency between two dependency structures. For the cross-lingual similarity between bilingual paragraphs, we simply define it as the average over the similarity between each sentence pairs.

3.2 Factorized Cooperative Similarization

The fundamental decisions for graph-based dependency parsing are to evaluate the candidate dependency edges. The cross-lingual similarization for fundamental decisions can be defined as some kinds of blending calculation on two evaluation scores, of which the one is directly given by the grammar of the current language (current grammar), and the other is bilingually projected from the grammar of the reference language (reference grammar).

For the words i and j in the sentence x_α in the current language, their evaluated score given by the current grammar is $\mathbf{S}_\alpha(i,j)$, which can be calculated according to formula 1. The score bilingually projected from the reference grammar, $\mathbf{S}^\beta(i,j)$, can be obtained according to the translation sentence x_β in the reference language and the word alignment between two sentences:

$$\mathbf{S}^\beta(i,j) = \sum_{i',j'\in x_\beta} \mathcal{A}_{i,i'}\mathcal{A}_{j,j'}\mathbf{S}_\beta(i',j') \quad (3)$$

where i' and j' are the corresponding words of i and j in the reference sentence x_β, $\mathcal{A}_{i,j}$ indicates the probability that i aligns to j, and $\mathbf{S}_\beta(i',j')$ is the evaluated score of the candidate edge (i',j') given by the reference grammar.

Given the two evaluated scores, we simply adopt the linear weighted summation:

$$\mathbf{S}^\beta_\alpha(i,j) = (1-\lambda)\mathbf{S}_\alpha(i,j) + \lambda\mathbf{S}^\beta(i,j) \quad (4)$$

where λ is the relative weight to control the degree of cross-lingual similarization, indicating to which degree we consider the decisions of the reference grammar when adjusting the decisions of the current grammar. We have to choose a value for λ to achieve an appropriate speed for effective cross-lingual similarization, in order to obtain similarized grammars with high cross-lingual similarity while maintaining the non-triviality of the grammars.

In the re-evaluated full-connected graph, the decoding algorithm searches for the cross-lingually similarized dependency structures, which are used to re-train the dependency grammars. Based on the cross-lingual similarization strategy, iterative cooperative learning simultaneously similarizes the sentences in the current and reference languages, and gradually improves the cross-lingual similarity between two grammars while maintaining the non-triviality of each monolingual grammar. The whole training algorithm is shown in Algorithm 1. To reduce the computation complexity, we choose the same λ for the cross-lingual similarization for both the current and the reference grammars. Another hyper-parameter for the iterative cooperative learning algorithm is the maximum training iteration, which can be determined according to the performance on the development sets.

3.3 Evaluation of Similarized Grammars

The *real* performance of a cross-lingually similarized grammar is hard to directly measured. The accuracy on the standard testing sets no longer reflects the actrual accuracy, since cross-lingual similarization leads to grammars with annotation styles different from those of the original treebanks. We adopt the transfer learning strategy to automatically adapt the divergence between different annotation styles, and design a *transfer classifier* to transform the dependency regularities from one annotation style to another.

The training procedure of the transfer classifier is analogous to the training of a normal classifier except for the features. The transfer classifier adopts guiding features where a guiding signal is attached to the tail of each normal feature. The guiding signal is the dependency path between the pair of words in the source annotations, as shown in Figure 2. Thus, the transfer classifier learns the statistical regularity of the transformation from the annotations of the cross-lingually similarized grammar to the annotations of the original treebank. Figure 1 shows

Algorithm 1 Cooperative cross-lingual similarization.

function BISIMILARIZE($\mathbf{G}_\alpha$, $\mathbf{G}_\beta$, λ, $\mathbf{C}$) ▷ $\mathbf{C}$ includes a set of sentence pairs (x_α, x_β)
  repeat
    $\mathbf{T}_\alpha, \mathbf{T}_\beta \leftarrow$ BIANNOTATE($\mathbf{G}_\alpha, \mathbf{G}_\beta, \lambda, \mathbf{C}$) ▷ it invokes BIPARSE to parse each (x_α, x_β)
    $\mathbf{G}_\alpha \leftarrow$ GRAMMARTRAIN($\mathbf{T}_\alpha$)
    $\mathbf{G}_\beta \leftarrow$ GRAMMARTRAIN($\mathbf{T}_\beta$)
  until SIMILARITY($\mathbf{G}_\alpha, \mathbf{G}_\beta$) converges ▷ according to formula 2, averaged across $\mathbf{C}$
  return $\mathbf{G}_\alpha$, $\mathbf{G}_\beta$
function BIPARSE($\mathbf{G}_\alpha$, $\mathbf{G}_\beta$, λ, x_α, x_β, $\mathcal{A}$)
  $y_\alpha \leftarrow \text{argmax}_y(1-\lambda)\mathbf{S}_\alpha(y) + \lambda\mathbf{S}^\beta(y)$ ▷ according to formula 4
  $y_\beta \leftarrow \text{argmax}_y(1-\lambda)\mathbf{S}_\beta(y) + \lambda\mathbf{S}^\alpha(y)$
  return y_α, y_β

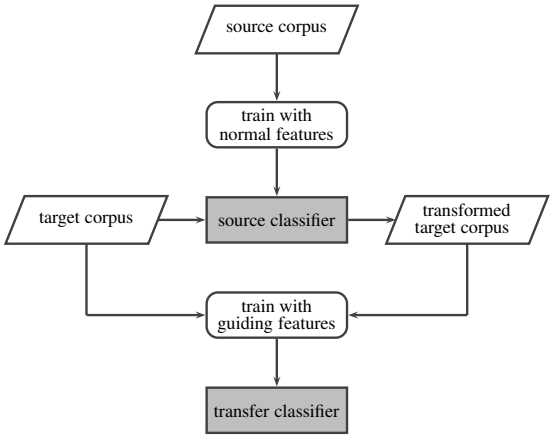
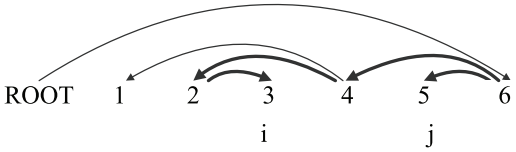


Figure 1: The training procedure of the transfer classifier.

the training pipeline for the transfer classifier, where source corpus and target corpus indicate the cross-lingually similarized treebank and the manually annotated treebank, respectively.

In decoding, a sentence is first parsed by the cross-lingually similarized grammar, and then parsed by the transfer classifier with the result of the similarized grammar as guiding signals to obtain the final parsing results. The improvement achieved by the transfer classifier against a normal classifier trained only on the original treebank reflects the promotion effect of the cross-lingually similarized grammar. The accuracy of the transfer classifier, therefore, *roughly* indicates the real performance of the cross-lingually similarized grammar.



path(i =3, j = 5) = up-up-up-down

Figure 2: The guiding signal for dependency parsing, where $path(i, j)$ denotes the dependency path between i and j. In this example, j is a son of the great-grandfather of i.

4 Tree-based Machine Translation

Syntax-based machine translation investigates the hierarchical structures of natural languages, including formal structures (Chiang, 2005), constituency structures (Galley et al., 2006; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008) and dependency structures (Lin, 2004; Quirk et al., 2005; Ding and Palmer, 2005; Xiong et al., 2007; Shen et al., 2008; Xie et al., 2011), so the performance is restricted to the quality and suitability of the parsers. Since the trees for training follow an annotation style not necessarily isomorphic to that of the target language, it would be not appropriate for syntax-based translation to directly use the parsers trained on the original treebanks. The cross-lingually similarized grammars, although performing poorly on a standard testing set, may be well suitable for syntax-based machine translation. In this work, we use the cross-lingually similarized dependency grammars in dependency tree-to-string machine translation (Xie et al., 2011), a state-of-the-art translation model resorting to dependency trees on the source side.

Treebank	Train	Develop	Test
CTB	1-270 400-931 1001-1151	301-325	271-300
WSJ	02-21	22	23

Table 1: Data partitioning for CTB and WSJ, in unit of section.

5 Experiments and Analysis

We first introduce the dependency parsing itself, then describe the cross-lingual similarization, and finally show the application of cross-lingually similarized grammars in tree-based machine translation. For convenience of description, a grammar trained by the conventional dependency model is named as *original grammar*, a grammar after cross-lingual similarization is named as *similarized grammar*, and the transferred version for a similarized grammar is named as *adapted grammar*.

5.1 Dependency Parsing

We take Chinese dependency parsing as a case study, and experiment on Penn Chinese Treebank (CTB) (Xue et al., 2005). The dependency structures are extracted from the original constituency trees according to the head-selection rules (Yamada and Matsumoto, 2003). The partitioning of the dataset is listed in the Table 1, where we also give the partitioning of Wall Street Journal (WSJ) (Marcus et al., 1993) used to train the English grammar. The evaluation metric for dependency parsing is unlabeled accuracy, indicating the proportion of the words correctly finding their parents. The MIRA algorithm is used to train the classifiers.

Figure 3 gives the performance curves on the development set with two searching modes, *projective* searching and *non-projective* searching. The curves show that the non-projective searching mode fall behind of the projective one, this is because the dependency structures extracted from constituency trees are projective, and the projective search mode implies appropriate constraints on the searching space. Therefore, we use the projective searching mode for the evaluation of the original grammar. Table 2 lists the performance of the original grammar on the CTB testing set.

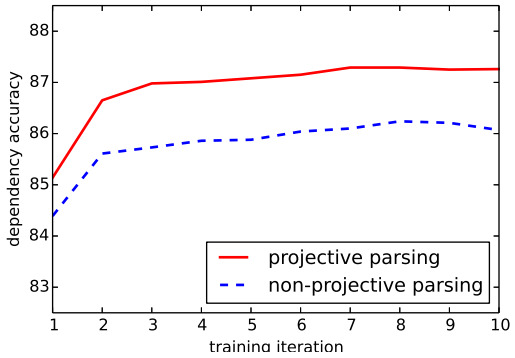


Figure 3: The developing curves of Chinese dependency parsing with both projective and non-projective searching modes.

5.2 Cross-Lingual Similarization

The experiments of cross-lingual similarization are conducted between Chinese and English, with FBIS Chinese-English dataset as bilingual corpus. The Chinese sentences are segmented into words with the character classification model (Ng and Low, 2004), which is trained by MIRA on CTB. The word sequences of both languages are labeled with part-of-speech tags with the maximum entropy hidden markov model (Ratnaparkhi and Adwait, 1996), which is reimplemented with MIRA and trained on CTB and WSJ. The word alignment information is obtained by summing and normalizing the 10 best candidate word alignment results of GIZA++ (Och and Ney, 2003).

The upmost configuration for cross-lingual similarization is the searching mode. On the Chinese side, both projective and non-projective modes can be adopted. For English, there is an additional *fixed* mode besides the previous two. In the fixed mode, the English dependency grammar remains unchanged during the whole learning procedure. The fixed mode on the English side means a degenerated version of cross-lingual similarization, where only the Chinese grammars are revolved during training. The combination of the searching modes for both languages results in a total of 6 kinds of searching configurations. For each configuration, the learning algorithm for cross-lingual similarization has two hyper-parameters, the coefficient λ and maximum iteration for iterative learning, which should be tuned first.

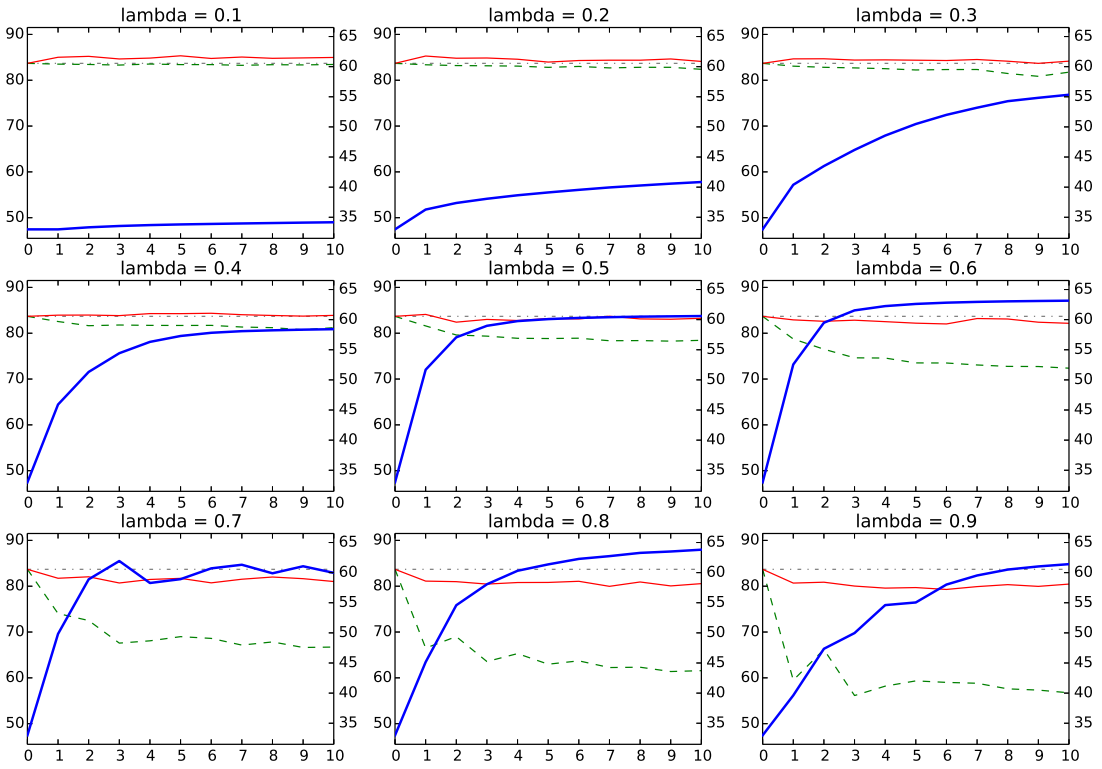


Figure 4: The developing curves of cross-lingual similarization with projective searching on both languages. X-axis: training iteration; Left Y-axis: parsing accuracy; Right Y-axis: cross-lingual similarity. Thin dash-dotted line (gray): accuracy of the baseline grammar; Thin dashed line (green): direct accuracy of cross-lingually similarized grammars; Thin solid line (red): adaptive accuracy of cross-lingually similarized grammars; Thick sold line (blue): the cross-lingual similarity of grammars.

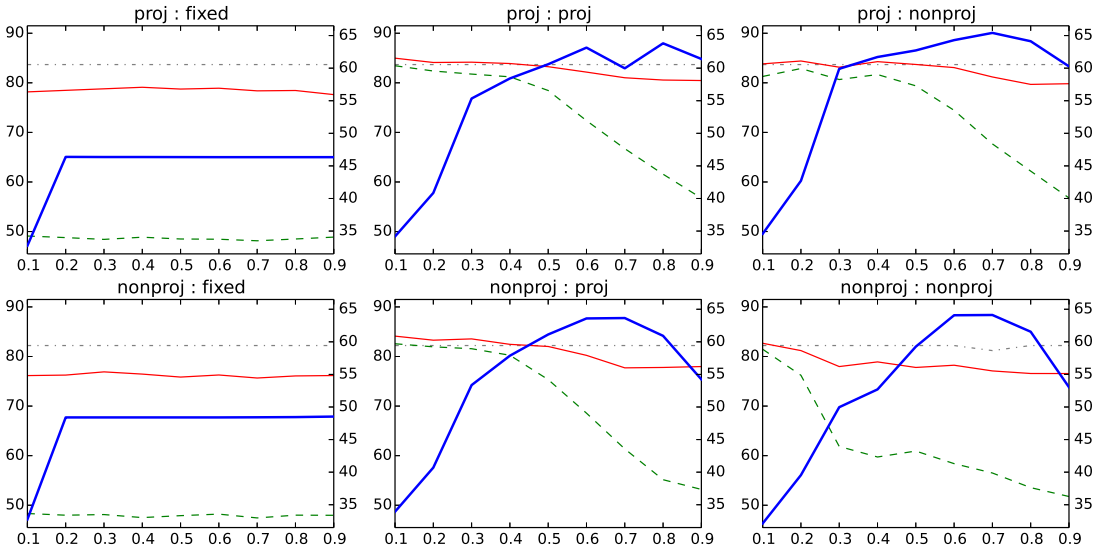


Figure 5: The developing curves of cross-lingual similarization with all searching configurations. X-axis: coefficient λ; Left Y-axis: parsing accuracy; Right Y-axis: cross-lingual similarity. The lines indicate the same as in Figure 4.

5.2.1 Determination of Hyper-Parameters

We select a subset of 40,000 sentence pairs out of the FBIS dataset, and use it as the smaller bilingual corpus to tune the hyper-parameters. For the coefficient λ we try from 0.1 to 0.9 with step 0.1; and for the iterative learning we simply set the maximum iteration as 10. The developing procedure results in a series of grammars. For the configuration with projective searching modes on both sides, a total of 90 pairs of Chinese and English grammars are generated. We use three indicators to validate each similarized grammar generated in the developing procedure, including the performance on the similarized grammar itself (direct accuracy), the performance of the corresponding adapted grammar (adaptive accuracy), and the cross-lingual similarity between the similarized grammar and its English counterpart. Figure 4 shows the developing curves for the configuration with projective searching on both sides. With the fixed maximum iteration 10, we draw the developing curves for the other searching configurations with respect to the weight coefficient, as shown in Figure 5.

We find that the optimal performance is also achieved at 0.6 in most situations. In all configurations, the training procedures increase the cross-lingual similarity of grammars. Along with the increment of cross-lingual similarity, the direct accuracy of the similarized grammars on the development set decreases, but the adaptive accuracy given by the corresponding adapted grammars approach to that of the original grammars. Note that the projective searching mode is adopted for the evaluation of the adapted grammar.

5.2.2 Selection of Searching Modes

With the hyper-parameters given by the developing procedures, cross-lingual similarization is conducted on the whole FBIS dataset. All the searching mode configurations are tried and 6 pairs of grammars are generated. For each of the 6 Chinese dependency grammars, we also give the three indicators as described before. Table 2 shows that, cross-lingual similarization results in grammars with much higher cross-lingual similarity, and the adaptive accuracies given by the adapted grammars approach to those of the original grammars. It indicates that the proposed algorithm improve the cross-lingual similarity without losing syntactic knowledge.

To determine the best searching mode for tree-based machine translation, we use the Chinese-English FBIS dataset as the small-scale bilingual corpus. A 4-gram language model is trained on the Xinhua portion of the Gigaword corpus with the SRILM toolkit (Stolcke and Andreas, 2002). For the analysis given by non-projective similarized grammars, The *projective transformation* should be conducted in order to produce projective dependency structures for rule extraction and translation decoding. In details, the projective transformation first traverses the non-projective dependency structures just as they are projective, then adjusts the order of the nodes according to the traversed word sequences. We take NIST MT Evaluation testing set 2002 (NIST 02) for developing , and use the case-sensitive BLEU (Papineni et al., 2002) to measure the translation accuracy.

The last column of Table 2 shows the performance of the grammars on machine translation. The cross-lingually similarized grammars corresponding to the configurations with projective searching for Chinese always improve the translation performance, while non-projective grammars always hurt the performance. It probably can be attributed to the low performance of non-projective parsing as well as the inappropriateness of the simple projective transformation method. In the final application in machine translation, we adopted the similarized grammar corresponding to the configuration with projective searching on the source side and non-projective searching on the target side.

5.3 Improving Tree-based Translation

Our large-scale bilingual corpus for machine translation consists of 1.5M sentence pairs from LDC data, including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06. The source sentences are parsed by the original grammar and the selected cross-lingually similarized grammar. The alignments are obtained by running GIZA++ on the corpus in both directions and applying grow-diag-and refinement (Koehn et al., 2003). The English language model is trained on the Xinhua portion of the Gigaword corpus with the SRILM toolkit (Stol-

Grammar	Similarity (%)	Dep. P (%)	Ada. P (%)	BLEU-4 (%)
baseline	34.2	84.5	84.5	24.6
proj : fixed	46.3	54.1	82.3	25.8 $_{(+1.2)}$
proj : proj	63.2	72.2	84.6	26.1 $_{(+1.5)}$
proj : nonproj	64.3	74.6	84.7	26.2 $_{(+1.6)}$
nonproj : fixed	48.4	56.1	82.6	20.1 $_{(-4.5)}$
nonproj : proj	63.6	71.4	84.4	22.9 $_{(-1.7)}$
nonproj : nonproj	64.1	73.9	84.9	20.7 $_{(-3.9)}$

Table 2: The performance of cross-lingually similarized Chinese dependency grammars with different configurations.

System	NIST 04	NIST 05
(Liu et al., 2006)	34.55	31.94
(Chiang, 2007)	35.29	33.22
(Xie et al., 2011)	35.82	33.62
Original Grammar	35.44	33.08
Similarized Grammar	**36.78**	**35.12**

Table 3: The performance of the cross-lingually similarized grammar on dependency tree-based translation, compared with related work.

cke and Andreas, 2002). We use NIST 02 as the development set, and NIST 04 and NIST 05 as the testing sets. The quality of translations is evaluated by the case insensitive NIST BLEU-4 metric.

Table 3 shows the performance of the cross-lingually similarized grammar on dependency tree-based translation, compared with previous work (Xie et al., 2011). We also give the performance of constituency tree-based translation (Liu et al., 2006) and formal syntax-based translation (Chiang, 2007). The original grammar performs slightly worse than the previous work in dependency tree-based translation, this can ascribed to the difference between the implementation of the original grammar and the dependency parser used in the previous work. However, the similarized grammar achieves very significant improvement based on the original grammar, and also significant surpass the previous work. Note that there is no other modification on the translation model besides the replacement of the source parser.

From the perspective of performance improvement, tree-to-tree translation would be a better scenario to verify the effectiveness of cross-lingual similarization. This is because tree-to-tree translation suffers from more serious non-isomorphism between the source and the target syntax structures, and our method for cross-lingual similarization can simultaneously similarize both the source and the target grammars. For dependency-based translation, however, there are no available tree-to-tree models for us to verify this assumption. In the future, we want to propose a specific tree-to-tree translation method to better utilize the isomorphism between cross-lingually similarized grammars.

6 Related Work

There are some work devoted to adjusting the syntactic structures according to bilingual constraints to improve constituency tree-based translation (Huang and Knight, 2006; Ambati and Lavie, 2008; Wang et al., 2010; Burkett and Klein, 2012; Liu et al., 2012). These efforts concentrated on constituency structures, adopted hand-crafted transformation templates or rules, and learnt the operation sequences of structure transformation on the bilingual corpora. Such methods are hard to be directly applied to dependency structures due to the great discrepancy between constituency and dependency grammars. There are also work on automatically adjusting the syntactic structures for machine translation resorting to self-training (Morishita et al., 2015), where the parsed trees for self-training are selected according to translation performance. Our work focuses on the automatic cross-lingual similarization of dependency grammars, and learnt grammars with higher cross-lingual similarity while maintaining the non-triviality of the grammars.

There are substantial efforts that have been made in recent years towards harmonizing syntactic representations across languages. This includes the HamleDT project (Zeman et al., 2012; Zeman et al., 2014), as well as the Universal Dependencies initiative (Petrov et al., 2012; McDonald et al., 2013).

Our work aims to automatically harmonize the dependency representations resorting to bilingual correspondence, thus can be grouped into the building strategies for harmonized or universal dependencies. These existing annotated treebanks would also permit interesting control experiments, both for the measurement of similarity and for parsing.

7 Conclusion and Future Work

We propose an automatic cross-lingual similarization algorithm for dependency grammars, design an automatic evaluation metric to measure the cross-lingual similarity between grammars, and use the similarized grammars to improve dependency tree-based machine translation. Experiments show the efficacy of this method. The cross-lingual similarization in this paper is still *soft similarization*, it is worth to investigate the *hard similarization*, where the syntactic structures are totally isomorphic between two languages. Of course, in such syntactic structures, the syntactic nodes should be super-node, that is, a graph containing one or more basic syntactic nodes. Hard similarization could be more suitable for cross-lingual applications, and we leave this aspect for future research.

Acknowledgments

The authors are supported by National Natural Science Foundation of China (Contract 61379086 and 61370130). Jiang is also supported by Open-end Fund of the Platform of Research Database and Information Standard of China (No. qhkj2015-01). We sincerely thank the anonymous reviewers for their insightful comments.

References

Vamshi Ambati and Alon Lavie. 2008. Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proceedings of Student Research Workshop of AMTA*.

David Burkett and Dan Klein. 2012. Transforming trees to improve syntactic convergence. In *Proceedings of EMNLP-CNLL*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Koby Crammer, Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. 2003. Online passive aggressive algorithms. In *Proceedings of NIPS*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the ACL*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345.

Michel Galley, Jonathan Graehl, Kevin Knight, and Daniel Marcu. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the COLING-ACL*.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of NAACL*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the AMTA*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the ACL*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the COLING*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the ACL*.

Shujie Liu, Chi-Ho Li, Mu Li, and Ming Zhou. 2012. Re-training monolingual parser bilingually for syntactic smt. In *Proceedings of EMNLP-CNLL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. In *Computational Linguistics*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundagez, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Nuria Bertomeu Castello, and Jungmee Leez. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of the ACL*.

Makoto Morishita, Koichi Akabe, Yuto Hatakoshi, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. 2015. Parser self-training for syntax-based machine translation. In *Proceedings of IWSLT*.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*.

Joakim Nivre, Johan Hall, Jens Nilsson, Gulsen Eryigit, and Svetoslav Marinov. 2006. Labeled pseudoprojective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE TKDE*.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. *Proceedings of LREC*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the ACL*.

Ratnaparkhi and Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

David Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*.

Stolcke and Andreas. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 311–318.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-alignment for syntax-based machine translation. *Computational Linguistics*.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of EMNLP*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of Workshop on SMT*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Jan Hajič, and Zdeněk Žabokrtský. 2012. Hamledt: To parse or not to parse?

Daniel Zeman, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. 2014. Hamledt: Harmonized multi-language dependency treebank. *Language Resources & Evaluation*.

IRT-based Aggregation Model of Crowdsourced Pairwise Comparisons for Evaluating Machine Translations

Naoki Otani[1] **Toshiaki Nakazawa**[2] **Daisuke Kawahara**[1] **Sadao Kurohashi**[1]
[1]Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto, Japan
[2]Japan Science and Technology Agency, Kawaguchi-shi, Saitama, Japan
otani.naoki.65v@st.kyoto-u.ac.jp nakazawa@pa.jst.jp {dk,kuro}@i.kyoto-u.ac.jp

Abstract

Recent work on machine translation has used crowdsourcing to reduce costs of manual evaluations. However, crowdsourced judgments are often biased and inaccurate. In this paper, we present a statistical model that aggregates many manual pairwise comparisons to robustly measure a machine translation system's performance. Our method applies graded response model from item response theory (IRT), which was originally developed for academic tests. We conducted experiments on a public dataset from the Workshop on Statistical Machine Translation 2013, and found that our approach resulted in highly interpretable estimates and was less affected by noisy judges than previously proposed methods.

1 Introduction

Manual evaluation is a primary means of interpreting the performance of machine translation (MT) systems and evaluating the accuracy of automatic evaluation metrics. It is also essential for natural language processing tasks such as summarization and dialogue systems, where (1) the number of correct outputs is unlimited, and (2) naïve text matching cannot judge the correctness, that is, an evaluator must consider syntactic and semantic information.

Recent work has used crowdsourcing to reduce costs of manual evaluations. However, the judgments of crowd workers are often noisy and unreliable because they are not experts.

To maintain quality, evaluation tasks implemented using crowdsourcing should be simple. Thus, many previous studies focused on pairwise comparisons instead of absolute evaluations. The same task is given to multiple workers, and their responses are aggregated to obtain a reliable answer.

We must, therefore, develop methods that robustly estimate the MT performance based on many pairwise comparisons.

Some aggregation methods have been proposed for MT competitions hosted by the Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2013; Hopkins and May, 2013; Sakaguchi et al., 2014), where a ranking of the submitted systems is produced by aggregating many manual judgments of pairwise comparisons of system outputs.

However, existing methods do not consider the following important issues.

Interpretability of the estimates: For the purpose of evaluation, their results must be interpretable so that we could use the results to improve MT systems and the next MT evaluation campaigns. Existing methods, however, only yield system-level scores.

Judge sensitivity: Some judges can examine the quality of translations with consistent standards, but others cannot (Graham et al., 2015). Sensitivities to the translation quality and judges' own standards are important factors.

Evaluation of a newly submitted system: Previous approaches considered all pairwise combinations of systems and must compare a newly submitted system with all the submitted systems. This made it difficult to allow participants to submit their systems after starting the evaluation step.

To address these issues, we use a model from

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 511–520,
Austin, Texas, November 1-5, 2016.

item response theory (IRT). This theory was originally developed for psychometrics, and has applications to academic tests. IRT models are highly interpretable and are supported by theoretical and empirical studies. For example, we can estimate the informativeness of a question in a test based on the responses of examinees.

We focused on aggregating many pairwise comparisons with a baseline translation so that we could use the analogy of standard academic tests. Figure 1 shows our problem setting. Each system of interest yields translations, and the translations are compared with a baseline translation by multiple human judges. Each judge produces a preference judgment.

The pairwise comparisons correspond to questions in academic tests, a judge's sensitivity to the translation quality is mapped to discrimination of questions, and the relative difficulty of winning the pairwise comparison is mapped to the difficulty of questions. MT systems correspond to students that take academic tests, and IRT models can be naturally applied to estimate the latent performance (ability) of MT systems (students).

Additionally, our approach, fixing baseline translations, can easily evaluate a newly submitted system. We only need to compare the new system with the baseline instead of testing all pairwise combinations of the submitted systems.

Our contributions are summarized as follows.[1]

1. We propose an IRT-based aggregation model of pairwise comparisons with highly interpretable parameters.
2. We simulated noisy judges on the WMT13 dataset and demonstrated that our model is less affected by the noisy judges than previously proposed methods.

2 Related Work

The WMT shared tasks have collected many manual judgments of segment-level pairwise comparisons and used them to produce system-level rankings for MT tasks. Various methods has been proposed to aggregate the judgments to produce reliable rankings.

[1]We also show that our method accurately replicated the WMT13 official system scores using a few comparisons. However, this is not the main focus of this paper.

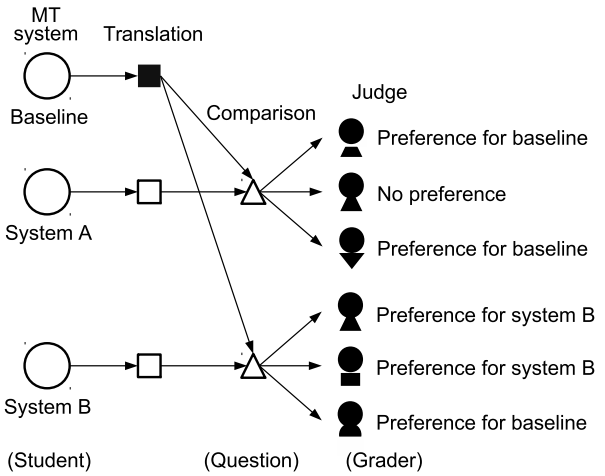


Figure 1: **Illustration of manual pairwise comparison.** Each system yields translations. Judges compare them with a baseline translation and report their preferences. Our goal is to aggregate the judgments to determine the performance of each system.

Frequency based approaches were used to produce the WMT13 official rankings (Bojar et al., 2013), considering statistical significance of the results (Koehn, 2012).

Hopkins and May (2013) noted that we should consider the relative matchup difficulty, and proposed a statistical aggregation model. Their model assumes that the quality of each system can be represented by a Gaussian distribution.

Sakaguchi et al. (2014) applied TrueSkill (Herbrich et al., 2006) to reduce the number of comparisons to reach the final estimate based on an active learning strategy. The same model was recently used for grammatical error correction (Grundkiewicz et al., 2015; Napoles et al., 2015).

These methods acquire the final system-level scores, whereas our model also estimates segment specific and judge specific parameters.

The Bradley–Terry (BT) model was the result of a seminal study on aggregating pairwise comparisons (Bradley and Terry, 1952; Chen et al., 2013; Dras, 2015). Recently, Chen et al. (2013) explicitly incorporated the quality of judges into the BT model, and applied it to quality control in crowdsourcing.

The previously mentioned methods focused on pairwise comparisons of all combination of the MT systems, and thus, the number of comparisons increases rapidly as the number of systems increases.

Our approach, however, only uses comparisons with a fixed baseline. This approach enables to apply IRT models for academic tests and makes it easy to evaluate a newly submitted system.

The work most relevant to our model is the IRT-based crowdsourcing model proposed by Baba and Kashima (2013). Their goal was to estimate the true quality of artifacts such as design works based on ratings assigned by reviewers. They also applied a graded response model to incorporate the authors' latent abilities and the reviewers' biases.

Yet their setting differs from ours in that they focused on the quality of the artifacts, whereas we are interested in the authors. Additionally, their model maps task difficulty and review bias to a difficulty parameter in IRT. However, we naturally extended the model so that standard analysis approaches can be applied to maintain interpretability.

Some studies have focused on absolute evaluations (Goto et al., 2014; Graham et al., 2015). Graham et al. (2015) gathered continuous scale evaluations in terms of adequacy and fluency for many segments, and filtered out noisy judgments based on their consistency. The proposed pipeline results in very accurate evaluations, but 40-50% of all the judgments were filtered out due to inconsistencies. This explains the difficulties of developing absolute evaluation methods in crowdsourcing.

3 Problem Setting

We first describe the problem setting, as shown in Figure 1.

Assume that there are a group of systems $\mathcal{I}$ indexed by i, a set of segments $\mathcal{J}$ indexed by j, and a set of judges $\mathcal{K}$ indexed by k.

Before a manual evaluation, we fix an arbitrary baseline system and use it to translate the segments $\mathcal{J}$. Then, each system $i \in \mathcal{I}$ produces a translation on segment $j \in \mathcal{J}$. One of the judges $k \in \mathcal{K}$ compares it with the baseline translation. The judge produces a preference judgment.

Let $u_{i,j,k}$ be the observed judgment that judge k assigns to a translation by system i on segment j, that is,

$$u_{i,j,k} = \begin{cases} 1 & \text{(preference for baseline)} \\ 2 & \text{(no preference)} \\ 3 & \text{(preference for system } i\text{)} \end{cases},$$

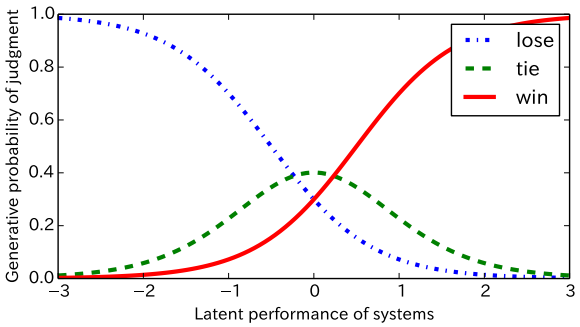


Figure 2: **ICC of graded response model for** $(b_1, b_2) = (-0.5, 0.5)$ **and** $a = 1.7$

and let $c \in \{1, 2, 3\}$ be the judgment label.

Each system i has its own latent performance $\theta_i \in \mathbb{R}$. Our goal is to estimate θ by using the observed judgments $U = \{u_{i,j,k}\}_{i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}}$.

4 Generative Judgment Model

We describe a statistical model for pairwise comparisons based on an IRT model.

4.1 Modified Graded Response Model

Based on the graded response model (GRM) proposed by Samejima (1968), we define a generative model of judgments. GRM deals with responses on ordered categories including ratings such as A+, A, B+ and B, and partial credits in tests. In our problem setting, judgments can be seen as partial credits. When a system beats a baseline translation, the system receives $c = 3$ credit. In the case of a tie, the system receives $c = 2$ credit. The system receives $c = 1$ credit when it lose to the baseline.

Let $\mathrm{P}^*_{jkc}(\theta_i)$ be the probability that judge k assigns judgment $\pi > c$ to a comparison on segment j between system i and a baseline.

$$\mathrm{P}^*_{jkc}(\theta_i) = \frac{1}{1 + \exp(-a_k(\theta_i - b_{jc}))},$$

where $\mathrm{P}^*_{jk0}(\theta_i) = 1, \mathrm{P}^*_{jk3}(\theta_i) = 0$. Parameters a and b are called *discrimination* and *difficulty* parameters, respectively. a represents the discriminablity or sensitivity of the judge, and b represents a segment-specific difficulty parameter. The discrimination parameter (a) is positive, and the difficulty parameter (b) satisfies $b_1 < b_2$, where b_1 corresponds to the difficulty of not losing to the baseline ($c > 1$), and b_2

corresponds to the difficulty of beating the baseline ($c > 2$).

The generative probability of judgment $u_{i,j,k}$ is defined as the difference in the probabilities defined above, that is,

$$\begin{aligned}\mathrm{P}_{jkc}(\theta_i) &= \mathrm{P}(u_{i,j,k} = c|\theta_i, b_j, a_k) \\ &= \mathrm{P}^*_{jkc-1}(\theta_i) - \mathrm{P}^*_{jkc}(\theta_i).\end{aligned}$$

This function is called *item characteristic curve* (ICC). Figure 2 illustrates the ICC in the GRM. The horizontal axis represents the latent performance of systems, and the vertical axis represents the generative probability of the judgments. This figure shows, for example, that the probability of the system with $\theta = 0$ beating the baseline is 0.3, whereas the system with $\theta = 1.0$ is much more likely to win. The discrimination parameter controls slope of the curves. If a is small, the probability drops a little when θ decreased.

The model described above is different from the original GRM, which assumed that the values of a are independent from question to question, and that each a belongs to exactly one question. However, in our problem setting, the judges evaluate multiple segments, and discrimination parameter a is independent from segment j. This modification means that the GRM can capture the judge's sensitivity.

4.2 Priors

We assign prior distributions to the parameters to obtain estimates stably. We assume Gaussian distributions on θ and b, that is, $\theta \sim \mathcal{N}(0, \tau^2)$ and $b_c \sim \mathcal{N}(\mu_{bc}, \sigma^2_{bc})$ $(c = 1, 2)$. The discrimination parameter is positive, so we assume a log Gaussian distribution on a, i.e., $\log(a) \sim \mathcal{N}(\mu_a, \sigma^2_a)$. Note that τ, μ, and σ are hyper parameters.

5 Parameter Estimation

We find the values of the parameters to maximize the log likelihood based on obtained judgments U:

$$\mathcal{L}(\theta, \xi) = \log\mathrm{P}(U, \theta, \xi).$$

We denote the parameters $a = \{a_k\}_{k\in\mathcal{K}}$ and $b = \{b_{j1}, b_{j2}\}_{j\in\mathcal{J}}$ to be ξ in this section.

5.1 Marginal Likelihood Maximization of Judge Sensitivity and Matchup Difficulty

Estimates are known to be inaccurate when all the parameters are optimized at once, so we first estimate the parameters ξ to maximize the marginal log likelihood w.r.t. the system performance θ.

$$\begin{aligned}m\mathcal{L}(\xi) &= \log\mathrm{P}(U, \xi) \\ &= \sum_{i\in\mathcal{I}} \log \int_{-\infty}^{\infty} \mathrm{P}(\theta)\mathrm{P}(U_i|\theta, \xi)d\theta + \log\mathrm{P}(\xi),\end{aligned}$$

where U_i is the set of judgments given to system i

The equation above can be approximated using Gauss-Hermite quadrature, i.e.,

$$\begin{aligned}m\mathcal{L}(\xi) &\approx \sum_{i\in\mathcal{I}} \log \sum_{t=1}^{T} \frac{1}{\sqrt{\pi}} w_t \mathrm{P}(U_i|\tau x_t, \xi) + \log\mathrm{P}(\xi) \\ w_t &= \frac{2^{T-1}T!\sqrt{\pi}}{T^2\left(H(x_t)\right)^2} \\ H(x_t) &= \left(2x_t - \frac{d}{dx_t}\right)^{T-1} \cdot 1,\end{aligned}$$

where a practically good approximation is obtained by taking $T \approx 20$.[2]

We solve the optimization problem using the gradient descent methods to maximize the approximated marginal likelihood. The inequality constraints on the parameters are handled by adding log barrier functions to the objective function.

5.2 Maximum A Posteriori (MAP) Estimation of System Performance

Given the estimates of ξ, we estimate the system performance $\theta = \{\theta_i\}_{i\in\mathcal{I}}$ by using MAP estimation.

We maximize the objective function,

$$\begin{aligned}\mathcal{L}(\theta) &= \log\mathrm{P}(U, \theta; \xi) \\ &= \sum_{i\in\mathcal{I}} \log\mathrm{P}(\theta_i) + \sum_{i\in\mathcal{I}} \log\mathrm{P}(U_i|\theta_i; \xi).\end{aligned}$$

The estimates of θ are obtained using the gradient descent method.

5.3 Discussion

So far we have assumed that the estimate is based on batch learning. However, it is known that active

[2] In this study, we set $T = 21$ to include $x = 0$.

learning can reduce the costs (i.e., the total number of comparisons) (Sakaguchi et al., 2014).

To extend our model to the active learning framework, one approach is to optimize the objective function online and actively select the next system to be compared based on criteria such as the uncertainty of the system's performance. We can apply stochastic gradient descent to the online optimization, which updates the estimates of the parameters using the gradients calculated based on a single comparison. This modification was left for future work.

6 Experiments

We conducted experiments on the WMT13 manual evaluation dataset for 10 language pairs.[3] For details of the evaluation data, see the overview of WMT13 (Bojar et al., 2013).

6.1 Setup

Models: Our method **(GRM)** was initialized using $a = 1.7, b = (-0.5, 0.5)$, and a θ value derived by summing up the judgments for each system and scaling θ to fit the prior distribution. For the hyper parameters, we set $\tau = \sqrt{2}, \mu_a = \log(1.7), \sigma_a = 1.0, \mu_b = (-0.5, 0.5), \sigma_b = 2.0$.

To compare with our method, we trained ExpectedWins **(EW)** (Bojar et al., 2013), the model by Hopkins and May (2013), **(HM)** and the two-stage crowdsourcing model proposed by Baba and Kashima (2013) **(TSt)**. We also trained TrueSkill **(TS)** (Sakaguchi et al., 2014), which was used to produce the gold score on this experiment.

We followed Sakaguchi et al. (2014), who also used the WMT13 datasets in their experiments, and initialized the HM and TS parameters. For TSt, we followed Baba and Kashima (2013).

Pairwise comparisons: The WMT dataset contains five-way partial rankings, so we converted the five-way partial rankings into pairwise comparisons. For example, given a five-way partial ranking A $>$ B $>$ C $>$ D $>$ E, we obtain ten pairwise comparisons A $>$ B, A $>$ C, A $>$ D, $\cdots$, and D $>$ E. We randomly sampled 800, 1,600, 3,200 and 6,400 pairwise comparisons from the whole dataset.

[3] http://statmt.org/wmt13/results.html

The training data differs between the models. For GRM and TSt, we first sampled five-way rankings that contained a baseline translation for each baseline system and obtained pairwise comparisons. For EW and HM, we first converted five-way rankings into pairwise comparisons and selected them at random.[4] TS first receives all the pairwise comparisons and selects the training data based on the active learning strategy, whereas we sampled the comparisons before running the other methods.

Gold scores: We followed the official evaluation procedure of the WMT14-15 (Bojar et al., 2014; Bojar et al., 2015) and made gold scores with TS. We produced 1,000 bootstrap-resampled datasets over all of the available comparisons. We then ran TS and collected the system scores. The gold score is the mean of the scores.

Evaluation metrics: We evaluated the models using the Pearson correlation coefficient and the normalized discounted cumulative gain (nDCG), comparing the estimated scores and gold scores. We used nDCG because we are often interested in ranks and scores, especially in MT competitions such as the WMT translation task.[5] These metrics were also used for experiments in Baba and Kashima (2013).

6.2 Results

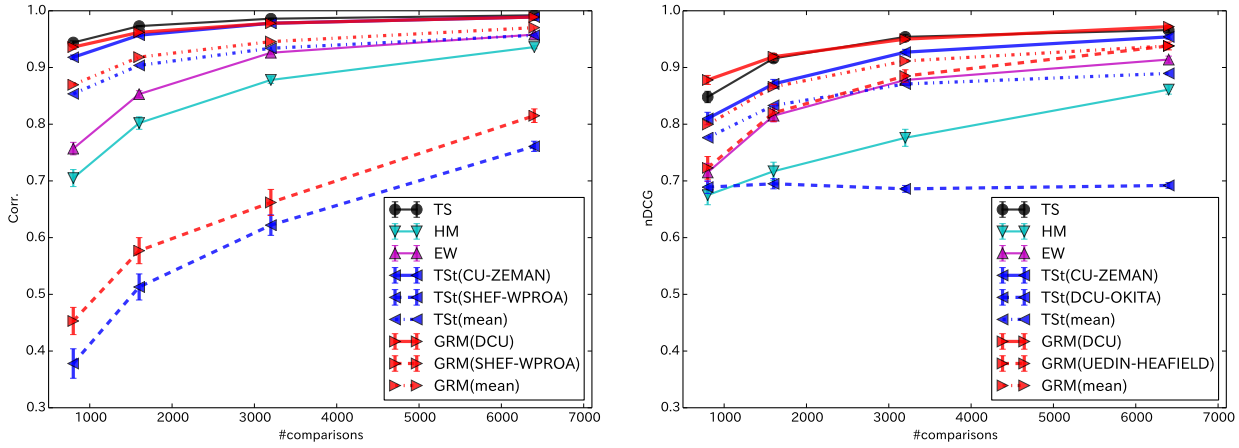
Figure 3 shows the correlation and nDCG between the estimated system performance and the gold scores for the WMT13 Spanish–English task. For the GRM and TSt, the baselines used in the evaluation are shown in parentheses in the labels. The other language pairs showed similar tendencies. The complete results for all language pairs can be found in the supplementary data files.

Note that the main contribution of our method is not to perform better than other methods in terms of correlation and nDCG to the gold scores, but to result in highly interpretable and robust estimates discussed later.

TS resulted in the highest correlation and nDCG. It is reasonable because the gold scores themselves were produced by TS, and because it estimates the

[4] We also applied the sampling procedure of GRM and TSt to EW and HM, but it made their estimation inaccurate.

[5] We did not use Spearman's rank correlation coefficient because it does not consider a margin between ranks.



(a) Correlation

(b) nDCG

Figure 3: **Correlation and nDCG comparing the estimated system performance and gold scores with the number of comparisons for the WMT13 Spanish–English task.** The baseline system is shown in parenthesis for TSt and GRM.

parameters using active learning, unlike the other models.

The GRM with the best baseline system (DCU) achieved almost the same scores as the TS, in terms of correlation and nDCG. Although the TSt with the best baseline resulted in accurate estimates in terms of correlation, it did not in terms of nDCG. With the worst baselines, the GRM and TSt both failed to replicate the gold scores, but the GRM was surprisingly accurate in terms of nDCG (even in the worst case). This implies that the GRM can effectively predict the top ranked systems.

6.3 Baseline Selection

It is likely that single pairwise comparisons do not work well if the baseline is very strong or weak. As shown in Figure 3, the baseline system influences the final result. When we used SHEF-WPROA as baseline, the estimated system performance was not accurate. This is because SHEF-WPROA loses 69.4% of the pairwise comparisons and fails to discriminate between the other systems. In contrast, DCU loses 34.5% and win 34.8% of the comparisons and discriminate the other systems successfully. Thus, when we used DCU as baseline, the best correlation and nDCG were achieved. Therefore, we must determine the appropriate baseline system before the comparisons.

One possible solution is to consider the system-level scores yielded by automatic evaluation metrics such as BLEU and METEOR. Figure 4 shows that we obtained relatively good results when we used a system whose system-level BLEU score and METEOR score[6] were close to the mean of all the systems. [7]

Noise(%)	0	10	20	30	40	50
Correlation						
GRM	.929	.917	.900	.879	.849	.807
HM	+.002	-.005	-.009	-.015	-.025	-.038
EW	-.025	-.028	-.035	-.038	-.040	-.046
nDCG						
GRM	.883	.867	.847	.822	.793	.752
HM	-.024	-.130	-.137	-.144	-.152	-.168
EW	-.035	-.054	-.064	-.060	-.060	-.069

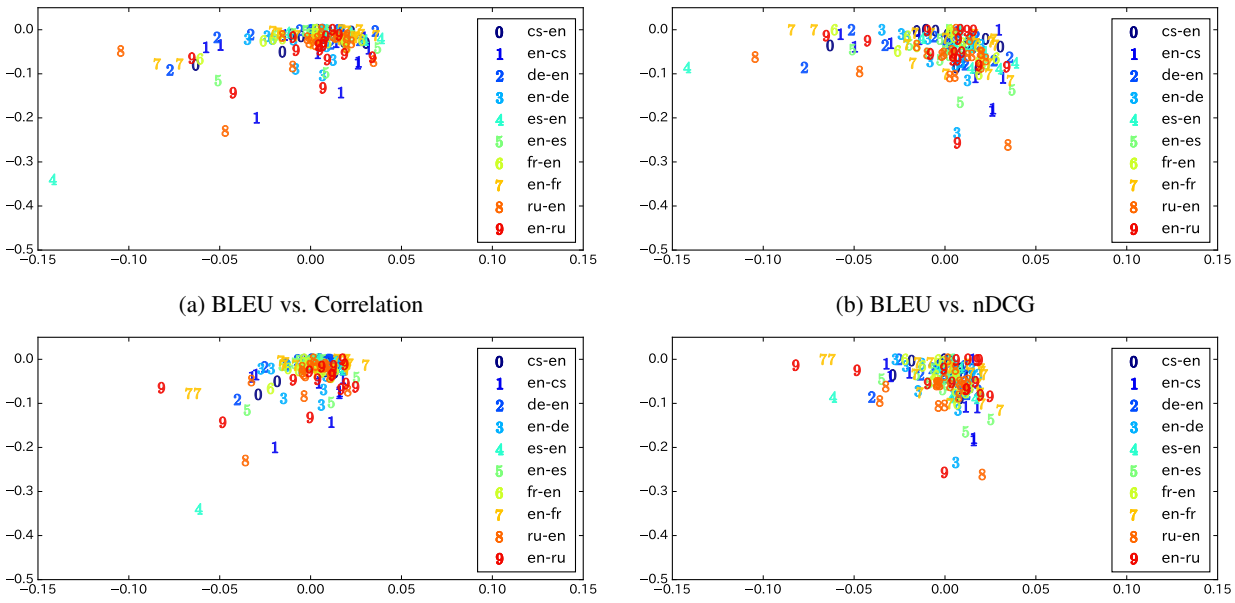
Table 1: **Correlation and nDCG between the estimated system performance and gold scores for the WMT13 Spanish–English task, based on noisy judges.** The values were averaged over all the datasets. The GRM scores were averaged over all baselines. The differences from the GRM are reported for the HM and EW.

6.4 Analysis of Judge Sensitivity

To investigate the robustness of the GRM, we simulated "noisy" judges. We selected a subset of

[6]BLEU and METEOR scores were given by the WMT13 organizers.

[7]The system-level scores can be found in the WMT13 Metrics Task dataset.



(a) BLEU vs. Correlation

(b) BLEU vs. nDCG

(c) METEOR vs. Correlation

(d) METEOR vs. nDCG

Figure 4: **Relationship between system-level BLEU/METEOR scores (horizontal) and correlation/nDCG scores (vertical).** The mean BLEU/METEOR was set to zero, and the best score was set to zero for each language pair.

judges and randomly changed their decisions based on a uniform distribution. The percentage of noisy judges varied between 10% and 50% (in increments of 10%).

We trained HM and EW on the simulated datasets. We excluded TS because it assumes that we can actively request more comparisons from judges when their decisions are ambiguous.

As shown in Table 1, the accuracy of the GRM was less affected by the noisy judges than HM and EW. This is because our model estimates judge-specific sensitivities and automatically reduces the influence of the noisy judges.

6.5 Analysis of the Interpretability of the Estimated Matchup Difficulty

Our model is a natural extension of the GRM Samejima (1968), so we can apply standard analyses for IRT models. Item information is one of the standard analysis methods and corresponds to sensitivity to a latent parameter of interest. Based on the item information, we can find which segment was difficult to be translated better than a baseline translation.

The item information is calculated using the esti-

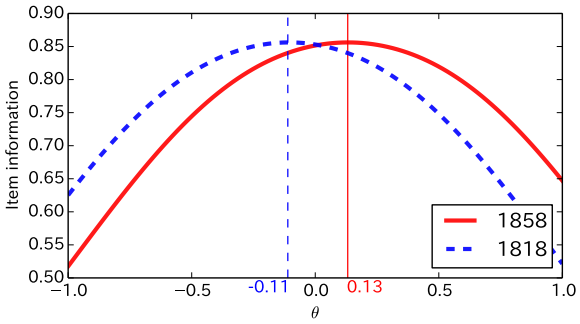


Figure 5: **Item information for the WMT13 Spanish–English task.** The DCU was used as a baseline. We used the averaged estimates of b on 100 sampled datasets with 6,400 comparisons to calculate the item information for all segments.

mated parameters ξ (Samejima, 1968), that is,

$$I_j(\theta) = -E\left[\frac{\partial^2 \mathcal{L}(\theta;\xi)}{\partial\theta^2}\right] = \sum_{c=1}^{3}\left[-\frac{\partial^2 \log \mathrm{P}_{jkc}(\theta)}{\partial\theta^2}\right]\mathrm{P}_{jkc} = \sum_{c=1}^{3}\frac{[\mathrm{P}^{*'}_{jkc-1}(\theta) - \mathrm{P}^{*'}_{jkc}(\theta)]^2}{\mathrm{P}^{*}_{jkc-1}(\theta) - \mathrm{P}^{*}_{jkc}(\theta)},$$

where $\mathrm{P}^{*'} = \partial \mathrm{P}^{*}/\partial\theta$.

Because the item information is only determined

Segment 1858: Difficult to beat the baseline translation.		
Source		Hasta 2007 los dos telescopios Keck situados en el volcán hawaiano de Mauna Kea eran considerados los más grandes del mundo.
Reference		Until 2007, the two Keck telescopes at the Hawaiian volcano, Mauna Kea, were the largest in the world.
DCU[baseline]		Until 2007, the two Keck telescopes located on the Hawaiian volcano Mauna of KEA were considered the largest in the world.
ONLINE-B	($\theta =$) 0.24	Until 2007 the two Keck telescopes located on the Hawaiian volcano Mauna Kea were considered the largest in the world.
UEDIN	0.12	Until 2007, the two Keck telescopes located on the Hawaiian volcano of Mauna Kea were considered the largest in the world.
LIMSI-NCODE-SOUL	0.10	Until 2007 the two Keck telescopes in the Hawaiian Mauna Kea volcano were considered the largest in the world.
CU-ZEMAN	-0.10	Until 2007, the two Keck telescope located in the volcano Mauna Kea hawaiano of were regarded as the world's largest.
JHU	-0.12	Until 2007, the two Telescope Keck located in the Kea volcano hawaiano of Mauna were considered the world's largest.
SHEF-WPROA	-0.92	Until 2007 the two telescope Keck located volcano hawaiano of Mauna KEA were regarded larger of world.
Segment 1818: Easy to beat the baseline translation.		
Source		Dependiendo de las tonalidades, algunas imágenes de galaxias espirales se convierten en verdaderas obras de arte.
Reference		Depending on the colouring, photographs of spiral galaxies can become genuine works of art.
DCU[baseline]		Depending on the drink, some images of galaxias galaxies become true works of art.
ONLINE-B	0.24	Depending on the shades, some images of spiral galaxies become true works of art.
UEDIN	0.12	(Same as ONLINE-B)
LIMSI-NCODE-SOUL	0.10	Depending on the color, some images of galaxies spirals become real works of art.
CU-ZEMAN	-0.10	Depending on the tonalidades, some images of spirals galaxies become true works of art.
JHU	-0.12	Depending on the tonalidades, some images of galaxies spirals become true works of art.
SHEF-WPROA	-0.92	Depending on the tonalidades, some images of galaxies spirals become real artwork.

Table 2: **Translation examples for the WMT13 Spanish–English task.** The reference is a correct translation given by the WMT organizers and was shown to human judges. Estimates of θ (averaged over 100 sampled datasets with 6,400 comparisons) are also reported in the table.

by segments and is independent of the judges, we set $a_k = 1$ $(k \in \mathcal{K})$.

Figure 5 gives two examples of the item information. The horizontal axis corresponds to the system performance θ, and the vertical axis represents the informativeness of a segment. This figure indicates that segment 1858 (red line) can effectively discriminate systems with $\theta \approx 0.13$, whereas segment 1818 (blue dashed line) is sensitive to those with $\theta \approx -0.11$. This means that systems with low θ tend to lose to a baseline translation on segment 1858, and the segment does not tell meaningful information on performance of the systems. However, they sometimes beat a baseline translation on segment 1818, and the segment can measure their performance accurately.

Table 2 shows translations for segments 1858 and 1818. We found that the baseline translation on segment 1818 was relatively good, whereas the baseline translation on segment 1858 contained wrong words such as "drink" and "galaxias". Consequently, systems with low θ tended to lose to the baseline on segment 1858 due to their wrong translation (see the translation of "hawaiano de Mauna Kea"). In contrast, some of the low-ranked systems beat the baseline on segment 1818, and the segment contributed to discriminate them.

The item information is used to design academic tests that can effectively capture students' abilities. It could analogously be used to preselect segments to be translated based on the item information in the MT evaluation.

7 Conclusion

We have addressed the task of manual judgment aggregation for MT evaluations. Our motivation was three folded: (1) to incorporate a judge's sensitivity to robustly measure a system's performance, (2) to maintain highly interpretable estimates, and (3) to handle with a newly submitted system.

To tackle these problems, we focused on pairwise comparisons with a fixed baseline translation so that we could apply the GRM model in IRT by using the analogy of standard academic tests. Unlike testing all pairwise combinations of systems, fixing baseline translations makes it easy to evaluate a newly submitted system. We demonstrated that our model gave robust and highly interpretable estimates on the WMT13 datasets.

In the future work, we will incorporate active learning to the proposed method so that we could reduce the total number of comparisons to obtain final results. Although we evaluated the correlation between the estimated system performance scores and the WMT official scores, other evaluation procedures might also be considered. For example, Hopkins and May (2013) considered model perplexity and Sakaguchi et al. (2014) compared accuracy. However, we cannot directly compare other methods to our method in terms of perplexity or accuracy because our method focuses on comparisons with a baseline translation, whereas they do not. It will be required to investigate correlation between the estimates and expert decisions.

Acknowledgments

We would like to thank Yukino Baba and Hisashi Kashima for providing an implementation of their method. We are also thankful for the useful comments from the anonymous reviewers.

References

Yukino Baba and Hisashi Kashima. 2013. Statistical quality estimation for general crowdsourcing tasks. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 554–562, New York, USA, August. ACM Press.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation (WMT)*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*, pages 12–58, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation (WMT)*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3-4):324–345.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 193–202, New York, New York, USA, February. ACM Press.

Mark Dras. 2015. Evaluating human pairwise preference judgments. *Computational Linguistics*, 41(2):337–345.

Shinsuke Goto, Donghui Lin, and Toru Ishida. 2014. Crowdsourcing for evaluating machine translation quality. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland, May. European Language Resources Association.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2015. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, FirstView:1–28.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. 2015. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 461–470, Lisbon, Portugal, June. Association for Computational Linguistics.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill™: A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 569–576, Vancouver, British Columbia, Canada, Demeber. MIT Press.

Mark Hopkins and Jonathan May. 2013. Models of translation competitions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1416–1424, Sofia, Bulgaria, August. Association for Computational Linguistics.

Philipp Koehn. 2012. Simulating human judgment in machine translation evaluation campaigns. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*, pages 179–184, Hongkong, China, December. International Speech Communication Association.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 588–593, Beijing, China, July. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*, pages 1–11, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Fumiko Samejima. 1968. Estimation of latent ability using a response pattern of graded scores. *ETS Research Bulletin Series*, 1968(1):i–169, June.

Variational Neural Machine Translation

Biao Zhang[1,2], **Deyi Xiong**[1]*, **Jinsong Su**[2], **Hong Duan**[2] **and Min Zhang**[1]
Provincial Key Laboratory for Computer Information Processing Technology
Soochow University, Suzhou, China 215006[1]
Xiamen University, Xiamen, China 361005[2]
zb@stu.xmu.edu.cn, {jssu,hduan}@xmu.edu.cn
{dyxiong, minzhang}@suda.edu.cn

Abstract

Models of neural machine translation are often from a discriminative family of encoder-decoders that learn a conditional distribution of a target sentence given a source sentence. In this paper, we propose a variational model to learn this conditional distribution for neural machine translation: a variational encoder-decoder model that can be trained end-to-end. Different from the vanilla encoder-decoder model that generates target translations from hidden representations of source sentences alone, the variational model introduces a *continuous latent variable* to explicitly model underlying semantics of source sentences and to guide the generation of target translations. In order to perform efficient posterior inference and large-scale training, we build a *neural posterior approximator* conditioned on both the source and the target sides, and equip it with a reparameterization technique to estimate the variational lower bound. Experiments on both Chinese-English and English-German translation tasks show that the proposed variational neural machine translation achieves significant improvements over the vanilla neural machine translation baselines.

1 Introduction

Neural machine translation (NMT) is an emerging translation paradigm that builds on a single and unified end-to-end neural network, instead of using a variety of sub-models tuned in a long training pipeline. It requires a much smaller memory than phrase- or syntax-based statistical machine translation (SMT) that typically has a huge phrase/rule table. Due to these advantages over traditional SMT system, NMT has recently attracted growing interests from both deep learning and machine translation community (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a; Luong et al., 2015b; Shen et al., 2015; Meng et al., 2015; Tu et al., 2016).

Current NMT models mainly take a discriminative *encoder-decoder* framework, where a *neural encoder* transforms source sentence $\mathbf{x}$ into distributed representations, and a *neural decoder* generates the corresponding target sentence $\mathbf{y}$ according to these representations[1] (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014). Typically, the underlying semantic representations of source and target sentences are learned in an implicit way in this framework, which heavily relies on the attention mechanism (Bahdanau et al., 2014) to identify semantic alignments between source and target words. Due to potential errors in these alignments, the attention-based context vector may be insufficient to capture the entire meaning of a source sentence, hence resulting in undesirable translation phenomena (Tu et al., 2016).

Unlike the vanilla encoder-decoder framework, we model underlying semantics of bilingual sentence pairs explicitly. We assume that there exists a continuous latent variable $\mathbf{z}$ from this underlying semantic space. And this variable, together with $\mathbf{x}$,

*Corresponding author

[1]In this paper, we use bold symbols to denote variables, and plain symbols to denote their values. Without specific statement, all variables are multivariate.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 521–530,
Austin, Texas, November 1-5, 2016.

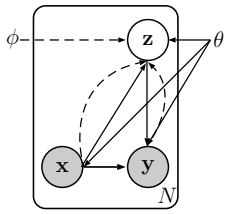


Figure 1: Illustration of VNMT as a directed graph. We use solid lines to denote the generative model $p_\theta(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})$, and dashed lines to denote the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p(\mathbf{z}|\mathbf{x},\mathbf{y})$. Both variational parameters ϕ and generative model parameters θ are learned jointly.

guides the translation process, i.e. $p(\mathbf{y}|\mathbf{z},\mathbf{x})$. With this assumption, the original conditional probability evolves into the following formulation:

$$p(\mathbf{y}|\mathbf{x}) = \int_z p(\mathbf{y}, z|\mathbf{x})d_z = \int_z p(\mathbf{y}|z,\mathbf{x})p(z|\mathbf{x})d_z \quad (1)$$

This brings in the benefits that the latent variable $\mathbf{z}$ can serve as a global semantic signal that is complementary to the attention-based context vector for generating good translations when the model learns undesirable attentions. However, although this latent variable enables us to explicitly model underlying semantics of translation pairs, the incorporation of it into the above probabilistic model has two challenges: 1) the posterior inference in this model is intractable; 2) large-scale training, which lays the ground for the data-driven NMT, is accordingly problematic.

In order to address these issues, we propose a variational encoder-decoder model to neural machine translation (VNMT), motivated by the recent success of variational neural models (Rezende et al., 2014; Kingma and Welling, 2014). Figure 1 illustrates the graphic representation of VNMT. As deep neural networks are capable of learning highly non-linear functions, we employ them to fit the latent-variable-related distributions, i.e. the prior and posterior, to make the inference tractable. The former is modeled to be conditioned on the source side alone $p_\theta(\mathbf{z}|\mathbf{x})$, because the source and target part of a sentence pair usually share the same semantics so that the source sentence should contain the prior information for inducing the underlying semantics. The latter, instead, is approximated from all observed variables $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$, i.e. both the source and the target sides. In order to efficiently train parameters, we apply a reparameterization technique (Rezende et al., 2014; Kingma and Welling, 2014) on the variational lower bound. This enables us to use standard stochastic gradient optimization for training the proposed model. Specifically, there are three essential components in VNMT (The detailed architecture is illustrated in Figure 2):

- A *variational neural encoder* transforms source/target sentence into distributed representations, which is the same as the encoder of NMT (Bahdanau et al., 2014) (see section 3.1).
- A *variational neural inferer* infers the representation of $\mathbf{z}$ according to the learned source representations (i.e. $p_\theta(\mathbf{z}|\mathbf{x})$) together with the target ones (i.e. $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$), where the reparameterization technique is employed (see section 3.2).
- And a *variational neural decoder* integrates the latent representation of $\mathbf{z}$ to guide the generation of target sentence (i.e. $p(\mathbf{y}|\mathbf{z},\mathbf{x})$) together with the attention mechanism (see section 3.3).

Augmented with the posterior approximation and reparameterization, our VNMT can still be trained end-to-end. This makes our model not only efficient in translation, but also simple in implementation. To train our model, we employ the conventional maximum likelihood estimation. Experiments on both Chinese-English and English-German translation tasks show that VNMT achieves significant improvements over several strong baselines.

2 Background: Variational Autoencoder

This section briefly reviews the variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014). Given an observed variable $\mathbf{x}$, VAE introduces a continuous latent variable $\mathbf{z}$, and assumes that $\mathbf{x}$ is generated from $\mathbf{z}$, i.e.,

$$p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \quad (2)$$

where θ denotes the parameters of the model. $p_\theta(\mathbf{z})$ is the prior, e.g, a simple Gaussian distribution. $p_\theta(\mathbf{x}|\mathbf{z})$ is the conditional distribution that models the generation procedure, typically estimated via a deep non-linear neural network.

Similar to our model, the integration of $\mathbf{z}$ in Eq. (2) imposes challenges on the posterior inference as

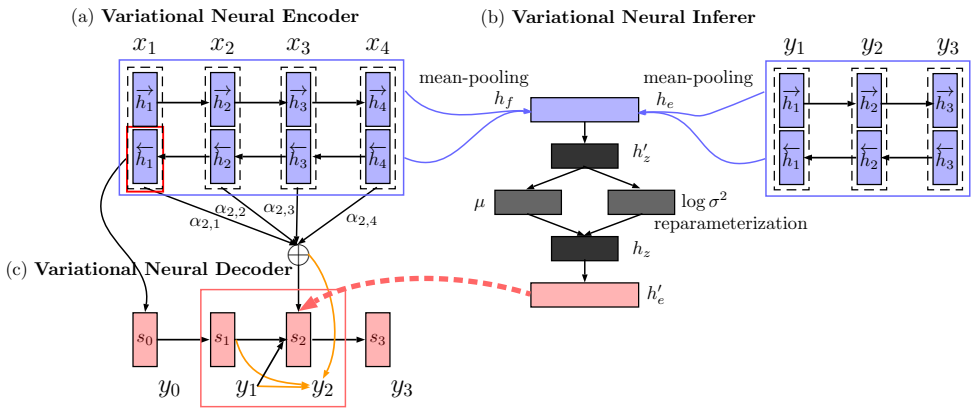


Figure 2: Neural architecture of VNMT. We use blue, gray and red color to indicate the encoder-related ($\mathbf{x}, \mathbf{y}$), underlying semantic ($\mathbf{z}$) and decoder-related ($\mathbf{y}$) representation respectively. The yellow lines show the flow of information employed for target word prediction. The dashed red line highlights the incorporation of latent variable $\mathbf{z}$ into target prediction. f and e represent the source and target language respectively.

well as large-scale learning. To tackle these problems, VAE adopts two techniques: *neural approximation* and *reparameterization*.

Neural Approximation employs deep neural networks to approximate the posterior inference model $q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ denotes the variational parameters. For the posterior approximation, VAE regards $q_\phi(\mathbf{z}|\mathbf{x})$ as a diagonal Gaussian $\mathcal{N}(\mu, \text{diag}(\sigma^2))$, and parameterizes its mean μ and variance σ^2 with deep neural networks.

Reparameterization reparameterizes $\mathbf{z}$ as a function of μ and σ, rather than using the standard sampling method. In practice, VAE leverages the "location-scale" property of Gaussian distribution, and uses the following reparameterization:

$$\tilde{z} = \mu + \sigma \odot \epsilon \tag{3}$$

where ϵ is a standard Gaussian variable that plays a role of introducing noises, and $\odot$ denotes an element-wise product.

With these two techniques, VAE tightly incorporates both the generative model $p_\theta(\mathbf{x}|\mathbf{z})$ and the posterior inference model $q_\phi(\mathbf{z}|\mathbf{x})$ into an end-to-end neural network. This facilitates its optimization since we can apply the standard backpropagation to compute the gradient of the following variational lower bound:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = &- \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \\ &+\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x})\end{aligned} \tag{4}$$

$\text{KL}(Q||P)$ is the Kullback-Leibler divergence between Q and P. Intuitively, VAE can be considered as a regularized version of the standard autoencoder. It makes use of the latent variable $\mathbf{z}$ to capture the variations ϵ in the observed variable $\mathbf{x}$.

3 Variational Neural Machine Translation

Different from previous work, we introduce a latent variable $\mathbf{z}$ to model the underlying semantic space as a global signal for translation. Formally, given the definition in Eq. (1) and Eq. (4), the variational lower bound of VNMT can be formulated as follows:

$$\begin{aligned}\mathcal{L}_{\text{VNMT}}(\theta, \phi; \mathbf{x}, \mathbf{y}) = &-\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) \\ &+\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})]\end{aligned} \tag{5}$$

where $p_\theta(\mathbf{z}|\mathbf{x})$ is our prior model, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ is our posterior approximator, and $p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ is the decoder with the guidance from $\mathbf{z}$. Based on this formulation, VNMT can be decomposed into three components, each of which is modeled by a neural network: a *variational neural inferer* that models $p_\theta(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ (see part (b) in Figure 2), a *variational neural decoder* that models $p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ (see part (c) in Figure 2), and a *variational neural encoder* that provides distributed representations of a source/target sentence for the above two modules (see part (a) in Figure 2). Following the information flow illustrated in Figure 2, we describe part (a), (b) and (c) successively.

3.1 Variational Neural Encoder

As shown in Figure 2 (a), the variational neural encoder aims at encoding an input sequence $(w_1, w_2,$

$\ldots, w_T$) into continuous vectors. In this paper, we adopt the encoder architecture proposed by Bahdanau et al. (2014), which is a bidirectional RNN with a forward and backward RNN. The forward RNN reads the sequence from left to right while the backward RNN in the opposite direction (see the parallel arrows in Figure 2 (a)):

$$\begin{aligned} \overrightarrow{h}_i &= \text{RNN}(\overrightarrow{h}_{i-1}, E_{w_i}) \\ \overleftarrow{h}_i &= \text{RNN}(\overleftarrow{h}_{i+1}, E_{w_i}) \end{aligned} \tag{6}$$

where $E_{w_i} \in \mathbb{R}^{d_w}$ is the embedding for word w_i, and $\overrightarrow{h}_i$, $\overleftarrow{h}_i$ are hidden states generated in two directions. Following Bahdanau et al. (2014), we employ the Gated Recurrent Unit (GRU) as our RNN unit due to its capacity in capturing long-distance dependencies.

We further concatenate each pair of hidden states at each time step to build a set of *annotation* vectors ($\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$), $\mathbf{h}_i^T = \left[\overrightarrow{h}_i^T; \overleftarrow{h}_i^T\right]$. In this way, each annotation vector $\mathbf{h}_i$ encodes information about the i-th word with respect to all the other surrounding words in the sequence. Therefore, these annotation vectors are desirable for the following modeling.

We use this encoder to represent both the source sentence $\{x_i\}_{i=1}^{T_f}$ and the target sentence $\{y_i\}_{i=1}^{T_e}$ (see the blue color in Figure 2). Accordingly, our encoder generates both the source annotation vectors $\{\mathbf{h}_i\}_{i=1}^{T_f} \in \mathbb{R}^{2d_f}$ and the target annotation vectors $\{\mathbf{h}'_i\}_{i=1}^{T_e} \in \mathbb{R}^{2d_e}$. The source vectors flow into the inferer and decoder while the target vectors the posterior approximator.

3.2 Variational Neural Inferer

A major challenge of variational models is how to model the latent-variable-related distributions. In VNMT, we employ neural networks to model both the prior $p_\theta(\mathbf{z}|\mathbf{x})$ and the posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$, and let them subject to a multivariate Gaussian distribution with a diagonal covariance structure.[2] As shown in Figure 1, these two distributions mainly differ in their conditions.

[2]The reasons of choosing Gaussian distribution are twofold: 1) it is a natural choice for modeling continuous variables; 2) it belongs to the family of "location-scale" distributions, which is required for the following reparameterization.

3.2.1 Neural Posterior Approximator

Exactly modeling the true posterior $p(\mathbf{z}|\mathbf{x}, \mathbf{y})$ exactly usually intractable. Therefore, we adopt an approximation method to simplify the posterior inference. Conventional models typically employ the *mean-field* approaches. However, a major limitation of this approach is its inability to capture the true posterior of $\mathbf{z}$ due to its oversimplification. Following the spirit of VAE, we use neural networks for better approximation in this paper, and assume the approximator has the following form:

$$q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}, \mathbf{y}), \sigma(\mathbf{x}, \mathbf{y})^2\mathbf{I}) \tag{7}$$

The mean μ and s.d. σ of the approximate posterior are the outputs of neural networks based on the observed variables $\mathbf{x}$ and $\mathbf{y}$ as shown in Figure 2 (b).

Starting from the variational neural encoder, we first obtain the source- and target-side representation via a *mean-pooling* operation over the annotation vectors, i.e. $\mathbf{h}_f = \frac{1}{T_f}\sum_i^{T_f} \mathbf{h}_i, \mathbf{h}_e = \frac{1}{T_e}\sum_i^{T_e} \mathbf{h}'_i$. With these representations, we perform a non-linear transformation that projects them onto our concerned latent semantic space:

$$\mathbf{h}'_z = g(W_z^{(1)}[\mathbf{h}_f; \mathbf{h}_e] + b_z^{(1)}) \tag{8}$$

where $W_z^{(1)} \in \mathbb{R}^{d_z \times 2(d_f + d_e)}, b_z^{(1)} \in \mathbb{R}^{d_z}$ is the parameter matrix and bias term respectively, d_z is the dimensionality of the latent space, and $g(\cdot)$ is an element-wise activation function, which we set to be $\tanh(\cdot)$ throughout our experiments.

In this latent space, we obtain the abovementioned Gaussian parameters μ and $\log\sigma^2$ through linear regression:

$$\mu = W_\mu \mathbf{h}'_z + b_\mu, \;\; \log\sigma^2 = W_\sigma \mathbf{h}'_z + b_\sigma \tag{9}$$

where $\mu, \log\sigma^2$ are both d_z-dimension vectors.

3.2.2 Neural Prior Model

Different from the posterior, we model (rather than approximate) the prior as follows:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu'(\mathbf{x}), \sigma'(\mathbf{x})^2\mathbf{I}) \tag{10}$$

We treat the mean μ' and s.d. σ' of the prior as neural functions of source sentence $\mathbf{x}$ alone. This is sound and reasonable because bilingual sentences are semantically equivalent, suggesting that either $\mathbf{y}$ or $\mathbf{x}$

is capable of inferring the underlying semantics of sentence pairs, i.e., the representation of latent variable $\mathbf{z}$.

The neural model for the prior $p_\theta(\mathbf{z}|\mathbf{x})$ is the same as that (i.e. Eq (8) and (9)) for the posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$, except for the absence of $\mathbf{h}_e$. Besides, the parameters for the prior are independent of those for the posterior.

To obtain a representation for latent variable $\mathbf{z}$, we employ the same technique as the Eq. (3) and reparameterized it as $\mathbf{h}_z = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$. During decoding, however, due to the absence of target sentence $\mathbf{y}$, we set $\mathbf{h}_z$ to be the mean of $p_\theta(\mathbf{z}|\mathbf{x})$, i.e., μ'. Intuitively, the reparameterization bridges the gap between the generation model $p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})$ and the inference model $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$. In other words, it connects these two neural networks. This is important since it enables the stochastic gradient optimization via standard backpropagation.

We further project the representation of latent variable $\mathbf{h}_z$ onto the target space for translation:

$$\mathbf{h}'_e = g(W_z^{(2)}\mathbf{h}_z + b_z^{(2)}) \tag{11}$$

where $\mathbf{h}'_e \in \mathbb{R}^{d'_e}$. The transformed $\mathbf{h}'_e$ is then integrated into our decoder. Notice that because of the noise from ϵ, the representation $\mathbf{h}'_e$ is not fixed for the same source sentence and model parameters. This is crucial for VNMT to learn to avoid overfitting.

3.3 Variational Neural Decoder

Given the source sentence $\mathbf{x}$ and the latent variable $\mathbf{z}$, our decoder defines the probability over translation $\mathbf{y}$ as a joint probability of ordered conditionals:

$$p(\mathbf{y}|\mathbf{z},\mathbf{x}) = \prod_{j=1}^{T_e} p(y_j|y_{<j},\mathbf{z},\mathbf{x}) \tag{12}$$

$$\text{where} \quad p(y_j|y_{<j},\mathbf{z},\mathbf{x}) = g'(y_{j-1}, s_{j-1}, c_j)$$

The feed forward model $g'(\cdot)$ (see the yellow arrows in Figure 2) and context vector $c_j = \sum_i \alpha_{ji}\mathbf{h}_i$ (see the "$\oplus$" in Figure 2) are the same as (Bahdanau et al., 2014). The difference between our decoder and Bahdanau et al.'s decoder (2014) lies in that in addition to the context vector, our decoder integrates the representation of the latent variable, i.e. $\mathbf{h}'_e$, into the computation of s_j, which is denoted by the bold dashed red arrow in Figure 2 (c).

Formally, the hidden state s_j in our decoder is calculated by[3]

$$\begin{aligned}
s_j &= (1-u_j)\odot s_{j-1} + u_j \odot \tilde{s}_j, \\
\tilde{s}_j &= \tanh(WE_{y_j} + U[r_j \odot s_{j-1}] + Cc_j + V\mathbf{h}'_e) \\
u_j &= \sigma(W_u E_{y_j} + U_u s_{j-1} + C_u c_j + V_u \mathbf{h}'_e) \\
r_j &= \sigma(W_r E_{y_j} + U_r s_{j-1} + C_r c_j + V_r \mathbf{h}'_e)
\end{aligned}$$

Here, r_j, u_j, $\tilde{s}_j$ denotes the reset gate, update gate and candidate activation in GRU respectively, and $E_{y_j} \in \mathbb{R}^{d_w}$ is the word embedding for target word. $W, W_u, W_r \in \mathbb{R}^{d_e \times d_w}$, $U, U_u, U_r \in \mathbb{R}^{d_e \times d_e}$, $C, C_u, C_r \in \mathbb{R}^{d_e \times 2d_f}$, and $V, V_u, V_r \in \mathbb{R}^{d_e \times d'_e}$ are parameter weights. The initial hidden state s_0 is initialized in the same way as Bahdanau et al. (2014) (see the arrow to s_0 in Figure 2).

In our model, the latent variable can affect the representation of hidden state s_j through the gate between r_j and u_j. This allows our model to access the semantic information of $\mathbf{z}$ indirectly since the prediction of y_{j+1} depends on s_j. In addition, when the model learns wrong attentions that lead to bad context vector c_j, the semantic representation $\mathbf{h_e}'$ can help to guide the translation process .

3.4 Model Training

We use the Monte Carlo method to approximate the expectation over the posterior in Eq. (5), i.e. $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\cdot] \simeq \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{y}|\mathbf{x},\mathbf{h}_z^{(l)})$, where L is the number of samples. The joint training objective for a training instance $(\mathbf{x},\mathbf{y})$ is defined as follows:

$$\begin{aligned}
\mathcal{L}(\theta,\phi) \simeq & -\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) \\
& + \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{T_e} \log p_\theta(y_j|y_{<j},\mathbf{x},\mathbf{h}_z^{(l)})
\end{aligned} \tag{13}$$

$$\text{where } \mathbf{h}_z^{(l)} = \mu + \sigma \odot \epsilon^{(l)} \text{ and } \epsilon^{(l)} \sim \mathcal{N}(0,\mathbf{I})$$

The first term is the KL divergence between two Gaussian distributions which can be computed and differentiated without estimation (see (Kingma and Welling, 2014) for details). And the second term is the approximate expectation, which is also differentiable. Suppose that L is 1 (which is used in our experiments), then our second term will be degenerated to the objective of conventional NMT. Intuitively, VNMT is exactly a regularized version of

[3] We omit the bias term for clarity.

System	MT05	MT02	MT03	MT04	MT06	MT08	AVG
Moses	33.68	34.19	34.39	35.34	29.20	22.94	31.21
GroundHog	31.38	33.32	32.59	35.05	29.80	22.82	30.72
VNMT w/o KL	31.40	33.50	32.92	34.95	28.74	22.07	30.44
VNMT	32.25	**34.50**$^{++}$	33.78^{++}	**36.72**$^{\Uparrow ++}$	**30.92**$^{\Uparrow ++}$	**24.41**$^{\uparrow ++}$	**32.07**

Table 1: BLEU scores on the NIST Chinese-English translation task. **AVG** = average BLEU scores on test sets. We highlight the best results in bold for each test set. "↑/⇑": significantly better than *Moses* ($p < 0.05$/$p < 0.01$); "+/++": significantly better than *GroundHog* ($p < 0.05$/$p < 0.01$);

NMT, where the introduced noise ϵ increases its robustness, and reduces overfitting. We verify this point in our experiments.

Since the objective function in Eq. (13) is differentiable, we can optimize the model parameter θ and variational parameter ϕ jointly using standard gradient ascent techniques.

4 Experiments

4.1 Setup

To evaluate the effectiveness of the proposed VNMT, we conducted experiments on both Chinese-English and English-German translation tasks. Our Chinese-English training data[4] consists of 2.9M sentence pairs, with 80.9M Chinese words and 86.4M English words respectively. We used the NIST MT05 dataset as the development set, and the NIST MT02/03/04/06/08 datasets as the test sets for the Chinese-English task. Our English-German training data[5] consists of 4.5M sentence pairs with 116M English words and 110M German words[6]. We used the newstest2013 (3000 sentences) as the development set, and the newstest2014 (2737 sentences) as the test set for English-German translation. We employed the case-insensitive BLEU-4 (Papineni et al., 2002) metric to evaluate translation quality, and paired bootstrap sampling (Koehn, 2004) for significance test.

We compared our model against two state-of-the-art SMT and NMT systems:

- *Moses* (Koehn et al., 2007): a phrase-based SMT system.
- *GroundHog* (Bahdanau et al., 2014): an attention-based NMT system.

Additionally, we also compared with a variant of *VNMT*, which does not contain the KL part in the objective (*VNMT w/o KL*). This is achieved by setting $\mathbf{h}_z$ to μ'.

For *Moses*, we adopted all the default settings except for the language model. We trained a 4-gram language model on the Xinhua section of the English Gigaword corpus (306M words) using the SRILM[7] toolkit with modified Kneser-Ney smoothing. Importantly, we used all words in the vocabulary.

For *GroundHog*, we set the maximum length of training sentences to be 50 words, and preserved the most frequent 30K (Chinese-English) and 50K (English-German) words as both the source and target vocabulary , covering approximately 98.9%/99.2% and 97.3%/93.3% on the source and target side of the two parallel corpora respectively . All other words were represented by a specific token "UNK". Following Bahdanau et al. (2014), we set $d_w = 620$, $d_f = 1000$, $d_e = 1000$, and $M = 80$. All other settings are the same as the default configuration (for *RNNSearch*). During decoding, we used the beam-search algorithm, and set beam size to 10.

For *VNMT*, we initialized its parameters with the trained *RNNSearch* model. The settings of our model are the same as that of *GroundHog*, except for some parameters specific to VNMT. Following VAE, we set the sampling number $L = 1$. Additionally, we set $d'_e = d_z = 2d_f = 2000$ according to preliminary experiments. We used the Adadelta algorithm for model training with $\rho = 0.95$. With regard to the source and target encoders, we shared their recurrent parameters but not word embeddings.

We implemented our VNMT based on *GroundHog*[8]. Both NMT systems are trained on a Telsa K40

[4]This corpus consists of LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hong Kong Hansards/Laws/News).

[5]This corpus is from the WMT'14 training data (Jean et al., 2015; Luong et al., 2015a)

[6]The preprocessed data can be found and downloaded from http://nlp.stanford.edu/projects/nmt/

[7]http://www.speech.sri.com/projects/srilm/download.html

[8]Our code is publicly available at

System	MT05	MT02	MT03	MT04	MT06	MT08
GroundHog	18.23	22.20	20.19	21.67	19.11	13.41
VNMT	**21.31**	**26.02**	**23.78**	**25.81**	**21.81**	**15.59**

Table 2: BLEU scores on the new dataset. All improvements are significant at $p < 0.01$.

System	Architecture	BLEU
Existing end-to-end NMT systems		
Jean et al. (2015)	RNNSearch	16.46
Jean et al. (2015)	RNNSearch + unk replace	18.97
Jean et al. (2015)	RNNsearch + unk replace + large vocab	19.40
Luong et al. (2015a)	LSTM with 4 layers + dropout + local att. + unk replace	20.90
Our end-to-end NMT systems		
this work	RNNSearch	16.40
	VNMT	17.13^{++}
	VNMT + unk replace	19.58^{++}

Table 3: BLEU scores on the English-German translation task.

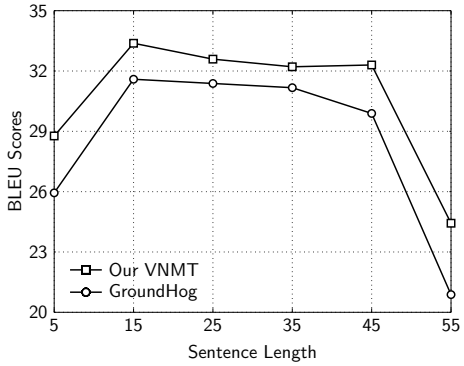


Figure 3: BLEU scores on different groups of source sentences in terms of their length.

GPU. In one hour, *GroundHog* processes about 1100 batches, while our *VNMT* processes 630 batches.

4.2 Results on Chinese-English Translation

Table 1 summarizes the BLEU scores of different systems on the Chinese-English translation tasks. Clearly VNMT significantly improves translation quality in terms of BLEU on most cases, and obtains the best average results that gain 0.86 and 1.35 BLEU points over *Moses* and *GroundHog* respectively. Besides, without the KL objective, *VNMT w/o KL* obtains even worse results than GroundHog. These results indicate the following two points: 1) explicitly modeling underlying semantics by a latent variable indeed benefits neural machine translation, and 2) the improvements of our model are not from enlarging the network.

https://github.com/DeepLearnXMU/VNMT.

4.3 Results on Long Sentences

We further testify VNMT on long sentence translation where the vanilla NMT usually suffers from attention failures (Tu et al., 2016; Bentivogli et al., 2016). We believe that the global latent variable can play an important role on long sentence translation.

Our first experiment is carried out on 6 disjoint groups according to the length of source sentences in our test sets. Figure 3 shows the BLEU scores of two neural models. We find that the performance curve of our VNMT model always appears to be on top of that of *GroundHog* with a certain margin. Specifically, on the final group with the longest source sentences, our VNMT obtains the biggest improvement (3.55 BLEU points). Overall, these obvious improvements on all groups in terms of the length of source sentences indicate that the global guidance from the latent variable benefits our VNMT model.

Our second experiment is carried out on a synthetic dataset where each new source sentence is a concatenation of neighboring source sentences in the original test sets. As a result, the average length of source sentences in the new dataset (> 50) is almost twice longer than the original one. Translation results is summarized in Table 2, where our VNMT obtains significant improvements on all new test sets. This further demonstrates the advantage of introducing the latent variable.

4.4 Results on English-German Translation

Table 3 shows the results on English-German translation. We also provide several existing NMT sys-

Source	两 国 官员 确定 了 今后 会谈 的 日程 和 模式 , 建立 起 进行 持续 对话 的 机制 , 此举 标志 着 巴 印 对话 进程 在 中断 两 年 后 重新 启动 , 为 两 国 逐步 解决 包括 克什米 尔 争端 在内 的 所有 悬而未决 的 问题 奠定 了 基础 , 体现 了 双方 可贵 的 和平 诚意 。
Reference	*the officials of the two countries have established the mechanism for continued dialogue down the road, including a confirmed schedule and model of the talks. this symbolizes the restart of the dialogue process between pakistan and india after an interruption of two years and has paved a foundation for the two countries to sort out gradually all the questions hanging in the air, including the kashmir dispute. it is also a realization of their precious sincerity for peace.*
Moses	*officials of the two countries set the agenda for future talks , and the pattern of a continuing dialogue mechanism . this marks a break in the process of dialogue between pakistan and india , two years after the restart of the two countries including kashmir dispute to gradually solve all the outstanding issues have laid the foundation of the two sides showed great sincerity in peace .*
GroundHog	*the two countries have decided to set up a mechanism for conducting continuous dialogue on the agenda and mode of the talks . this indicates that the ongoing dialogue between the two countries has laid the foundation for the gradual settlement of all outstanding issues including the dispute over kashmir .*
VNMT	*the officials of the two countries set up a mechanism for holding a continuous dialogue on the agenda and mode of the future talks, and this indicates that the ongoing dialogue between pakistan and india has laid a foundation for resolving all outstanding issues , including the kashmir disputes , and this serves as a valuable and sincere peace sincerity .*

Table 4: Translation examples of different systems. We highlight important parts in red color.

tems that use the same training, development and testing data. The results show that VNMT significantly outperforms GroundHog and achieves a significant gain of 0.73 BLEU points ($p < 0.01$). With unknown word replacement (Jean et al., 2015; Luong et al., 2015a), VNMT reaches the performance level that is comparable to the previous state-of-the-art NMT results.

4.5 Translation Analysis

Table 4 shows a translation example that helps understand the advantage of VNMT over NMT . As the source sentence in this example is long (more than 40 words), the translation generated by *Moses* is relatively messy and incomprehensible. In contrast, translations generated by neural models (both *GroundHog* and *VNMT*) are much more fluent and comprehensible. However, there are essential differences between *GroundHog* and our *VNMT*. Specifically, *GroundHog* does not translate the phrase "官员" at the beginning of the source sentence. The translation of the clause "体现 了 双方 可贵 的 和平 诚意 。" at the end of the source sentence is completely lost. In contrast, our VNMT model does not miss or mistake these fragments and can convey the meaning of entire source sentence to the target side.

From these examples, we can find that although attention networks can help NMT trace back to relevant parts of source sentences for predicting target translations, capturing the semantics of entire sentences still remains a big challenge for neural machine translation. Since NMT implicitly models variable-length source sentences with fixed-size hidden vectors, some details of source sentences (e.g., the red sequence of words in Table 4) may not be encoded in these vectors at all. VNMT seems to be able to capture these details through a latent variable that explicitly model underlying semantics of source sentences. The promising results suggest that VNMT provides a new mechanism to deal with sentence semantics.

5 Related Work

5.1 Neural Machine Translation

Neural machine translation starts from the sequence to sequence learning, where Sutskever et al. (2014) employ two multilayered Long Short-Term Memory (LSTM) models that first encode a source sentence into a single vector and then decode the translation word by word until a special end token is generated. In order to deal with issues caused by encoding all source-side information into a fixed-length vector, Bahdanau et al. (2014) introduce attention-based

NMT that aims at automatically concentrating on relevant source parts for predicting target words during decoding. The incorporation of attention mechanism allows NMT to cope better with long sentences, and makes it really comparable to or even superior to conventional SMT.

Following the success of attentional NMT, a number of approaches and models have been proposed for NMT recently, which can be grouped into different categories according to their motivations: dealing with rare words or large vocabulary (Jean et al., 2015; Luong et al., 2015b; Sennrich et al., 2015), learning better attentional structures (Luong et al., 2015a), integrating SMT techniques (Cheng et al., 2015; Shen et al., 2015; Feng et al., 2016; Tu et al., 2016), memory network (Meng et al., 2015), etc. All these models are designed within the discriminative encoder-decoder framework, leaving the explicit exploration of underlying semantics with a variational model an open problem.

5.2 Variational Neural Model

In order to perform efficient inference and learning in directed probabilistic models on large-scale dataset, Kingma and Welling (2014) as well as Rezende et al. (2014) introduce variational neural networks. Typically, these models utilize an neural inference model to approximate the intractable posterior, and optimize model parameters jointly with a reparameterized variational lower bound using the standard stochastic gradient technique. This approach is of growing interest due to its success in various tasks.

Kingma et al. (2014) revisit the approach to semi-supervised learning with generative models and further develop new models that allow effective generalization from a small labeled dataset to a large unlabeled dataset. Chung et al. (2015) incorporate latent variables into the hidden state of a recurrent neural network, while Gregor et al. (2015) combine a novel spatial attention mechanism that mimics the foveation of human eyes, with a sequential variational auto-encoding framework that allows the iterative construction of complex images. Very recently, Miao et al. (2015) propose a generic variational inference framework for generative and conditional models of text.

The most related work is that of Bowman et al. (2015), where they develop a variational autoencoder for unsupervised generative language modeling. The major difference is that they focus on the monolingual language model, while we adapt this technique to bilingual translation. Although variational neural models have been widely used in NLP tasks and the variational decoding has been investigated for SMT (Li et al., 2009), the adaptation and utilization of variational neural model to neural machine translation, to the best of our knowledge, has never been investigated before.

6 Conclusion and Future Work

In this paper, we have presented a variational model for neural machine translation that incorporates a continuous latent variable to model the underlying semantics of sentence pairs. We approximate the posterior distribution with neural networks and reparameterize the variational lower bound. This enables our model to be an end-to-end neural network that can be optimized through the stochastic gradient algorithms. Comparing with the conventional attention-based NMT, our model is better at translating long sentences. It also greatly benefits from a special regularization term brought with this latent variable. Experiments on Chinese-English and English-German translation tasks verified the effectiveness of our model.

In the future, since the latent variable in our model is at the sentence level, we want to explore more fine-grained latent variables for neural machine translation, such as the *Recurrent Latent Variable Model* (Chung et al., 2015). We are also interested in applying our model to other similar tasks.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos 61303082, 61672440, 61622209 and 61403269), Natural Science Foundation of Fujian Province (Grant No. 2016J05161), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355), and Research fund of the Provincial Key Laboratory for Computer Information Processing Technology in Soochow University (Grant No. KJS1520). We also thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

L. Bentivogli, A. Bisazza, M. Cettolo, and M. Federico. 2016. Neural versus Phrase-Based Machine Translation Quality: a Case Study. *ArXiv e-prints*, August.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. 2015. Generating Sentences from a Continuous Space. *ArXiv e-prints*, November.

Y. Cheng, S. Shen, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. 2015. Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation. *ArXiv e-prints*, December.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, pages 1724–1734, October.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Proc. of NIPS*.

S. Feng, S. Liu, M. Li, and M. Zhou. 2016. Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model. *ArXiv e-prints*, January.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proc. of ACL-IJCNLP*, pages 1–10, July.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proc. of EMNLP*, pages 1700–1709, October.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. of ICLR*.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proc. of NIPS*, pages 3581–3589.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. of ACL*, pages 593–601, August.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421, September.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. of ACL-IJCNLP*, pages 11–19, July.

F. Meng, Z. Lu, Z. Tu, H. Li, and Q. Liu. 2015. A Deep Memory-based Architecture for Sequence-to-Sequence Learning. *ArXiv e-prints*, June.

Y. Miao, L. Yu, and P. Blunsom. 2015. Neural Variational Inference for Text Processing. *ArXiv e-prints*, November.

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. of ICML*, pages 1278–1286.

R. Sennrich, B. Haddow, and A. Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *ArXiv e-prints*, August.

S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. 2015. Minimum Risk Training for Neural Machine Translation. *ArXiv e-prints*, December.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *CoRR*, abs/1601.04811.

Towards a Convex HMM Surrogate for Word Alignment

Andrei Arsene Simion
Columbia University *
New York, NY, 10011
aas2148@columbia.edu

Michael Collins
Columbia University†
Computer Science
New York, NY, 10027
mc3354@columbia.edu

Clifford Stein
Columbia University
IEOR Department
New York, NY, 10027
cs2035@columbia.edu

Abstract

Among the alignment models used in statistical machine translation (SMT), the hidden Markov model (HMM) is arguably the most elegant: it performs consistently better than IBM Model 3 and is very close in performance to the much more complex IBM Model 4. In this paper we discuss a model which combines the structure of the HMM and IBM Model 2. Using this surrogate, our experiments show that we can attain a similar level of alignment quality as the HMM model implemented in GIZA++ (Och and Ney, 2003). For this model, we derive its convex relaxation and show that it too has strong performance despite not having the local optima problems of non-convex objectives. In particular, the word alignment quality of this new convex model is significantly above that of the standard IBM Models 2 and 3, as well as the popular (and still non-convex) IBM Model 2 variant of (Dyer et al., 2013).

1 Introduction

The IBM translation models are widely used in modern statistical translation systems. Typically, one seeds more complex models with simpler models, and the parameters of each model are estimated through an Expectation Maximization (EM) procedure. Among the IBM Models, perhaps the most elegant is the HMM model (Vogel et al., 1996). The HMM is the last model whose expectation step is both exact and simple, and it attains a level of accuracy that is very close to the results achieved by much more complex models. In particular, experiments have shown that IBM Models 1, 2, and 3 all perform worse than the HMM and Model 4 benefits greatly from being seeded by the HMM (Och and Ney, 2003).

In this paper we make the following contributions:

- We derive a new alignment model which combines the structure of the HMM and IBM Model 2 and show that its performance is very close to that of the HMM. There are several reasons why such a result would be of value (for more on this, see (Simion et al., 2013) and (Simion et al., 2015a), for example).
- *The main goal of this work is not to eliminate highly non-convex models such as the HMM entirely but, rather, to develop a new, powerful, convex alignment model and thus push the boundary of these theoretically justified techniques further.* Building on the work of (Simion et al., 2015a), we derive a convex relaxation for the new model and show that its performance is close to that of the HMM. Although it does not beat the HMM, the new convex model improves upon the standard IBM Model 2 significantly. Moreover, the convex relaxation also performs better than the strong IBM 2 variant FastAlign (Dyer et al., 2013), IBM Model 3, and the other available convex alignment models detailed in (Simion et al., 2015a) and (Simion et al., 2013).
- We derive a parameter estimation algorithm for

*Currently at Google.
†Currently on leave at Google.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 531–540,
Austin, Texas, November 1-5, 2016.

new model and its convex relaxation based on the EM algorithm. Our model has both HMM emission probabilities and IBM Model 2's distortions, so we can use Model 2 to seed both the model's lexical and distortion parameters. For the convex model, we need not use any initialization heuristics since the EM algorithm we derive is guaranteed to converge to a local optima that is also global.

The goal of our work is to present a model which is convex and has state of the art empirical performance. Although one step of this task was achieved for IBM Model 2 (Simion et al., 2015a), our target goal deals with a much more local-optima-laden, non-convex objective. Finally, whereas IBM 2 in some ways leads to a clear method of attack, we will discuss why the HMM presents challenges that require the insertion of this new surrogate.

Notation. We adopt the notation introduced in (Och and Ney, 2003) of having $1^m 2^n$ denote the training scheme of m IBM Model 1 EM iterations followed by initializing Model 2 with these parameters and running n IBM Model 2 EM iterations. We denote by H the HMM and note that it too can be seeded by running Model 1 followed by Model 2. Additionally, we denote our model as 2_{H}, and note that it has distortion parameters like IBM Model 2 and emission parameters like that of the HMM. Under this notation, we let $1^m 2^n 2^o_{\text{H}}$ denote running Model 1 for m iterations, then Model 2 for n iteration, and then finally our Model for o iterations. As before, we are seeding from the more basic to the more complex model in turn. We denote the convex relaxation of 2_{H} by 2_{HC}. Throughout this paper, for any integer N, we use $[N]$ to denote $\{1 \dots N\}$ and $[N]_0$ to denote $\{0 \dots N\}$. Finally, in our presentation, "convex function" means a function for which a local maxima also global, for example, $f(x) = -x^2$.

2 IBM Models 1 and 2 and the HMM

In this section we give a brief review of IBM Models 1, 2, and the HMM, as well as the the optimization problems arising from these models. The standard approach for optimization within these latent variable models is the EM algorithm.

Throughout this section, and the remainder of the paper, we assume that our set of training examples is $(e^{(k)}, f^{(k)})$ for $k = 1 \dots n$, where $e^{(k)}$ is the k'th English sentence and $f^{(k)}$ is the k'th French sentence. Following standard convention, we assume the task is to translate from *French* (the "source" language) into *English* (the "target" language) [1]. We use E to denote the English vocabulary (set of possible English words), and F to denote the French vocabulary. The k'th English sentence is a sequence of words $e^{(k)}_1 \dots e^{(k)}_{l_k}$ where l_k is the length of the k'th English sentence, and each $e^{(k)}_i \in E$; similarly the k'th French sentence is a sequence $f^{(k)}_1 \dots f^{(k)}_{m_k}$, where m_k is the length of the k'th French sentence, and each $f^{(k)}_j \in F$. We define $e^{(k)}_0$ for $k = 1 \dots n$ to be a special NULL word (note that E contains the NULL word).

For each English word $e \in E$, we will assume that $D(e)$ is a *dictionary* specifying the set of possible French words that can be translations of e. The set $D(e)$ is a subset of F. In practice, $D(e)$ can be derived in various ways; in our experiments we simply define $D(e)$ to include all French words f such that e and f are seen in a translation pair.

Given these definitions, the IBM Model 2 optimization problem is presented in several sources, for example, (Simion et al., 2013). The parameters in this problem are $t(f|e)$ and $d(i|j,l,m)$. The objective function for IBM Model 2 is then the log-likelihood of the training data; we can simplify the log-likelihood (Koehn, 2008) as

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log p(f^{(k)}_j|e^{(k)}) \,,$$

where

$$p(f^{(k)}_j|e^{(k)}) = \sum_{i=0}^{l_k} t(f^{(k)}_j|e^{(k)}_i)d(i|j,l_k,m_k) \,.$$

[1] Technically, in most standard sources (Koehn, 2008), this goes as follows: when we want to translate from *French* to *English* we note that $p(e|f) \propto p(f|e)p(e)$ by Bayes's Theorem. When translating, the alignment models we consider are concerned with modeling $p(f|e)$ while the rest of the translation is handled by the language model $p(e)$. Therefore, in the context of the original task, we have that English is the target language while French is the source. However, for the sake of clarity, we emphasize that the alignment models we study are concerned with the development of $p(f|e)$.

This last simplification is crucial as it allows for a simple multinomial EM implementation, and can be done for IBM Model 1 as well (Koehn, 2008). Furthermore, the ability to write out the marginal likelihood per sentence in this manner has seen other applications: it was crucial, for example, in deriving a convex relaxation of IBM Model 2 and solving the new problem using subgradient methods (Simion et al., 2013).

An improvement on IBM Model 2, called the HMM alignment model, was introduced by Vogel et al (Vogel et al., 1996). For this model, the distortion parameters are replaced by *emission* parameters $d(a_j|a_{j-1}, l)$. These emission parameters specify the probability of the next alignment variable for the j^{th} target word is a_j, given that the previous source word was aligned to a target word whose position was a_{j-1} in a target sentence with length of l. The objective of the HMM is given by

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{a_1^{(k)}\ldots a_{m_k}^{(k)}}\log\prod_{j=1}^{m_k} t(f_j^{(k)}|e_{a_j^k}^{(k)})d(a_j^{(k)}|a_{j-1}^{(k)}, l_k)$$

and we present this in Fig 1. We note that unlike IBM Model 2, we cannot simplify the exponential sum within the log-likelihood of the HMM, and so EM training for this model requires the use of a special EM implementation knows as the Baum-Welch algorithm (Rabiner and Juang., 1986).

Once these models are trained, each model's highest probability (Viterbi) alignment is computed. For IBM Models 1 and 2, the Viterbi alignment splits easily (Koehn, 2008). For the HMM, dynamic programming is used (Vogel et al., 1996). Although it is non-convex and thus its initialization is important, the HMM is the last alignment model in the classical setting that has an exact EM procedure (Och and Ney, 2003): from IBM Model 3 onwards heuristics are used within the expectation and maximization steps of each model's associated EM procedure.

3 Distortion and emission parameter structure

The structure of IBM Model 2's distortion parameters and the HMM's emission parameters is important and is used in our model as well, so we detail this here. We are using the roughly same structure as (Liang et al., 2006) and (Dyer et al., 2013): the distortions and emissions of our model are parametrized by forcing the model to concentrate its alignments on the diagonal.

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|i, l_k)$ for each $i \in [l_k]_0$, $i' \in [l_k]_0$.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \quad (1)$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \quad (2)$$

$$\forall i, i' \in [l_k]_0, d(i'|i, l_k) \geq 0 \quad (3)$$

$$\forall i \in [l_k]_0, \sum_{i' \in [l_k]_0} d(i'|i, l_k) = 1 \quad (4)$$

Objective: Maximize

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{a_1^{(k)}\ldots a_{m_k}^{(k)}}\log\prod_{j=1}^{m_k} t(f_j^{(k)}|e_{a_j^k}^{(k)})d(a_j^{(k)}|a_{j-1}^{(k)}, l_k)$$

with respect to the $t(f|e)$ parameters $d(i'|i, l)$.

Figure 1: The HMM Optimization Problem

3.1 Distortion Parameters for IBM2

Let $\lambda > 0$. For the IBM Model 2 distortions we set the NULL word probability as $d(0|j, l, m) = p_0$, where $p_0 = \frac{1}{l+1}$ and note that this will generally depend on the target sentence length within a bitext training pair that we are considering. For $i \neq 0$ which satisfies we set

$$d(i|j, l, m) = \frac{(1 - p_0)e^{-\lambda|\frac{i}{l} - \frac{j}{m}|}}{Z_\lambda(j, l, m)},$$

where $Z_\lambda(j, l, m)$ is a normalization constant as in (Dyer et al., 2013).

3.2 Emission Parameters for HMM

Let $\theta > 0$. For the HMM emissions we first set the NULL word generation to $d(0|i, l) = p_0$, with

$p_0 = \frac{1}{l+1}$. For target word position $i, i' \neq 0$, we set

$$d(i'|i,l) = \frac{(1-p_0)e^{-\theta|\frac{i'}{l}-\frac{i}{l}|}}{Z_\theta(i,l,m)} ,$$

where $Z_\theta(i,l,m)$ is a suitable normalization constant. Lastly, if $i = 0$ so that we are jumping from the NULL word onto a possibly different word, we set $d(i'|0,l) = p_0$. Aside from making the NULL word have uniform jump probability, the above emission parameters are modeled to favor a jumping to an adjacent English word.

4 Combining IBM Model 2 and the HMM

In deriving the new HMM surrogate, our main goal was to allow the current alignment to know as much as possible about the previous alignment variable and still have a likelihood that factors as that of IBM Model 2 (Simion et al., 2013; Koehn, 2008). We combine IBM Model 2 and the HMM by incorporating the generation of words using the structure of both models. The model we introduce, IBM2-HMM, is displayed in Fig 2.

Consider a target-source sentence pair (e, f) with $|e| = l$ and $|f| = m$. For source sentence positions j and $j+1$ we have source words f_j and f_{j+1} and we assign a joint probability involving the alignments a_j and a_{j+1} as:

$$q(j, a_j, a_{j+1}, l, m) \quad = \tag{5}$$
$$t(f_j|e_{a_j})d(a_j|j,l,m)t(f_{j+1}|e_{a_{j+1}})d(a_{j+1}|a_j,l) . \tag{6}$$

From the equation above, we note that we use the IBM Model 2's word generation method for position j and the HMM generative structure for position $j+1$. The generative nature of the above procedure introduces dependency between adjacent words two at a time. Since we want to mimic the HMM's structure as much as possible, we devise our likelihood function to mimic the HMM's dependency between alignments using q. Essentially, we move the source word position j from 1 to m and allow for overlapping terms when $j \in \{2, \ldots, m-1\}$. In what follows, we describe this representation in detail.

The likelihood in Eq. 16 is actually the sum of two likelihoods which use equations Eq. 5 and 6 repeatedly. To this end, we will discuss how our objective is actually

$$\frac{1}{n}\sum_{k=1}^{n} \log \sum_{a^{(k)}, b^{(k)}} p(f^{(k)}, a^{(k)}, b^{(k)}|e^{(k)}) , \tag{7}$$

where $a^{(k)}$ and $b^{(k)}$ both are alignment vectors whose components are independent and can take on any values in $[l_k]_0$. To see how $p(f, a, b|e)$ comes about, note that we could generate the sentence f by generating pairs $(1,2), (3,4), (5,6), \ldots$ using equations Eqs. 5 and 6 for each pair. Taking all this together, the upshot of our discussion is that generating the pair (e, f) in this way gives us that the likelihood for an alignment a would be given by:

$$p_1(f, a|e) = \prod_{j \text{ odd}}^{m-1} q(j, a_j, a_{j+1}, l, m) . \tag{8}$$

Using the same idea as above, we could also skip the first target word position and generate pairs $(2,3), (4,5), \ldots$ using Eqs. 5 and 6. Under this second generative method, the joint probability for f and alignment b is:

$$p_2(f, b|e) = \prod_{j \text{ even}}^{m-1} q(j, b_j, b_{j+1}, l, m) , \tag{9}$$

Finally, we note that if m is even we do not generate f_1 and f_m under p_2 but we do generate these words under p_1. Similarly, if m is odd we do not generate f_1 under p_2 and we do not generate f_m under p_1; however in this case as in the first, we still generate these missing words under the other generative method. Using $p(f, a, b|e) = p_1(f, a|e)p_2(f, b|e)$ and factoring the log-likelihood as in IBM Model 1 and 2 (Koehn, 2008), we get the log-likelihood in Fig 2. Finally, we note that our model's log-likelihood could be viewed as the sum of the log-likelihoods of a model which generates (e, f) using p_1 and another model which generates sentences using p_2. These models share parameters but generate words using different recipes, as discussed above.

5 The parameter estimation for IBM2-HMM

To fully optimize our new model (over t, λ, and θ), we can use an EM algorithm in the same fashion as

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $\lambda > 0$ for distortion centering.
- A parameter $\theta > 0$ for emission centering.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \quad (10)$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \quad (11)$$

$$\forall i \in [l_k]_0, j \in [m_k], d(i|j, l_k, m_k) \geq 0 \quad (12)$$

$$\forall j \in [m_k], \sum_{i \in [l_k]_0} d(i|j, l_k, m_k) = 1 \quad (13)$$

$$\forall i, i' \in [l_k]_0, d(i'|i, l_k) \geq 0 \quad (14)$$

$$\forall i \in [l_k]_0, \sum_{i' \in [l_k]_0} d(i'|i, l_k) = 1 \quad (15)$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k - 1} \log \sum_{i=0}^{l_k} \sum_{i'=0}^{l_k} q(j, i, i', l_k, m_k) \quad (16)$$

with respect to the parameters $t(f|e)$, $d(i'|i, l)$ $d(i|j, l, m)$, and $q(j, i, i', l_k, m_k)$ set as

$$t(f_j^{(k)}|e_i^{(k)}) d(i|j, l, m) t(f_{j+1}^{(k)}|e_{i'}) d(i'|i, l) \quad (17)$$

Figure 2: The IBM2-HMM Optimization Problem. We use equation (5) within the likelihood definition.

(Dyer et al., 2013). Specifically, for the model in question the EM algorithm still applies but we have to use a gradient-based algorithm within the learning step. On the other hand, since such a gradient-based method introduces the necessary complication of a learning rate, we could also optimize the objective by picking θ and λ via cross-validation and using a multinomial EM algorithm for the learning of the lexical t terms. For our experiments, we opted for this simpler choice: we derived a multinomial EM algorithm and cross-validated the centering parameters for the distortion and emission terms. With λ and θ fixed, the derivation of this algorithm is very similar to the one used for IBM2-HMM's convex relaxation and this uses the path discussed in (Simion et al., 2015a) and (Simion et al., 2015b). We detail the EM algorithm for the convex relaxation below.

6 A Convex HMM Surrogate

In this section we detail a procedure to get a convex relaxation for IBM2-HMM. Let $(\mathbf{t}, \mathbf{d})$ be all the parameters of the HMM. As a first step in getting a convex HMM, one could follow the path developed in (Simion et al., 2015a) and directly replace the HMM's objective terms

$$\prod_{j=1}^{m_k} t(f_j^{(k)}|e_{a_j^k}^{(k)}) d(a_j^{(k)}|a_{j-1}^{(k)}, l_k)$$

by

$$\left(\prod_{j=1}^{m_k} t(f_j^{(k)}|e_{a_j^k}^{(k)}) d(a_j^{(k)}|a_{j-1}^{(k)}, l_k)\right)^{\frac{1}{2m_k}} .$$

In particular, the geometric mean function $h(x_1, \ldots, h_{2m_k}) = (\prod_{j=1}^{2m_k} x_j)^{\frac{1}{2m_k}}$ is convex ((Boyd and Vandenberghe, 2004)) and, for a given sentence pair $(e^{(k)}, f^{(k)})$ with alignment $a^{(k)}$ we can find a projection matrix $\mathbf{P}$ so that $\mathbf{P}(\mathbf{t}, \mathbf{d}) = (\tilde{\mathbf{t}}, \tilde{\mathbf{d}})$ where $\tilde{\mathbf{t}} = \{t(f_j^{(k)}|e_{a_j^k}^{(k)})\}_{j=1}^{m_k}$ and $\tilde{\mathbf{d}} = \{d(a_j^{(k)}|a_{j-1}^{(k)}, l_k)\}_{j=1}^{m_k}$ are exactly the parameters used in the term above (in particular, $\mathbf{t}, \mathbf{d}$ are the set of all parameters while $\tilde{\mathbf{t}}, \tilde{\mathbf{d}}$ are the set of parameters for the specific training pair k; $\mathbf{P}$ projects from the full space onto only the parameters used for training pair k). Given this, we then have that $g(\mathbf{t}, \mathbf{d}) = h(\mathbf{P}(\mathbf{t}, \mathbf{d})) = h(\tilde{\mathbf{t}}, \tilde{\mathbf{d}})$ is convex and, by composition, so is $\log g(\mathbf{t}, \mathbf{d})$ (see (Simion et al., 2015a; Boyd and Vandenberghe, 2004) for details; the main idea lies in the fact that as linear transformations preserve convexity, so do compositions of convex functions with increasing convex functions such as $\log$). Finally, if we run this plan for all terms in the objective, the new objective is convex since it is the sum of convex functions (the new optimization problem is convex as it has linear constraints). Although this gives a convex program, we observed that the powers being so small made the optimized probabilities very uninformative (i.e. uniform). The above makes sense: no matter what the parameters are, we will easily get the 1 we seek

for each term in the objective since all terms are taken to a low ($\frac{1}{2m_k}$) power .

Since this direct relaxation does not yield fruit, we next could turn to our model. Developing its relaxation in the vein of (Simion et al., 2015a), we could be to let $d(i|j,l,m)$ and $d(i'|i,l)$ be multinomial probabilities (that is, these parameters would not have centering parameters λ and θ and would be just standard probabilities as in the GIZA++ versions of the HMM and IBM Model 2 (Och and Ney, 2003)) and replace all the terms $q(j,i',i,l,m)$ in (16) by $(q(j,i',i,l,m))^{\frac{1}{4}}$. Although this method is feasible, experiments showed that the relaxation is not very competitive and performs on par with IBM Model 2; this relaxation is far in performance from the HMM even though we are relaxing (only) the product of 4 terms (lastly, we mention that we tried other variants were we replaced $d(i|j,l,m)d(i'|i,l)$ by $d(i,i'|j,l,m)$ so that we would have only three terms; unfortunately, this last attempt also produced parameters that were "too uniform").

The above analysis motivates why we defined our model as we did: we now have only two terms to relax. In particular, to rectify the above, we left in place the structure discussed in Section 3 and made λ and θ be tuning parameters which we can cross-validate for on a small held-out data set. This last constraint effectively removes the distortion and emission parameters from the model but we still maintain the structural property of these parameters: we maintain their favoring the diagonal or adjacent alignment. To get the relaxation, we replace $q(j,i,i',l,m)$ by

$$p(j,i,i',l,m) \propto \sqrt{t(f_j^{(k)}|e_i^{(k)})t(f_{j+1}^{(k)}|e_{i'})}$$

and set the proportionality constant to be $d(i|j,l,m)d(i'|i,l)$. Using this setup we now have a convex objective to optimize over. In particular, we've formulated a convex relaxation of the IBM2-HMM problem which, like the Support Vector Machine, includes parameters that can be cross-validated over (Boyd and Vandenberghe, 2004).

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2. Pick $\lambda, \theta > 0$ as in Section 3 via cross-validation.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \quad (18)$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \quad (19)$$

Objective: Maximize

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k-1}\log\sum_{i=0}^{l_k}\sum_{i'=0}^{l_k} p(j,i,i',l_k,m_k) \quad (20)$$

with respect to the parameters $t(f|e)$ and $p(j,i,i',l_k,m_k)$ set as

$$\sqrt{t(f_j^{(k)}|e_i^{(k)})}d(i|j,l,m)\sqrt{t(f_{j+1}^{(k)}|e_{i'})}d(i'|i,l)$$

Figure 3: The IBM2-HMM convex relaxation optimization problem. Note that the distortions $d(i|j,l,,m)$ and emissions $d(i'|i,l)$ are constants held fixed and parameterized by cross-validated parameters λ and θ as in Section 3.

7 An EM algorithm for the convex surrogate

The EM algorithm for the convex relaxation of our surrogate is given in Fig 4. As the model's objective is the sum of the objectives of two models generated by a multinomial rule, we can get a very succinct EM algorithm. For more details on this and a similar derivation, please refer to (Simion et al., 2015a), (Koehn, 2008) or (Simion et al., 2015b). For this algorithm, we again note that the distortion and emission parameters are constants so that the only estimation that needs to be conducted is on the lexical t terms.

To be specific, we have that the M step requires optimizing

$$\frac{1}{n}\sum_{k=1}^{n}\log\sum_{a^{(k)},b^{(k)}} q(a^{(k)},b^{(k)}|e^{(k)},f^{(k)})p(f^{(k)},a^{(k)},b^{(k)}|e^{(k)}).$$

In the above, we have that

$$q(a^{(k)}, b^{(k)} | e^{(k)}, f^{(k)})$$

are constants proportional to

$$\prod_{j=1}^{m_k - 1} \sqrt{t(f_j^{(k)} | e^{(k)}_{a_j^{(k)}}) t(f_{j+1}^{(k)} | e^{(k)}_{a_{j+1}^{(k)}})} \prod_{j=2}^{m_k} \sqrt{t(f_j^{(k)} | e^{(k)}_{b_j^{(k)}}) t(f_{j+1}^{(k)} | e^{(k)}_{b_{j+1}^{(k)}})}$$

and gotten through the E step. This optimization step is very similar to the regular Model 2 M step since the β drops down using $\log t^\beta = \beta \log t$; the exact same count-based method can be applied. The upshot of this is given in Fig 4; similar to the logic above for 2_{HC}, we can get the EM algorithm for 2_{H}.

8 Decoding methods for IBM2-HMM

When computing the optimal alignment we wanted to compare our model with the HMM as closely as possible. Because of this, the most natural method of evaluating the quality of the parameters would be to use the same rule as the HMM. Specifically, for a sentence pair (e, f) with $|e| = l$ and $|f| = m$, in HMM decoding we aim to find $(a_1 \dots a_m)$ which maximizes

$$\max_{a_1,\dots,a_m} \prod_{j=1}^{m} t(f_j | e_{a_j}) d(a_j | a_{j-1}, l).$$

As is standard, dynamic programming can now be used to find the Viterbi alignment. Although there are a number of ways we could define the optimal alignment, we felt that the above would be the best since it tests dependance between alignment variables and allows for easy comparison with the GIZA++ HMM. Finding the optimal alignment under the HMM setting is labelled "HMM" in Table 1.

We can also find the optimal alignment by taking the objective literally (see (Simion et al., 2014) for a similar argument dealing with the convex relaxation of IBM Model 2) and computing

$$\max_{a_1 \dots a_m} p_1(f, a|e) p_2(f, a|e).$$

Above, we are asking for the optimal alignment that yields the highest probability alignment through generating technique p_1 and p_2. This method of decoding is a lot like the HMM style and also relies on dynamic programming. In this case we have the recursion for Q_{Joint} given by

$$Q_{Joint}(1, i) = t(f_1 | e_i) d^2(i | 1, l, m) \ ,$$

$\forall i \in [l]_0$, and

$$Q_{Joint}(j, i') = t^2(f_j | e_{i'}) d(i' | j, l, m) M_{Joint}(j-1, i') \ ,$$

where $M_{Joint}(j-1, i')$ is

$$M_{Joint}(j-1, i') = \max_{i=0}^{l} \{ d(i' | i, l) Q_{Joint}(j-1, i) \} \ ,$$

$\forall\, 2 \le j \le m, \forall\, i' \in [l]_0$. The alignment results gotten by decoding with this method is labelled "Joint" in Table 1.

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2. Two parameters $\lambda, \theta > 0$ picked by cross-validation so that the distortions and emissions are constants obeying the structure in Section 3. An integer T specifying the number of passes over the data.
Parameters:
• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
Initialization:
• $\forall e \in E,\ f \in D(e)$, set $t(f|e) = \frac{1}{D(e)}$.
EM Algorithm: Expectation
for all $k = 1 \dots N$ **do**
  for all $j = 1 \dots m_k$ **do**
    $\delta = 0$
    $\Delta = 0$
    for all $i = 0 \dots l_k$ **do**
      for all $i' = 0 \dots l_k$ **do**
        $\delta[i, i'] = p(j, i', i, l_k, m_k)$
        $\Delta += \delta[i, i']$
    for all $i = 0 \dots l_k$ **do**
      for all $i' = 0 \dots l_k$ **do**
        $\delta[i, i'] = \frac{\delta[i, i']}{\Delta}$
        $counts(f_j^{(k)}, e_i^{(k)}) += \delta[i, i']$
        $counts(e_i^{(k)}) += \delta[i, i']$
        $counts(f_{j+1}^{(k)}, e_{i'}^{(k)}) += \delta[i, i']$
        $counts(e_{i'}^{(k)}) += \delta[i, i']$
EM Algorithm: Maximization
for all $e \in E$ **do**
  for all $f \in D(e)$ **do**
    $t(f|e) = \frac{counts(e,f)}{counts(e)}$
Output: t parameters.

Figure 4: Pseudocode for the EM algorithm of the IBM2-HMM's convex relaxation. As the distortion and emission parameters are constants, the algorithm is very similar to that of IBM Model 1.

9 Experiments

In this section we describe experiments using the IBM2-HMM optimization problem combined with the EM algorithm for parameter estimation.

9.1 Data Sets

We use data from the bilingual word alignment workshop held at HLT-NAACL 2003 (Michalcea and Pederson, 2003). We use the Canadian Hansards bilingual corpus, with 743,989 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (note that we use a randomly chosen subset of the original training set of 1.1 million sentences, similar to the setting used in (Moore, 2004)). The development and test data have been manually aligned at the word level, annotating alignments between source and target words in the corpus as either "sure" (S) or "possible" (P) alignments, as described in (Och and Ney, 2003). As is standard, we lower-cased all words before giving the data to GIZA++ and we ignored NULL word alignments in our computation of alignment quality scores.

9.2 Methodology

We test several models in our experiments. In particular, we empirically evaluate our models against the GIZA++ IBM Model 3 and HMM, as well as the FastAlign IBM Model 2 implementation of (Dyer et al., 2013) that uses Variational Bayes. For each of our models, we estimated parameters and got alignments in turn using models in the source-target and target-source directions; using the same setup as (Simion et al., 2013), we present the gotten intersected alignments. In training, we employ the standard practice of initializing non-convex alignment models with simpler non-convex models. In particular, we initialize, the GIZA++ HMM with IBM Model 2, IBM Model 2 with IBM Model 1, and IBM2-HMM and IBM Model 3 with IBM Model 2 preceded by Model 1. Lastly, for FastAlign, we initialized all parameters uniformly since this empirically was a more favorable initialization, as discussed in (Dyer et al., 2013).

We measure the performance of the models in terms of *Precision*, *Recall*, *F-Measure*, and *AER* using only sure alignments in the definitions of the first three metrics and sure and possible alignments in the definition of *AER* , as in (Simion et al., 2013) and (Marcu et al., 2006). For our experiments, we report results in both AER (lower is better) and F-Measure (higher is better) (Och and Ney, 2003).

Table 1 shows the alignment summary statistics for the 447 sentences present in the Hansard test data. We present alignments quality scores using either the FastAlign IBM Model 2, the GIZA++ HMM, and our model and its relaxation using either the "HMM" or "Joint" decoding. First, we note that in deciding the decoding style for IBM2-HMM, the HMM method is better than the Joint method. We expected this type of performance since HMM decoding introduces positional dependance among the entire set of words in the sentence, which is shown to be a good modeling assumption (Vogel et al., 1996).

From the results in Table 1 we see that the HMM outperforms all other models, including IBM2-HMM and its convex relaxation. However, IBM2-HMM is not far in AER performance from the HMM and both it and its relaxation do better than FastAlign or IBM Model 3 (the results for IBM Model 3 are not presented; a one-directional English-French run of $1^5 2^5 3^{15}$ gave AER and F-Measure numbers of 0.1768 and 0.6588, respectively, and this was behind both the IBM Model 2 FastAlign and our models).

As a further set of experiments, we also appended an IBM Model 1 or IBM Model 2 objective to our models's original objectives, so that the constraints and parameters are the same but now we are maximizing the average of two log-likelihoods. With regard to the EM optimization, we would only need to add another δ parameter: we'd now have probabilities $\delta_1[i] \propto t(f_j^{(k)}|e_i^{(k)})d(i|j,l_l,m_k)$ (this is for IBM Model 2 smoothing; we have $d = 1$ for IBM 1 smoothing) and $\delta_2[i,i'] \propto p(j,i,i'.l_k,m_k)$ in the EM Algorithm that results (for more, see (Simion et al., 2015a)). We note that the appended IBM Model 2 objective is still convex if we fix the distortions' λ parameter and then optimize for the t parameters via EM (thus, model 2_{HC} is still convex). For us, there were significant gains, especially in the convex model. The results for all these experiments are shown in Table 2, with IBM 2 smoothing for the convex model displayed in the rightmost column.

Finally, we also tested our model in the full

Training	$1^5 2^{10}_{\text{H}}$	$1^5 2^{10}_{\text{H}}$	2^{10}_{HC}	2^{10}_{HC}	FA^{10}	$1^5 2^5 \text{H}^{10}$
Decoding	HMM	Joint	HMM	Joint	IBM2	HMM
Iteration			AER			
1	0.0956	0.1076	0.1538	0.1814	0.5406	0.1761
2	0.0884	0.0943	0.1093	0.1343	0.1625	0.0873
3	0.0844	0.0916	0.1023	0.1234	0.1254	0.0786
4	0.0828	0.0904	0.0996	0.1204	0.1169	0.0753
5	0.0808	0.0907	0.0992	0.1197	0.1131	0.0737
6	0.0804	0.0906	0.0989	0.1199	0.1128	0.0719
7	0.0795	0.0910	0.0986	0.1197	0.1116	0.0717
8	0.0789	0.0900	0.0988	0.1195	0.1086	0.0725
9	0.0793	0.0904	0.0986	0.1195	0.1076	0.0738
10	0.0793	0.0902	0.0986	0.1195	0.1072	0.0734
Iteration			F-Measure			
1	0.7829	0.7797	0.7199	0.6914	0.2951	0.7219
2	0.7854	0.7805	0.7594	0.7330	0.7111	0.8039
3	0.7899	0.7806	0.7651	0.7427	0.7484	0.8112
4	0.7908	0.7813	0.7668	0.7457	0.7589	0.8094
5	0.7928	0.7806	0.7673	0.7461	0.7624	0.8058
6	0.7928	0.7807	0.7678	0.7457	0.7630	0.8056
7	0.7939	0.7817	0.7679	0.7457	0.7633	0.8046
8	0.7942	0.7814	0.7679	0.7458	0.7658	0.8024
9	0.7937	0.7813	0.7680	0.7457	0.7672	0.8007
10	0.7927	0.7816	0.7680	0.7457	0.7679	0.8010

Table 1: Alignment quality results for IBM2-HMM (2_{H}) and its convex relaxation (2_{HC}) using either HMM-style dynamic programming or "Joint" decoding. The first and last columns above are for the GIZA++ HMM initialized either with IBM Model 1 or Model 1 followed by Model 2. FA above refers to the improved IBM Model 2 (FastAlign) of (Dyer et al., 2013).

Training	$1^5 2^{10}_{\text{H}}$	$1^5 2^{10}_{\text{H}}$	2^{10}_{HC}	2^{10}_{HC}
Smoothing	IBM1	IBM2	IBM1	IBM2
Decoding	HMM	HMM	HMM	HMM
Iteration		AER		
1	0.1003	0.0958	0.1703	0.1482
2	0.0949	0.0890	0.1172	0.1057
3	0.0904	0.0840	0.1039	0.0955
4	0.0886	0.0816	0.0984	0.0927
5	0.0866	0.0795	0.0948	0.0894
6	0.0851	0.0794	0.0933	0.0888
7	0.0837	0.0790	0.0922	0.0886
8	0.0825	0.0788	0.0921	0.0880
9	0.0820	0.0785	0.0921	0.0881
10	0.0820	0.0777	0.0920	0.0881
Iteration		F-Measure		
1	0.7791	0.7817	0.7065	0.7251
2	0.7822	0.7839	0.7559	0.7637
3	0.7856	0.7897	0.7689	0.7740
4	0.7873	0.7923	0.7729	0.7760
5	0.7899	0.7938	0.7771	0.7782
6	0.7904	0.7943	0.7789	0.7788
7	0.7917	0.7946	0.7800	0.7791
8	0.7928	0.7944	0.7806	0.7795
9	0.7930	0.7941	0.7806	0.7797
10	0.7925	0.7947	0.7806	0.7796

Table 2: Alignment quality results for IBM2-HMM and its relaxation using IBM 1 and IBM 2 smoothing (in this case, "smoothing" means adding these log-likelihoods to the original objective as in (Simion et al., 2013). For the convex relaxation of IBM2-HMM, we can only smooth by adding in the convex IBM Model 1 objective, or by adding in an IBM Model 2 objective where the distortions are taken to be constants (these distortions are identical to the ones that are used within the relaxation itself and are cross-validated for optimal λ).

SMT pipeline using the cdec system (Dyer et al., 2013). For our experiments, we compared our models' alignments (gotten by training $1^5 2^5_{\text{H}}$ and 2^5_{HC}) against the alignments gotten by the HMM ($1^5 2^5 H^5$), IBM Model 4 ($1^5 H^5 3^3 4^3$), and FastAlign. Unfortunately, we found that all 4 systems led to roughly the same BLEU score of 40 on a Spanish-English training set of size 250000 which was a subset of version 7 of the Europarl dataset (Dyer et al., 2013). For our development and test sets, we used data each of size roughly 1800 and we preprocessed all data by considering only sentences of size less than 80 and filtering out sentences which had a very large (or small) ratio of target and source sentence lengths. Although the SMT results were not a success in that our gains were not significant, we felt that the experiments at least highlight that our model mimics the HMM's alignments even though its structure is much more local. Lastly, we in regards to the new convex model's performance, we observe much better alignment quality than any other convex alignment models in print, for example, (Simion et al., 2015a).

10 Conclusions and Future Work

Our work has explored some of the details of a new model which combines the structure of IBM Model 2 the alignment HMM Model. We've shown that this new model and its convex relaxation performs very close to the standard GIZA++ implementation of the HMM. Bridging the gap between the HMM and convex models proves difficult for a number of reasons (Guo and Schuurmans, 2007). In this paper, we have introduced a new set of ideas aimed at tightening this gap.

Acknowledgments

Andrei Simion was supported by a Google research award. Cliff Stein was partially supported by NSF grant CCF-1421161. We thank the reviewers for their insightful commentary and suggestions.

References

Steven Boyd and Lieven Vandenberghe. 2004. Convex Optimization. Cambridge University Press.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263-311.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the royal statistical society, series B*, 39(1):1-38.

Chris Dyer , Victor Chahuneau, Noah A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. *In Proceedings of NAACL.*

Alexander Fraser and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Journal Computational Linguistics*, 33(3): 293-303.

Joao V. Graca, Kuzman Ganchev and Ben Taskar. 2007. Expectation Maximization and Posterior Constraints. *In Proceedings of NIPS.*

Yuhong Guo and Dale Schuurmans. 2007. Convex Relaxations of Latent Variable Training. *In Proceedings of NIPS*.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael Jordan. 2008. Word Alignment via Quadratic Assignment. *In Proceedings of the HLT-NAACL.*

Phillip Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. *In Proceedings of the EMNLP*.

Phillip Koehn. 2008. Statistical Machine Translation. Cambridge University Press.

Percy Liang, Ben Taskar and Dan Klein. 2006. Alignment by Agreement. *In Proceedings of NAACL.*

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. *In Proceedings of the EMNLP*.

Rada Michalcea and Ted Pederson. 2003. An Evaluation Exercise in Word Alignment. *HLT-NAACL 2003: Workshop in building and using Parallel Texts: Data Driven Machine Translation and Beyond*.

Robert C. Moore. 2004. Improving IBM Word-Alignment Model 1. *In Proceedings of the ACL.*

Stephan Vogel, Hermann Ney and Christoph Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. *In Proceedings of COLING.*

Franz Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational-Linguistics*, 29(1): 19-52.

L.R. Rabiner and B.H. Juang. 1986. An Introduction to Hidden Markov Models. *In IEEE ASSP Magazine*.

Andrei Simion, Michael Collins and Cliff Stein. 2013. A Convex Alternative to IBM Model 2. *In Proceedings of EMNLP*.

Andrei Simion, Michael Collins and Cliff Stein. 2013. Some Experiments with a Convex IBM Model 2. *In Proceedings of EACL.*

Andrei Simion, Michael Collins and Cliff Stein. 2015. A Family of Latent Variable Convex Relaxations for IBM Model 2. *In Proceedings of the AAAI*.

Andrei Simion, Michael Collins and Cliff Stein. 2015. On a Strictly Concave IBM Model 1. *In Proceedings of EMNLP*.

Kristina Toutanova and Michel Galley. 2011. Why Initialization Matters for IBM Model 1: Multiple Optima and Non-Strict Convexity. *In Proceedings of the ACL.*

Ashish Vaswani, Liang Huang and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the $L0$-norm. *In Proceedings of the ACL.*

Solving Verbal Questions in IQ Test by Knowledge-Powered Word Embedding

Huazheng Wang
University of Virginia
hw7ww@virginia.edu

Fei Tian
Microsoft Research
fetia@microsoft.com

Bin Gao
Microsoft
bingao@microsoft.com

Chengjieren Zhu
University of California, San Diego
chz191@ucsd.edu

Jiang Bian
Yidian Inc.
jiang.bian.prc@gmail.com

Tie-Yan Liu
Microsoft Research
tyliu@microsoft.com

Abstract

Verbal comprehension questions appear very frequently in Intelligence Quotient (IQ) tests, which measure human's verbal ability including the understanding of the words with multiple senses, the synonyms and antonyms, and the analogies among words. In this work, we explore whether such tests can be solved automatically by the deep learning technologies for text data. We found that the task was quite challenging, and simply applying existing technologies like word embedding could not achieve a good performance, due to the multiple senses of words and the complex relations among words. To tackle these challenges, we propose a novel framework to automatically solve the verbal IQ questions by leveraging improved word embedding by jointly considering the multi-sense nature of words and the relational information among words. Experimental results have shown that the proposed framework can not only outperform existing methods for solving verbal comprehension questions but also exceed the average performance of the Amazon Mechanical Turk workers involved in the study.

1 Introduction

The Intelligence Quotient (IQ) test (Stern, 1914) is a test of intelligence designed to formally study the success of an individual in adapting to a specific situation under certain conditions. Common IQ tests measure various types of abilities such as verbal, mathematical, logical, and reasoning skills. These tests have been widely used in the study of psychology, education, and career development. In the community of artificial intelligence, agents have been invented to fulfill many interesting and challenging tasks like face recognition, speech recognition, handwriting recognition, and question answering. However, as far as we know, there are very limited studies of developing an agent to solve IQ tests, which in some sense is more challenging, since even common human beings do not always succeed on such tests. Considering that IQ test scores have been widely considered as a measure of *intelligence*, we think it is worth further investigating whether we can develop an agent that can solve IQ test questions.

The commonly used IQ tests contain several types of questions like verbal, mathematical, logical, and picture questions, among which a large proportion (near 40%) are verbal questions (Carter, 2005). The recent progress on deep learning for natural language processing (NLP), such as word embedding technologies, has advanced the ability of machines (or AI agents) to understand the meaning of words and the relations among words. This inspires us to solve the verbal questions in IQ tests by leveraging the word embedding technologies. However, our attempts show that a straightforward application of word embedding does not result in satisfactory performances. This is actually understandable. Standard word embedding technologies learn one embedding vector for each word based on the co-occurrence information in a text corpus. However, verbal comprehension questions in IQ tests usually consider the multiple senses of a word (and often focus on the rare senses), and the complex relations among (polysemous) words. This has clearly exceeded the capability of standard word embedding

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 541–550,
Austin, Texas, November 1-5, 2016.

technologies.

To tackle the aforementioned challenges, we propose a novel framework that consists of three components.

First, we build a classifier to recognize the specific type (e.g., analogy, classification, synonym, and antonym) of verbal questions. For different types of questions, different kinds of relationships need to be considered and the solvers could have different forms. Therefore, with an effective question type classifier, we may solve the questions in a divide-and-conquer manner.

Second, we obtain distributed representations of words and relations by leveraging a novel word embedding method that considers the multi-sense nature of words and the relational knowledge among words (or their senses) contained in dictionaries. In particular, for each polysemous word, we retrieve its number of senses from a dictionary, and conduct clustering on all its context windows in the corpus. Then we attach the example sentences for every sense in the dictionary to the clusters, such that we can tag the polysemous word in each context window with a specific word sense. On top of this, instead of learning one embedding vector for each *word*, we learn one vector for each *pair of word-sense*. Furthermore, in addition to learning the embedding vectors for words, we also learn the embedding vectors for relations (e.g., synonym and antonym) at the same time, by incorporating relational knowledge into the objective function of the word embedding learning algorithm. That is, the learning of word-sense representations and relation representations interacts with each other, such that the relational knowledge obtained from dictionaries is effectively incorporated.

Third, for each type of question, we propose a specific solver based on the obtained distributed word-sense representations and relation representations. For example, for analogy questions, we find the answer by minimizing the distance between word-sense pairs in the question and the word-sense pairs in the candidate answers.

We have conducted experiments using a combined IQ test set to test the performance of our proposed framework. The experimental results show that our method can outperform several baseline methods for verbal comprehension questions on IQ tests. We further deliver the questions in the test set to human beings through Amazon Mechanical Turk[1]. The average performance of the human beings is even a little lower than that of our proposed method.

2 Related Work

2.1 Verbal Questions in IQ Test

In common IQ tests, a large proportion of questions are verbal comprehension questions, which play an important role in deciding the final IQ scores. For example, in Wechsler Adult Intelligence Scale (Wechsler, 2008), which is among the most famous IQ test systems, the full-scale IQ is calculated from two IQ scores: Verbal IQ and Performance IQ, and around 40% of questions in a typical test are verbal comprehension questions. Verbal questions can test not only the verbal ability (e.g., understanding polysemy of a word), but also the reasoning ability and induction ability of an individual. According to previous studies (Carter, 2005), verbal questions mainly have the types elaborated in Table 1, in which the correct answers are highlighted in bold font.

Analogy-I questions usually take the form "A is to B as C is to ?". One needs to choose a word D from a given list of candidate words to form an analogical relation between pair (A, B) and pair (C, D). Such questions test the ability of identifying an implicit relation from word pair (A, B) and apply it to compose word pair (C, D). Note that the Analogy-I questions are also used as a major evaluation task in the *word2vec* models (Mikolov et al., 2013). Analogy-II questions require two words to be identified from two given lists in order to form an analogical relation like "A is to ? as C is to ?". Such questions are a bit more difficult than the Analogy-I questions since the analogical relation cannot be observed directly from the questions, but need to be searched for in the word pair combinations from the candidate answers. Classification questions require one to identify the word that is different (or dissimilar) from others in a given word list. Such questions are also known as *odd-one-out*, which have been studied in (Pintér et al., 2012). Classification questions test the ability to summarize the majority

[1] http://www.mturk.com/

Type	Example
Analogy-I	Isotherm is to temperature as isobar is to? (i) atmosphere, (ii) wind, (iii) **pressure**, (iv) latitude, (v) current.
Analogy-II	Identify two words (one from each set of brackets) that form a connection (analogy) when paired with the words in capitals: CHAPTER (**book**, verse, read), ACT (stage, audience, **play**).
Classification	Which is the odd one out? (i) calm, (ii) **quiet**, (iii) relaxed, (iv) serene, (v) unruffled.
Synonym	Which word is closest to IRRATIONAL? (i)intransigent, (ii) irredeemable, (iii) unsafe, (iv) lost, (v) **nonsensical**.
Antonym	Which word is most opposite to MUSICAL? (i) **discordant**, (ii) loud, (iii) lyrical, (iv) verbal, (v) euphonious.

Table 1: Types of verbal questions.

sense of the words and identify the outlier. Synonym questions require one to pick one word out of a list of words such that it has the closest meaning to a given word. Synonym questions test the ability of identifying all senses of the candidate words and selecting the correct sense that can form a synonymous relation to the given word. Antonym questions require one to pick one word out of a list of words such that it has the opposite meaning to a given word. Antonym questions test the ability of identifying all senses of the candidate words and selecting the correct sense that can form an antonymous relation to the given word. (Turney, 2008; Turney, 2011) studied the analogy, synonym and antonym problem using a supervised classification approach.

Although there are some efforts to solve mathematical, logical, and picture questions in IQ test (Sanghi and Dowe, 2003; Strannegard et al., 2012; Kushmany et al., 2014; Seo et al., 2014; Hosseini et al., 2014; Weston et al., 2015), there have been very few efforts to develop automatic methods to solve verbal questions.

2.2 Deep Learning for Text Mining

Building distributed word representations (Bengio et al., 2003), a.k.a. word embeddings, has attracted increasing attention in the area of machine learning. Different from conventional *one-hot* representations of studies or distributional word representations based on co-occurrence matrix between words such as LSA (Dumais et al., 1988) and LDA (Blei et al., 2003), distributed word representations are usually low-dimensional dense vectors trained with neural networks by maximizing the likelihood of a text corpus. Recently, a series of works applied deep learning techniques to learn high-quality word representations (Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014).

Nevertheless, since the above works learn word representations mainly based on the word co-occurrence information, it is quite difficult to obtain high quality embeddings for those words with very little context information; on the other hand, a large amount of noisy or biased context could give rise to ineffective word embeddings. Therefore, it is necessary to introduce extra knowledge into the learning process to regularize the quality of word embedding. Some efforts have paid attention to learn word embedding in order to address knowledge base completion and enhancement (Bordes et al., 2011; Socher et al., 2013; Weston et al., 2013a), and some other efforts have tried to leverage knowledge to enhance word representations (Luong et al., 2013; Weston et al., 2013b; Fried and Duh, 2014; Celikyilmaz et al., 2015). Moreover, all the above models assume that one word has only one embedding no matter whether the word is polysemous or not, which might cause some confusion for the polysemous words. To solve the problem, there are several efforts like (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014). However, these models do not leverage any extra knowledge (e.g., relational knowledge) to enhance word representations.

3 Solving Verbal Questions

In this section, we introduce our proposed framework to solve the verbal questions, which consists of the following three components.

3.1 Classification of Question Types

The first component of the framework is a question classifier, which identifies different types of verbal questions. Since different types of questions have their unique ways of expression, the classification task is relatively easy, and we therefore take a simple approach to fulfill the task. Specifically, we regard each verbal question as a short document and use the TF·IDF features to build its representation. Then we train an SVM classifier with linear kernel on a portion of labeled question data, and apply it to other

questions. The question labels include Analogy-I, Analogy-II, Classification, Synonym, and Antonym. We use the *one-vs-rest* training strategy to obtain a linear SVM classifier for each question type.

3.2 Embedding of Word-Senses and Relations

The second component of our framework leverages deep learning technologies to learn distributed representations for words (i.e. word embedding). Note that in the context of verbal question answering, we have some specific requirements on this learning process. Verbal questions in IQ tests usually consider the multiple senses of a word (and focus on the rare senses), and the complex relations among (polysemous) words, such as synonym and antonym relation. Figure 1 shows an example of the multi-sense of words and the relations among word senses. We can see that *irrational* has three senses. Its first sense has an antonym relation with the second sense of *rational*, while its second sense has a synonym relation with *nonsensical* and an antonym relation with the first sense of *rational*.

The above challenge has exceeded the capability of standard word embedding technologies. To address this problem, we propose a novel approach that considers the multi-sense nature of words and integrate the relational knowledge among words (or their senses) into the learning process. In particular, our approach consists of two steps. The first step aims at labeling a word in the text corpus with its specific sense, and the second step employs both the labeled text corpus and the relational knowledge contained in dictionaries to simultaneously learn embeddings for both word-sense pairs and relations.

3.2.1 Multi-Sense Identification

First, we learn a single-sense word embedding by using the skip-gram method in *word2vec* (Mikolov et al., 2013).

Second, we gather the context windows of all occurrences of a word used in the skip-gram model, and represent each context by a weighted average of the pre-learned embedding vectors of the context words. We use TF·IDF to define the weighting function, where we regard each context window of the word as a short document to calculate the document frequency. Specifically, for a word w_0, each of its context window can be denoted by $(w_{-N}, \cdots, w_0, \cdots, w_N)$. Then we represent the window by calculating the weighted average of the pre-learned embedding vectors of the context words as $\xi = \frac{1}{2N}\sum_{i=-N, i\neq 0}^{N} g_{w_i} v_{w_i}$,where g_{w_i} is the TF·IDF score of w_i, and v_{w_i} is the pre-learned embedding vector of w_i. After that, for each word, we use spherical k-means to cluster all its context representations, where cluster number k is set as the number of senses of this word in the online dictionary.

Third, we match each cluster to the corresponding sense in the dictionary. On one hand, we represent each cluster by the average embedding vector of all those context windows included in the cluster. For example, suppose word w_0 has k senses and thus it has k clusters of context windows, we denote the average embedding vectors for these clusters as $\bar{\xi}_1, \cdots, \bar{\xi}_k$. On the other hand, since the online dictionary uses some descriptions and example sentences to interpret each word sense, we can represent each word sense by the average embedding of those words including its description words and the words in the corresponding example sentences. Here, we assume the representation vectors (based on the online dictionary) for the k senses of w_0 are $\zeta_1, \cdots, \zeta_k$. After that, we consecutively match each cluster to its closest word sense in terms of the distance computed in the word embedding space:

$$(\bar{\xi}_{i'}, \zeta_{j'}) = \underset{i,j=1,\cdots,k}{\operatorname{argmin}} d(\bar{\xi}_i, \zeta_j), \tag{1}$$

where $d(\cdot,\cdot)$ calculates the Euclidean distance and $(\bar{\xi}_{i'}, \zeta_{j'})$ is the first matched pair of window cluster and word sense. Here, we simply take a greedy strategy. That is, we remove $\bar{\xi}_{i'}$ and $\zeta_{j'}$ from the cluster vector set and the sense vector set, and recursively run (1) to find the next matched pair till all the pairs are found. Finally, each word occurrence in the corpus is relabeled by its associated word sense, which will be used to learn the embeddings for word-sense pairs in the next step.

3.2.2 Co-Learning Word-Sense Pair Representations and Relation Representations

After relabeling the text corpus, different occurrences of a polysemous word may correspond to its different senses, or more accurately word-sense pairs. We then learn the embeddings for word-

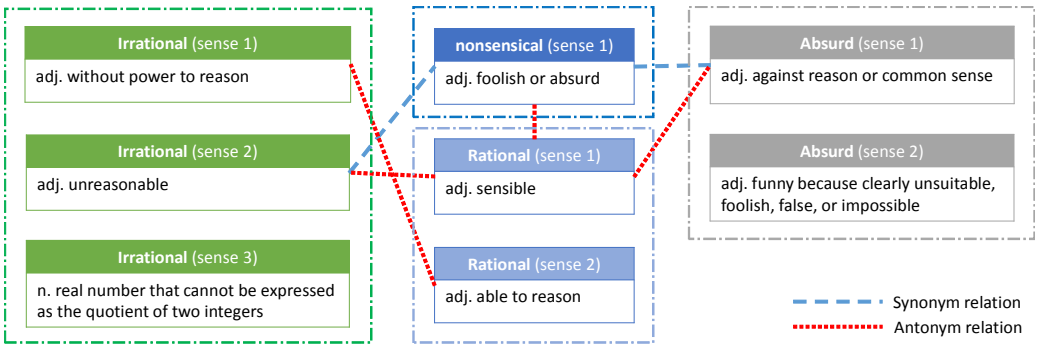


Figure 1: An example on the multi-sense of words and the relations between word senses.

sense pairs and relations (obtained from dictionaries, such as synonym and antonym) simultaneously, by integrating relational knowledge into the objective function of the word embedding learning model like skip-gram. We propose to use a function E_r as described below to capture the relational knowledge.

Specifically, the existing relational knowledge extracted from dictionaries, such as synonym, antonym, etc., can be naturally represented in the form of a triplet (*head*, *relation*, *tail*) (denoted by $(h_i, r, t_j) \in S$, where S is the set of relational knowledge), which consists of two word-sense pairs (i.e. word h with its i-th sense and word t with its j-th sense), $h, t \in W$ (W is the set of words) and a relationship $r \in R$ (R is the set of relationships). To learn the relation representations, we make an assumption that relationships between words can be interpreted as translation operations and they can be represented by vectors. The principle in this model is that if the relationship (h_i, r, t_j) exists, the representation of the word-sense pair t_j should be close to that of h_i plus the representation vector of the relationship r, i.e. $h_i + r$; otherwise, $h_i + r$ should be far away from t_j. Note that this model learns word-sense pair representations and relation representations in a unified continuous embedding space.

According to the above principle, we define E_r as a margin-based regularization function over the set of relational knowledge S,

$$E_r = \sum_{\substack{(h_i,r,t_j)\in S \\ (h^{'},r,t^{'})\in S'_{(h_i,r,t_j)}}} \left[\gamma + d(h_i + r, t_j) - d(h^{'} + r, t^{'})\right]_+ .$$

Here $[X]_+ = \max(X, 0)$, $\gamma > 0$ is a margin hyper-parameter, and $d(\cdot, \cdot)$ is the Euclidean distance between two words in the embedding space.The set of corrupted triplets $S'_{(h,r,t)}$ is defined as $S'_{(h_i,r,t_j)} = \{(h^{'}, r, t)\} \bigcup \{(h, r, t^{'})\}$,which is constructed from S by replacing either the head word-sense pair or the tail word-sense pair by another randomly selected word with its randomly selected sense.

To avoid the trivial solution that simply increases the norms of representation vectors, we use an additional soft norm constraint on the relation representations as $r_i = 2\sigma(x_i) - 1$, where $\sigma(\cdot)$ is the sigmoid function $\sigma(x_i) = 1/(1 + e^{-x_i})$, r_i is the i-th dimension of relation vector r, and x_i is a latent variable, which guarantees that every dimension of the relation representation vector is within the range $(-1, 1)$.

By combining the skip-gram objective function and the regularization function derived from relational knowledge, we get the combined objective $J_r = \alpha E_r - L$ that incorporates relational knowledge into the word-sense pair embedding calculation process, where α is the combination coefficient. Our goal is to minimize J_r, which can be optimized using back propagation neural networks. Figure 2 shows the structure of the proposed model.By using this model, we can obtain the distributed representations for both word-sense pairs and relations simultaneously.

3.3 Solvers for Each Type of Questions

3.3.1 Analogy-I

For the Analogy-I questions like "A is to B as C is to ?", we answer them by optimizing:

$$D = \underset{\substack{i_b,i_a,i_c,i_{d'};\\ D'\in T}}{\operatorname{argmax}} \; cos(v_{(B,i_b)} - v_{(A,i_a)} + v_{(C,i_c)}, v_{(D',i_{d'})}) \tag{2}$$

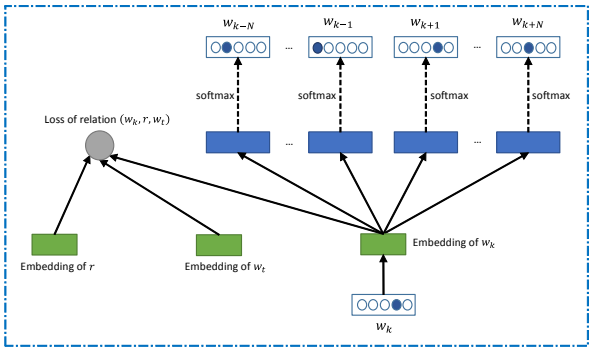


Figure 2: The structure of the proposed model.

where T contains all the candidate answers, cos means cosine similarity, and $i_b, i_a, i_c, i_{d'}$ are the indexes for the word senses of B, A, C, D' respectively. Finally D is selected as the answer.

3.3.2 Analogy-II

As the form of the Analogy-II questions is like "A is to ? as C is to ?" with two lists of candidate answers, we can apply an optimization method as below to select the best (B, D) pair,

$$\underset{\substack{i_{b'}, i_a, i_c, i_{d'}; \\ B' \in T_1, D' \in T_2}}{\operatorname{argmax}} cos(v_{(B', i_{b'})} - v_{(A, i_a)} + v_{(C, i_c)}, v_{(D', i_{d'})}), \tag{3}$$

where T_1, T_2 are two lists of candidate words. Thus we get the answers B and D that can form an analogical relation between word pair (A, B) and word pair (C, D) under a certain specific word sense combination.

3.3.3 Classification

For the Classification questions, we leverage the property that words with similar co-occurrence information are distributed close to each other in the embedding space. The candidate word that is not similar to others does not have similar co-occurrence information to other words in the training corpus, and thus this word should be far away from other words in the word embedding space. Therefore we first calculate a group of mean vectors $m_{i_{w_1}, \cdots, i_{w_N}}$ of all the candidate words with any possible word senses as below $m_{i_{w_1}, \cdots, i_{w_N}} = \frac{1}{N} \sum_{w_j \in T} v_{(w_j, i_{w_j})}$, where T is the set of candidate words, N is the capacity of T, w_j is a word in T; $i_{w_j} (j = 1, \cdots, N; i_{w_j} = 1, \cdots, k_{w_j})$ is the index for the word senses of w_j, and $k_{w_j} (j = 1, \cdots, N)$ is the number of word senses of w_j. Therefore, the number of the mean vectors is $M = \prod_{j=1}^{N} k_{w_j}$. As both N and k_{w_j} are very small, the computation cost is acceptable. Then, we choose the word with such a sense that its closest sense to the corresponding mean vector is the largest among the candidate words as the answer, i.e.,

$$w = \underset{w_j \in T}{\operatorname{argmax}} \min_{i_{w_j}; l=1, \cdots, M} d(v_{(w_j, i_{w_j})}, m_l). \tag{4}$$

3.3.4 Synonym

For the Synonym questions, we empirically explored two solvers. For the first solver, we also leverage the property that words with similar co-occurrence information are located closely in the word embedding space. Therefore, given the question word w_q and the candidate words w_i, we can find the answer by solving:

$$w = \underset{i_{w_q}, i_{w_j}; w_j \in T}{\operatorname{argmin}} d(v_{(w_j, i_{w_j})}, v_{(w_q, i_{w_q})}), \tag{5}$$

where T is the set of candidate words. The second solver is based on the minimization objective of the translation distance between entities in the relational knowledge model (2). Specifically, we calculate the offset vector between the embedding of question word w_q and each word w_j in the candidate list. Then, we set the answer w as the candidate word with which the offset is the closest to the representation vector of the synonym relation r_s, i.e.,

$$w = \underset{i_{w_q}, i_{w_j}; w_j \in T}{\operatorname{argmin}} \big| |v_{(w_j, i_{w_j})} - v_{(w_q, i_{w_q})}| - r_s \big|. \tag{6}$$

In practice, we found the second solver performs better (the results are listed in Section 4). For our baseline embedding model skip-gram, since it does not assume the relation representations explicitly, we use the first solver for it.

3.3.5 Antonym

Similar to solving the Synonym questions, we explored two solvers for Antonym questions as well. That is, the first solver (7) is based on the small offset distance between semantically close words whereas the second solver (8) leverages the translation distance between two words' offset and the embedding vector of the antonym relation. The first solver is based on the fact that since an antonym and

its original word have similar co-occurrence information from which the embedding vectors are derived, the embedding vectors of both words with antonym relation will still lie closely in the embedding space.

$$w = \underset{i_{w_q}, i_{w_j}; w_j \in T}{\operatorname{argmin}} d(v_{(w_j, i_{w_j})}, v_{(w_q, i_{w_q})}), \quad (7)$$

$$w = \underset{i_{w_q}, i_{w_j}; w_j \in T}{\operatorname{argmin}} \big| |v_{(w_j, i_{w_j})} - v_{(w_q, i_{w_q})}| - r_a \big|, \quad (8)$$

Here T is the set of candidate words and r_a is the representation vector of the antonym relation. Again we found that the second solver performs better. Similarly, for skip-gram, the first solver is applied.

4 Experiments

We conduct experiments to examine whether our proposed framework can achieve satisfying results on verbal comprehension questions.

4.1 Data Collection

4.1.1 Training Set for Word Embedding

We trained word embeddings on a publicly available text corpus named *wiki2014*[2], which is a large text snapshot from Wikipedia. After being preprocessed by removing all the *html* meta-data and replacing the digit numbers by English words, the final training corpus contains more than 3.4 billion word tokens, and the number of unique words, i.e. the vocabulary size, is about 2 million.

4.1.2 IQ Test Set

According to our study, there is no online dataset specifically released for verbal comprehension questions, although there are many online IQ tests for users to play with. In addition, most of the online tests only calculate the final IQ scores but do not provide the correct answers. Therefore, we only use the online questions to train the verbal question classifier described in Section 3.1. Specifically, we manually collected and labeled 30 verbal questions from the online IQ test Websites[3] for each of the five types (i.e. Analogy-I, Analogy-II, Classification, Synonym, and Antonym) and trained an *one-vs-rest* SVM classifier for each type. The total accuracy on the training set itself is 95.0%. The classifier was then applied in the test set below.

We collected a set of verbal comprehension questions associated with correct answers from published IQ test books, such as (Carter, 2005; Carter, 2007; Pape, 1993; Ken Russell, 2002), and we used this collection as the test set to evaluate the effectiveness of our new framework. In total, this test set contains 232 questions with the corresponding answers.[4] The number of each question type (i.e., Analogy-I, Analogy-II, Classification, Synonym, Antonym) are respectively $50, 29, 53, 51, 49$.

4.2 Compared Methods

In our experiments, we compare our new relation knowledge powered model to several baselines.

Random Guess Model (RG). Random guess is the most straightforward way for an agent to solve questions. In our experiments, we used a random guess agent which would select an answer randomly regardless of what the question was. To measure the performance of random guess, we ran each task for 5 times and calculated the average accuracy.

Human Performance (HP). Since IQ tests are designed to evaluate human intelligence, it is quite natural to leverage human performance as a baseline. To collect human answers on the test questions, we delivered them to human beings through Amazon Mechanical Turk (AMT), a crowd-sourcing Internet marketplace that allows people to participate in Human Intelligence Tasks. In our study, we published five AMT jobs, one job corresponding to one specific question type. The jobs were delivered to 200 people. To control the quality of the collected results, we used several strategies: (i) we imposed high restrictions on the workers by requiring all the workers to be native English speakers in North America and to be AMT Masters (who have demonstrated high accuracy on previous tasks on AMT marketplace); (ii) we recruited a large number of workers in order to guarantee the statistical confidence in their performances; (iii) we tracked their age distribution and education background, which

[2] http://en.wikipedia.org/wiki/Wikipedia:Database_download

[3] http://wechsleradultintelligencescale.com/

[4] It can be downloaded from https://www.dropbox.com/s/o0very1gwv3mrt5/VerbalQuestions.zip?dl=0.

are very similar to those of the overall population in the U.S.

Latent Dirichlet Allocation Model (LDA). This baseline model leveraged one of the most common classical distributional word representations, i.e. Latent Dirichlet Allocation (LDA) (Blei et al., 2003). In particular, we trained word representations using LDA on *wiki2014* with the topic number 1000.

Skip-Gram Model (SG). In this baseline, we applied the word embedding trained by skip-gram (Mikolov et al., 2013) (denoted by **SG-1**) on *wiki2014*. In particular, we set the window size as 5, the embedding dimension as 500, the negative sampling count as 3, and the epoch number as 3. In addition, we also employed a pre-trained word embedding by Google[5] with the dimension of 300 (denoted by **SG-2**).

Glove. Another powerful word embedding model (Pennington et al., 2014). **Glove** configurations are the same as those in running **SG-1**.

Multi-Sense Model (MS). In this baseline, we applied the multi-sense word embedding models proposed in (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014) (denoted by **MS-1**, **MS-2** and **MS-3** respectively). For **MS-1**, we directly used the published multi-sense word embedding vectors by the authors[6], in which they set 10 senses for the top 5% most frequent words. For **MS-2** and **MS-3**, we get the embedding vectors by usingf the released codes from the authors using the same configurations as **MS-1**.

Relation Knowledge Powered Model (RK). This is our proposed method in Section 3. In particular, when learning the embedding on *wiki2014*, we set the window size as 5, the embedding dimension as 500, the negative sampling count as 3, and the epoch number as 3. We adopted the online Longman Dictionary as the dictionary used in multi-sense clustering. We used a public relation knowledge set, WordRep (Gao et al., 2014), for relation training.

4.3 Experimental Results

4.3.1 Accuracy of Question Classifier

We applied the question classifier trained in Section 4.1.2 on the test set, and got the total accuracy 93.1%. For RG and HP, the question classifier was not needed. For other methods, the wrongly classified questions were also sent to the corresponding wrong solver to find an answer. If the solver returned an empty result (which was usually caused by invalid input format, e.g., an Analogy-II question was wrongly input to the Classification solver), we would randomly select an answer.

4.3.2 Overall Accuracy

Table 2 demonstrates the accuracy of answering verbal questions by using all the approaches mentioned in Section 4.2. The numbers for all the models are mean values from five repeated runs. From this table, we observe: (i) RK can achieve the best overall accuracy than all the other methods. In particular, RK can raise the overall accuracy by about 4.63% over HP[7]. (ii) RK is empirically superior to the skip-gram models SG-1/SG-2 and Glove. According to our understanding, the improvement of RK over SG-1/SG-2/Glove comes from two aspects: multi-sense and relational knowledge. Note that the performance difference between MS-1/MS-2/MS-3 and SG-1/SG-2/Glove is not significant, showing that simply changing single-sense word embedding to multi-sense word embedding does not bring too much benefit. One reason is that the rare word-senses do not have enough training data (contextual information) to produce high-quality word embedding. By further introducing the relational knowledge among word-senses, the training for rare word-senses will be linked to the training of their related word-senses. As a result, the embedding quality of the rare word-senses will be improved. (iii) RK is empirically superior than the two multi-sense algorithms MS-1, MS-2 and MS-3, demonstrating the effectiveness brought by adopting fewer model parameters and using an online dictionary in building the multi-sense embedding model.

These results are quite impressive, indicating the potential of using machines to comprehend human knowledge and even achieve a comparable level of human intelligence.

4.3.3 Accuracy on Different Question Types

Table 2 reports the accuracy of answering various types of verbal questions by each method. From the

[5] https://code.google.com/p/word2vec/

[6] http://ai.stanford.edu/~ehhuang/

[7] With the t-test score $p = 0.036$.

	Analogy-I	Analogy-II	Classification	Synonym	Antonym	**Total**
RG	**24.60**	**11.72**	**20.75**	**19.27**	**23.13**	**20.51**
LDA	**28.00**	**13.79**	**39.62**	**27.45**	**30.61**	**29.31**
HP	**45.87**	**34.37**	**47.23**	**50.38**	**53.30**	**46.23**
SG						
SG-1	**38.00**	**24.14**	**37.74**	**45.10**	**40.82**	**38.36**
SG-2	**38.00**	**20.69**	**39.62**	**47.06**	**44.90**	**39.66**
Glove	**45.09**	**24.14**	**32.08**	**47.06**	**40.82**	**39.03**
MS						
MS-1	**36.36**	**19.05**	**41.30**	**50.00**	**36.59**	**38.67**
MS-2	**40.00**	**20.69**	**41.51**	**49.02**	**40.82**	**40.09**
MS-3	**17.65**	**20.69**	**47.17**	**47.06**	**30.61**	**36.73**
RK	**48.00**	**34.48**	**52.83**	**60.78**	**51.02**	**50.86**

Table 2: Accuracy of different methods among different human groups.

table, we can observe that the SG and MS models can achieve competitive accuracy on certain question types (like Synonym) compared with HP. After incorporating knowledge into learning word embedding, our RK model can improve the accuracy over all question types. Moreover, the table shows that RK can result in a big improvement over HP on the question types of Synonym and Classification.

To sum up, the experimental results have demonstrated the effectiveness of the proposed RK model compared with several baseline methods. Although the test set is not large, the generalization of RK to other test sets should not be a concern due to the unsupervised nature of our model.

5 Conclusions

We investigated how to automatically solve verbal comprehension questions in IQ Tests by using word embedding techniques. In particular, we proposed a three-step framework: (i) to recognize the specific type of a verbal comprehension question by a classifier, (ii) to leverage a novel deep learning model to co-learn the representations of both word-sense pairs and relations among words (or their senses), (iii) to design dedicated solvers, based on the obtained word-sense pair representations and relation representations, for addressing each type of questions. Experimental results have demonstrated that this novel framework can achieve better performance than existing methods for solving verbal comprehension questions and even exceed the average performance of the Amazon Mechanical Turk workers involved in the experiments.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Philip Carter. 2005. *The complete book of intelligence tests*. John Wiley & Sons Ltd.

Philip Carter. 2007. *The Ultimate IQ Test Book: 1,000 Practice Test Questions to Boost Your Brain Power*. Kogan Page Publishers.

Asli Celikyilmaz, Dilek Hakkani-Tur, Panupong Pasupat, and Ruhi Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. In *Proceedings of AAAI*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167. ACM.

Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of SIGCHI*.

Daniel Fried and Kevin Duh. 2014. Incorporating both distributional and relational semantics in word representations. *CoRR*, abs/1412.4369.

Bin Gao, Jiang Bian, and Tie-Yan Liu. 2014. Wordrep: A benchmark for research on learning word representations. *arXiv preprint arXiv:1407.1640*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of EMNLP*, pages 523–533.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Association for Computational Linguistics (ACL)*, pages 873–882.

Philip Carter Ken Russell. 2002. *The Times Book of IQ Tests*. Kogan Page Limited.

Nate Kushmany, Yoav Artziz, Luke Zettlemoyerz, and Regina Barzilayy. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL*.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

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

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.

Dan Pape. 1993. *The Original Cambridge Self Scoring IQ Test*. The Magni Group, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of EMNLP*, 12:1532–1543.

Balázs Pintér, Gyula Vörös, Zoltán Szabó, and András Lörincz. 2012. Automated word puzzle generation via topic dictionaries. *CoRR*, abs/1206.0377.

Pritika Sanghi and David Dowe. 2003. A computer program capable of passing i.q. tests. In *Proceedings of the Joint International Conference on Cognitive Science*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 - 31, 2014, Québec City, Québec, Canada.*, pages 2831–2838.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, pages 926–934.

William Stern. 1914. *The Psychological Methods of Testing Intelligence*. Warwick & York.

Claes Strannegard, Mehrdad Amirghasemi, and Simon Ulfsbacker. 2012. An anthropomorphic method for number sequence problems. *Cognitive Systems Research*.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING*.

Peter D Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the Coling 2008*, pages 905–912.

Peter D Turney. 2011. Analogy perception applied to seven tests of word comprehension. *Journal of Experimental & Theoretical Artificial Intelligence*, 23(3):343–362.

David Wechsler. 2008. Wechsler adult intelligence scale–fourth edition (wais–iv). *San Antonio, TX: NCS Pearson*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013a. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013b. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of EMNLP*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Long Short-Term Memory-Networks for Machine Reading

Jianpeng Cheng, Li Dong and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
{jianpeng.cheng,li.dong}@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we address the question of how to render sequence-level networks better at handling structured input. We propose a machine reading simulator which processes text incrementally from left to right and performs shallow reasoning with memory and attention. The reader extends the Long Short-Term Memory architecture with a memory network in place of a single memory cell. This enables adaptive memory usage during recurrence with neural attention, offering a way to weakly induce relations among tokens. The system is initially designed to process a single sequence but we also demonstrate how to integrate it with an encoder-decoder architecture. Experiments on language modeling, sentiment analysis, and natural language inference show that our model matches or outperforms the state of the art.

1 Introduction

How can a sequence-level network induce relations which are presumed latent during text processing? How can a recurrent network attentively memorize longer sequences in a way that humans do? In this paper we design a machine reader that automatically learns to understand text. The term machine reading is related to a wide range of tasks from answering reading comprehension questions (Clark et al., 2013), to fact and relation extraction (Etzioni et al., 2011; Fader et al., 2011), ontology learning (Poon and Domingos, 2010), and textual entailment (Dagan et al., 2005). Rather than focusing on a specific task, we develop a general-purpose reading simulator, drawing inspiration from human language processing and the fact language comprehension is incremental with readers continuously extracting the meaning of utterances on a word-by-word basis.

In order to understand texts, our machine reader should provide facilities for extracting and representing meaning from natural language text, storing meanings internally, and working with stored meanings to derive further consequences. Ideally, such a system should be robust, open-domain, and degrade gracefully in the presence of semantic representations which may be incomplete, inaccurate, or incomprehensible. It would also be desirable to simulate the behavior of English speakers who process text sequentially, from left to right, fixating nearly every word while they read (Rayner, 1998) and creating partial representations for sentence prefixes (Konieczny, 2000; Tanenhaus et al., 1995).

Language modeling tools such as recurrent neural networks (RNN) bode well with human reading behavior (Frank and Bod, 2011). RNNs treat each sentence as a sequence of words and recursively compose each word with its previous *memory*, until the meaning of the whole sentence has been derived. In practice, however, sequence-level networks are met with at least three challenges. The first one concerns model training problems associated with vanishing and exploding gradients (Hochreiter, 1991; Bengio et al., 1994), which can be partially ameliorated with gated activation functions, such as the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and gradient clipping (Pascanu et al., 2013). The second issue relates to memory compression problems. As the input sequence gets compressed and blended into a single dense vector, suf-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 551–561,
Austin, Texas, November 1-5, 2016.

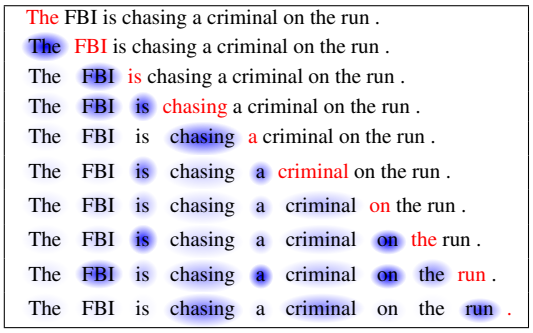


Figure 1: Illustration of our model while reading the sentence *The FBI is chasing a criminal on the run.* Color *red* represents the current word being fixated, *blue* represents memories. Shading indicates the degree of memory activation.

ficiently large memory capacity is required to store past information. As a result, the network generalizes poorly to long sequences while wasting memory on shorter ones. Finally, it should be acknowledged that sequence-level networks lack a mechanism for handling the structure of the input. This imposes an inductive bias which is at odds with the fact that language has inherent structure. In this paper, we develop a text processing system which addresses these limitations while maintaining the incremental, generative property of a recurrent language model.

Recent attempts to render neural networks more structure aware have seen the incorporation of external memories in the context of recurrent neural networks (Weston et al., 2015; Sukhbaatar et al., 2015; Grefenstette et al., 2015). The idea is to use multiple memory slots outside the recurrence to piece-wise store representations of the input; read and write operations for each slot can be modeled as an attention mechanism with a recurrent controller. We also leverage memory and attention to empower a recurrent network with stronger memorization capability and more importantly the ability to discover relations among tokens. This is realized by inserting a memory network module in the update of a recurrent network together with attention for memory addressing. The attention acts as a weak inductive module discovering relations between input tokens, and is trained without direct supervision. As a point of departure from previous work, the memory network we employ is internal to the recurrence, thus strengthening the interaction of the two and leading to a representation learner which is able to reason over shallow structures. The resulting model, which we term Long Short-Term Memory-Network (LSTMN), is a reading simulator that can be used for sequence processing tasks.

Figure 1 illustrates the reading behavior of the LSTMN. The model processes text incrementally while learning which past tokens in the memory and to what extent they relate to the current token being processed. As a result, the model induces undirected relations among tokens as an intermediate step of learning representations. We validate the performance of the LSTMN in language modeling, sentiment analysis, and natural language inference. In all cases, we train LSTMN models end-to-end with task-specific supervision signals, achieving performance comparable or better to state-of-the-art models and superior to vanilla LSTMs.

2 Related Work

Our machine reader is a recurrent neural network exhibiting two important properties: it is incremental, simulating human behavior, and performs shallow structure reasoning over input streams.

Recurrent neural network (RNNs) have been successfully applied to various sequence modeling and sequence-to-sequence transduction tasks. The latter have assumed several guises in the literature such as machine translation (Bahdanau et al., 2014), sentence compression (Rush et al., 2015), and reading comprehension (Hermann et al., 2015). A key contributing factor to their success has been the ability to handle well-known problems with exploding or vanishing gradients (Bengio et al., 1994), leading to models with gated activation functions (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), and more advanced architectures that enhance the information flow within the network (Koutník et al., 2014; Chung et al., 2015; Yao et al., 2015).

A remaining practical bottleneck for RNNs is memory compression (Bahdanau et al., 2014): since the inputs are recursively combined into a single memory representation which is typically too small in terms of parameters, it becomes difficult to accurately memorize sequences (Zaremba and Sutskever, 2014). In the encoder-decoder architecture, this problem can be sidestepped with an attention mechanism which learns soft alignments *between* the decoding states and the encoded memories (Bahdanau

et al., 2014). In our model, memory and attention are added *within* a sequence encoder allowing the network to uncover lexical relations between tokens.

The idea of introducing a structural bias to neural models is by no means new. For example, it is reflected in the work of Socher et al. (2013a) who apply recursive neural networks for learning natural language representations. In the context of recurrent neural networks, efforts to build modular, structured neural models date back to Das et al. (1992) who connect a recurrent neural network with an external memory stack for learning context free grammars. Recently, Weston et al. (2015) propose Memory Networks to explicitly segregate memory storage from the computation of neural networks in general. Their model is trained end-to-end with a memory addressing mechanism closely related to soft attention (Sukhbaatar et al., 2015) and has been applied to machine translation (Meng et al., 2015). Grefenstette et al. (2015) define a set of differentiable data structures (stacks, queues, and dequeues) as memories controlled by a recurrent neural network. Tran et al. (2016) combine the LSTM with an external memory block component which interacts with its hidden state. Kumar et al. (2016) employ a structured neural network with episodic memory modules for natural language and also visual question answering (Xiong et al., 2016).

Similar to the above work, we leverage memory and attention in a recurrent neural network for inducing relations between tokens as a module in a larger network responsible for representation learning. As a property of soft attention, all intermediate relations we aim to capture are soft and differentiable. This is in contrast to shift-reduce type neural models (Dyer et al., 2015; Bowman et al., 2016) where the intermediate decisions are hard and induction is more difficult. Finally, note that our model captures undirected lexical relations and is thus distinct from work on dependency grammar induction (Klein and Manning, 2004) where the learned head-modifier relations are directed.

3 The Machine Reader

In this section we present our machine reader which is designed to process structured input while retaining the incrementality of a recurrent neural network. The core of our model is a Long Short-Term Memory (LSTM) unit with an extended memory tape that explicitly simulates the human memory span. The model performs implicit relation analysis between tokens with an attention-based memory addressing mechanism at every time step. In the following, we first review the standard Long Short-Term Memory and then describe our model.

3.1 Long Short-Term Memory

A Long Short-Term Memory (LSTM) recurrent neural network processes a variable-length sequence $x = (x_1, x_2, \cdots, x_n)$ by incrementally adding new content into a single memory slot, with gates controlling the extent to which new content should be memorized, old content should be erased, and current content should be exposed. At time step t, the memory c_t and the hidden state h_t are updated with the following equations:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t] \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3}$$

where i, f, and o are gate activations. Compared to the standard RNN, the LSTM uses additive memory updates and it separates the memory c from the hidden state h, which interacts with the environment when making predictions.

3.2 Long Short-Term Memory-Network

The first question that arises with LSTMs is the extent to which they are able to memorize sequences under recursive compression. LSTMs can produce a list of state representations during composition, however, the next state is always computed from the current state. That is to say, given the current state h_t, the next state h_{t+1} is conditionally independent of states $h_1 \cdots h_{t-1}$ and tokens $x_1 \cdots x_t$. While the recursive state update is performed in a Markov manner, it is assumed that LSTMs maintain unbounded memory (i.e., the current state alone summarizes well the tokens it has seen so far). This assumption may fail in practice, for example when the sequence is long

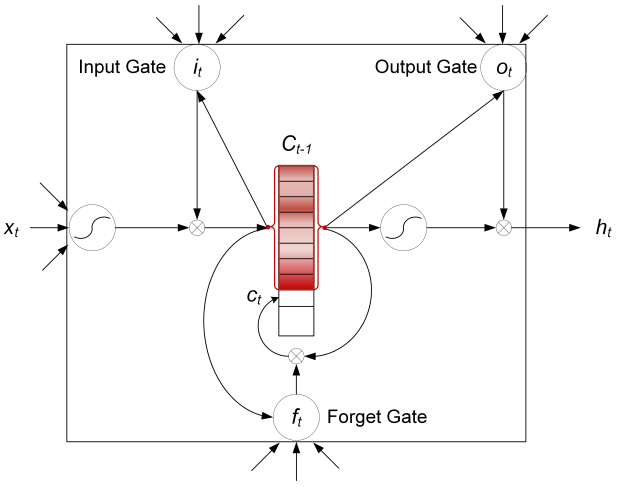


Figure 2: Long Short-Term Memory-Network. Color indicates degree of memory activation.

or when the memory size is not large enough. Another undesired property of LSTMs concerns modeling structured input. An LSTM aggregates information on a token-by-token basis in sequential order, but there is no explicit mechanism for reasoning over structure and modeling relations between tokens.

Our model aims to address both limitations. Our solution is to modify the standard LSTM structure by replacing the memory cell with a memory network (Weston et al., 2015). The resulting Long Short-Term Memory-Network (LSTMN) stores the contextual representation of each input token with a unique memory slot and the size of the memory grows with time until an upper bound of the memory span is reached. This design enables the LSTM to reason about relations between tokens with a neural attention layer and then perform non-Markov state updates. Although it is feasible to apply both write and read operations to the memories with attention, we concentrate on the latter. We conceptualize the *read* operation as attentively linking the current token to previous memories and selecting useful content when processing it. Although not the focus of this work, the significance of the *write* operation can be analogously justified as a way of incrementally updating previous memories, e.g., to correct wrong interpretations when processing garden path sentences (Ferreira and Henderson, 1991).

The architecture of the LSTMN is shown in Figure 2 and the formal definition is provided as follows. The model maintains two sets of vectors stored in a hidden state tape used to interact with the environment (e.g., computing attention), and a memory tape used to represent what is actually stored in memory.[1] Therefore, each token is associated with a hidden vector and a memory vector. Let x_t denote the current input; $C_{t-1} = (c_1, \cdots, c_{t-1})$ denotes the current memory tape, and $H_{t-1} = (h_1, \cdots, h_{t-1})$ the previous hidden tape. At time step t, the model computes the relation between x_t and $x_1 \cdots x_{t-1}$ through $h_1 \cdots h_{t-1}$ with an attention layer:

$$a_i^t = v^{\mathrm{T}} \tanh(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1}) \quad (4)$$

$$s_i^t = \mathrm{softmax}(a_i^t) \quad (5)$$

This yields a probability distribution over the hidden state vectors of previous tokens. We can then compute an adaptive summary vector for the previous hidden tape and memory tape denoted by $\tilde{c}_t$ and $\tilde{h}_t$, respectively:

$$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} h_i \\ c_i \end{bmatrix} \quad (6)$$

and use them for computing the values of c_t and h_t in the recurrent update as:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [\tilde{h}_t, x_t] \quad (7)$$

$$c_t = f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (9)$$

where v, W_h, W_x and $W_{\tilde{h}}$ are the new weight terms of the network.

A key idea behind the LSTMN is to use attention for inducing relations between tokens. These relations are soft and differentiable, and components of a larger representation learning network. Although it is appealing to provide direct supervision for the attention layer, e.g., with evidence collected from a dependency treebank, we treat it as a submodule being optimized within the larger network in a downstream task. It is also possible to have a more structured relational reasoning module by stacking multiple memory and hidden layers in an alternating fashion, resembling a stacked LSTM (Graves,

[1] For comparison, LSTMs maintain a hidden vector and a memory vector; memory networks (Weston et al., 2015) have a set of key vectors and a set of value vectors.

2013) or a multi-hop memory network (Sukhbaatar et al., 2015). This can be achieved by feeding the output h_t^k of the lower layer k as input to the upper layer $(k+1)$. The attention at the $(k+1)$th layer is computed as:

$$a_{i,k+1}^t = v^{\mathrm{T}} \tanh(W_h h_i^{k+1} + W_l h_t^k + W_{\tilde{h}} \tilde{h}_{t-1}^{k+1}) \quad (10)$$

Skip-connections (Graves, 2013) can be applied to feed x_t to upper layers as well.

4 Modeling Two Sequences with LSTMN

Natural language processing tasks such as machine translation and textual entailment are concerned with modeling two sequences rather than a single one. A standard tool for modeling two sequences with recurrent networks is the encoder-decoder architecture where the second sequence (also known as the *target*) is being processed conditioned on the first one (also known as the *source*). In this section we explain how to combine the LSTMN which applies attention for intra-relation reasoning, with the encoder-decoder network whose attention module learns the inter-alignment between two sequences. Figures 3a and 3b illustrate two types of combination. We describe the models more formally below.

Shallow Attention Fusion Shallow fusion simply treats the LSTMN as a separate module that can be readily used in an encoder-decoder architecture, in lieu of a standard RNN or LSTM. As shown in Figure 3a, both encoder and decoder are modeled as LSTMNs with intra-attention. Meanwhile, inter-attention is triggered when the decoder reads a target token, similar to the inter-attention introduced in Bahdanau et al. (2014).

Deep Attention Fusion Deep fusion combines inter- and intra-attention (initiated by the decoder) when computing state updates. We use different notation to represent the two sets of attention. Following Section 3.2, C and H denote the target memory tape and hidden tape, which store representations of the target symbols that have been processed so far. The computation of intra-attention follows Equations (4)–(9). Additionally, we use $A = [\alpha_1, \cdots, \alpha_m]$ and $Y = [\gamma_1, \cdots, \gamma_m]$ to represent the source memory tape and hidden tape, with m being the length of the source sequence conditioned upon. We compute inter-attention between the input at time step t and tokens in the entire source sequence as follows:

$$b_j^t = u^{\mathrm{T}} \tanh(W_\gamma \gamma_j + W_x x_t + W_{\tilde{\gamma}} \tilde{\gamma}_{t-1}) \quad (11)$$

$$p_j^t = \mathrm{softmax}(b_j^t) \quad (12)$$

After that we compute the adaptive representation of the source memory tape $\tilde{\alpha}_t$ and hidden tape $\tilde{\gamma}_t$ as:

$$\begin{bmatrix} \tilde{\gamma}_t \\ \tilde{\alpha}_t \end{bmatrix} = \sum_{j=1}^{m} p_j^t \cdot \begin{bmatrix} \gamma_j \\ \alpha_j \end{bmatrix} \quad (13)$$

We can then transfer the adaptive source representation $\tilde{\alpha}_t$ to the target memory with another gating operation r_t, analogous to the gates in Equation (7).

$$r_t = \sigma(W_r \cdot [\tilde{\gamma}_t, x_t]) \quad (14)$$

The new target memory includes inter-alignment $r_t \odot \tilde{\alpha}_t$, intra-relation $f_t \odot \tilde{c}_t$, and the new input information $i_t \odot \hat{c}_t$:

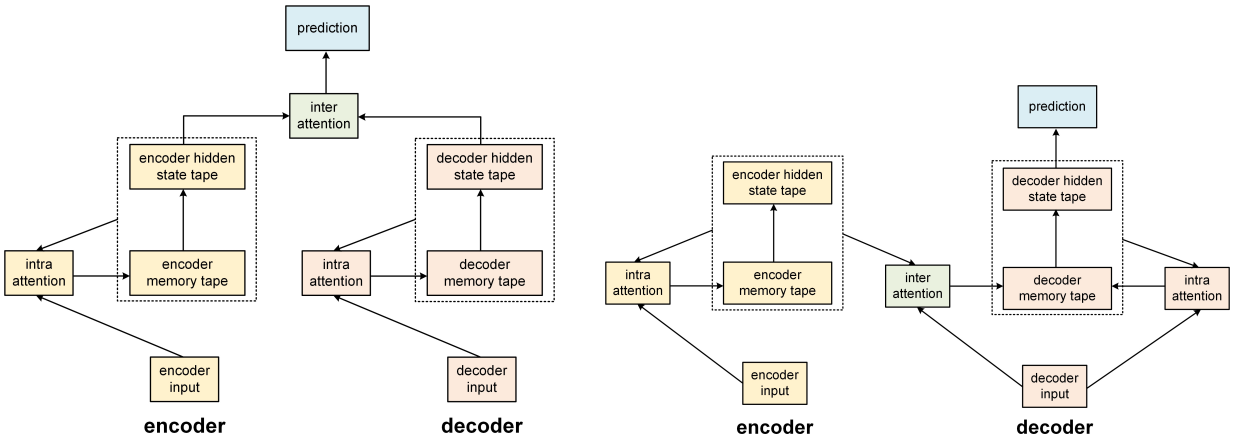
$$c_t = r_t \odot \tilde{\alpha}_t + f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \quad (15)$$

$$h_t = o_t \odot \tanh(c_t) \quad (16)$$

As shown in the equations above and Figure 3b, the major change of deep fusion lies in the recurrent storage of the inter-alignment vector in the target memory network, as a way to help the target network review source information.

5 Experiments

In this section we present our experiments for evaluating the performance of the LSTMN machine reader. We start with language modeling as it is a natural testbed for our model. We then assess the model's ability to extract meaning representations for generic sentence classification tasks such as sentiment analysis. Finally, we examine whether the LSTMN can recognize the semantic relationship between two sentences by applying it to a natural language inference task. Our code is available at `https://github.com/cheng6076/SNLI-attention`.



(a) Decoder with shallow attention fusion.

(b) Decoder with deep attention fusion.

Figure 3: LSTMNs for sequence-to-sequence modeling. The encoder uses intra-attention, while the decoder incorporates both intra- and inter-attention. The two figures present two ways to combine the intra- and inter-attention in the decoder.

Models	Layers	Perplexity
KN5	—	141
RNN	1	129
LSTM	1	115
LSTMN	1	**108**
sLSTM	3	115
gLSTM	3	107
dLSTM	3	109
LSTMN	3	**102**

Table 1: Language model perplexity on the Penn Treebank. The size of memory is 300 for all models.

5.1 Language Modeling

Our language modeling experiments were conducted on the English Penn Treebank dataset. Following common practice (Mikolov et al., 2010), we trained on sections 0–20 (1M words), used sections 21–22 for validation (80K words), and sections 23–24 (90K words for testing). The dataset contains approximately 1 million tokens and a vocabulary size of 10K. The average sentence length is 21. We use perplexity as our evaluation metric: $PPL = \exp(NLL/T)$, where NLL denotes the negative log likelihood of the entire test set and T the corresponding number of tokens. We used stochastic gradient descent for optimization with an initial learning rate of 0.65, which decays by a factor of 0.85 per epoch if no significant improvement has been observed on the validation set. We renormalize the gradient if its norm is greater than 5. The mini-batch size was set to 40. The dimensions of the word embeddings were set to 150 for all models.

In this suite of experiments we compared the LSTMN against a variety of baselines. The first one is a Kneser-Ney 5-gram language model (KN5) which generally serves as a non-neural baseline for the language modeling task. We also present perplexity results for the standard RNN and LSTM models. We also implemented more sophisticated LSTM architectures, such as a stacked LSTM (sLSTM), a gated-feedback LSTM (gLSTM; Chung et al. (2015)) and a depth-gated LSTM (dLSTM; Yao et al. (2015)). The gated-feedback LSTM has feedback gates connecting the hidden states across multiple time steps as an adaptive control of the information flow. The depth-gated LSTM uses a depth gate to connect memory cells of vertically adjacent layers. In general, both gLSTM and dLSTM are able to capture long-term dependencies to some degree, but they do not explicitly keep past memories. We set the number of layers to 3 in this experiment, mainly to agree with the language modeling experiments of Chung et al. (2015). Also note that that there are no single-layer variants for gLSTM and dLSTM; they have to be implemented as multi-layer systems. The hidden unit size of the LSTMN and all comparison models (except KN5) was set to 300.

The results of the language modeling task are shown in Table 1. Perplexity results for KN5 and RNN are taken from Mikolov et al. (2015). As can be seen, the single-layer LSTMN outperforms these

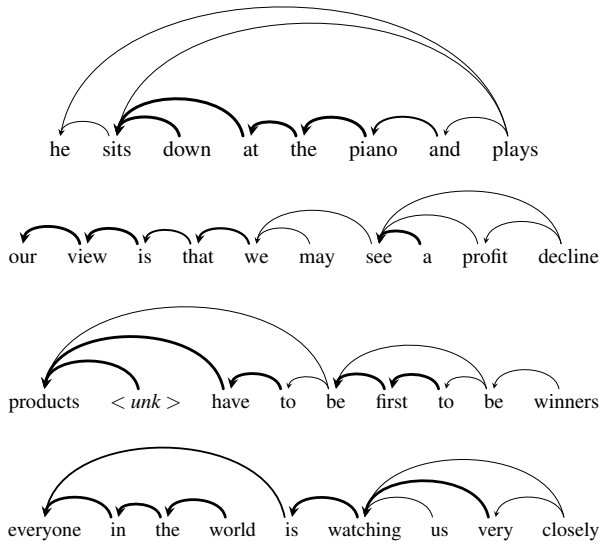


Figure 4: Examples of intra-attention (language modeling). Bold lines indicate higher attention scores. Arrows denote which word is being focused when attention is computed, but not the direction of the relation.

two baselines and the LSTM by a significant margin. Amongst all deep architectures, the three-layer LSTMN also performs best. We can study the memory activation mechanism of the machine reader by visualizing the attention scores. Figure 4 shows four sentences sampled from the Penn Treebank validation set. Although we explicitly encourage the reader to attend to any memory slot, much attention focuses on recent memories. This agrees with the linguistic intuition that long-term dependencies are relatively rare. As illustrated in Figure 4 the model captures some valid lexical relations (e.g., the dependency between *sits* and *at*, *sits* and *plays*, *everyone* and *is*, *is* and *watching*). Note that arcs here are undirected and are different from the directed arcs denoting head-modifier relations in dependency graphs.

5.2 Sentiment Analysis

Our second task concerns the prediction of sentiment labels of sentences. We used the Stanford Sentiment Treebank (Socher et al., 2013a), which contains fine-grained sentiment labels (very positive, positive, neutral, negative, very negative) for 11,855 sentences. Following previous work on this dataset, we used 8,544 sentences for training, 1,101 for validation, and 2,210 for testing. The average sentence length is 19.1. In addition, we also performed a binary classification task (positive, negative) after removing the neutral label. This resulted in 6,920 sentences for training, 872 for validation and 1,821 for testing. Table 2 reports results on both fine-grained and binary classification tasks.

Models	Fine-grained	Binary
RAE (Socher et al., 2011)	43.2	82.4
RNTN (Socher et al., 2013b)	45.7	85.4
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
DCNN (Blunsom et al., 2014)	48.5	86.8
CNN-MC (Kim, 2014)	48.0	88.1
T-CNN (Lei et al., 2015)	**51.2**	**88.6**
PV (Le and Mikolov, 2014)	48.7	87.8
CT-LSTM (Tai et al., 2015)	51.0	88.0
LSTM (Tai et al., 2015)	46.4	84.9
2-layer LSTM (Tai et al., 2015)	46.0	86.3
LSTMN	**47.6**	**86.3**
2-layer LSTMN	**47.9**	**87.0**

Table 2: Model accuracy (%) on the Sentiment Treebank (test set). The memory size of LSTMN models is set to 168 to be compatible with previously published LSTM variants (Tai et al., 2015).

We experimented with 1- and 2-layer LSTMNs. For the latter model, we predict the sentiment label of the sentence based on the averaged hidden vector passed to a 2-layer neural network classifier with ReLU as the activation function. The memory size for both LSTMN models was set to 168 to be compatible with previous LSTM models (Tai et al., 2015) applied to the same task. We used pretrained `300-D Glove 840B` vectors (Pennington et al., 2014) to initialize the word embeddings. The gradient for words with `Glove` embeddings, was scaled by 0.35 in the first epoch after which all word embeddings were updated normally.

We used Adam (Kingma and Ba, 2015) for optimization with the two momentum parameters set to 0.9 and 0.999 respectively. The initial learning rate was set to 2E-3. The regularization constant was 1E-4 and the mini-batch size was 5. A dropout rate of 0.5 was applied to the neural network classifier.

We compared our model with a wide range of top-performing systems. Most of these models (including ours) are LSTM variants (third block in Table 2), recursive neural networks (first block), or convolu-

tional neural networks (CNNs; second block). Recursive models assume the input sentences are represented as parse trees and can take advantage of annotations at the phrase level. LSTM-type models and CNNs are trained on sequential input, with the exception of CT-LSTM (Tai et al., 2015) which operates over tree-structured network topologies such as constituent trees. For comparison, we also report the performance of the paragraph vector model (PV; Le and Mikolov (2014); see Table 2, second block) which neither operates on trees nor sequences but learns distributed document representations parameterized directly.

The results in Table 2 show that both 1- and 2-layer LSTMNs outperform the LSTM baselines while achieving numbers comparable to state of the art. The number of layers for our models was set to be comparable to previously published results. On the fine-grained and binary classification tasks our 2-layer LSTMN performs close to the best system T-CNN (Lei et al., 2015). Figure 5 shows examples of intra-attention for sentiment words. Interestingly, the network learns to associate sentiment important words such as *though* and *fantastic* or *not* and *good*.

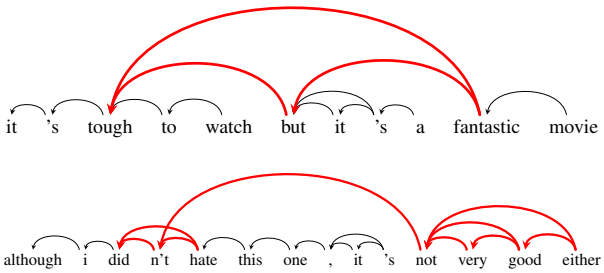


Figure 5: Examples of intra-attention (sentiment analysis). Bold lines (red) indicate attention between sentiment important words.

5.3 Natural Language Inference

The ability to reason about the semantic relationship between two sentences is an integral part of text understanding. We therefore evaluate our model on recognizing textual entailment, i.e., whether two premise-hypothesis pairs are entailing, contradictory, or neutral. For this task we used the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), which contains premise-hypothesis pairs and target labels indicating their relation. After removing sentences with unknown labels, we end up with 549,367 pairs for training, 9,842 for development and 9,824 for testing. The vocabulary size is 36,809 and the average sentence length is 22. We performed lower-casing and tokenization for the entire dataset.

Recent approaches use two sequential LSTMs to encode the premise and the hypothesis respectively, and apply neural attention to reason about their logical relationship (Rocktäschel et al., 2016; Wang and Jiang, 2016). Furthermore, Rocktäschel et al. (2016) show that a non-standard encoder-decoder architecture which processes the hypothesis conditioned on the premise results significantly boosts performance. We use a similar approach to tackle this task with LSTMNs. Specifically, we use two LSTMNs to read the premise and hypothesis, and then match them by comparing their hidden state tapes. We perform average pooling for the hidden state tape of each LSTMN, and concatenate the two averages to form the input to a 2-layer neural network classifier with ReLU as the activation function.

We used pre-trained `300-D Glove 840B` vectors (Pennington et al., 2014) to initialize the word embeddings. Out-of-vocabulary (OOV) words were initialized randomly with Gaussian samples (μ=0, σ=1). We only updated OOV vectors in the first epoch, after which all word embeddings were updated normally. The dropout rate was selected from [0.1, 0.2, 0.3, 0.4]. We used Adam (Kingma and Ba, 2015) for optimization with the two momentum parameters set to 0.9 and 0.999 respectively, and the initial learning rate set to 1E-3. The mini-batch size was set to 16 or 32. For a fair comparison against previous work, we report results with different hidden/memory dimensions (i.e., 100, 300, and 450).

We compared variants of our model against different types of LSTMs (see the second block in Table 3). Specifically, these include a model which encodes the premise and hypothesis independently with two LSTMs (Bowman et al., 2015), a shared LSTM (Rocktäschel et al., 2016), a word-by-word attention model (Rocktäschel et al., 2016), and a matching LSTM (mLSTM; Wang and Jiang (2016)). This model sequentially processes the hypothesis, and at each position tries to match the current word with an attention-weighted representation of the premise (rather than basing its predictions on whole sentence embeddings). We also compared our mod-

Models	h	$\|\theta\|_M$	Test
BOW concatenation	—	—	59.8
LSTM (Bowman et al., 2015)	100	221k	77.6
LSTM-att (Rocktäschel et al., 2016)	100	252k	83.5
mLSTM (Wang and Jiang, 2016)	300	1.9M	86.1
LSTMN	100	260k	81.5
LSTMN shallow fusion	100	280k	84.3
LSTMN deep fusion	100	330k	84.5
LSTMN shallow fusion	300	1.4M	85.2
LSTMN deep fusion	300	1.7M	85.7
LSTMN shallow fusion	450	2.8M	86.0
LSTMN deep fusion	450	3.4M	**86.3**

Table 3: Parameter counts $|\theta|_M$, size of hidden unit h, and model accuracy (%) on the natural language inference task.

els with a bag-of-words baseline which averages the pre-trained embeddings for the words in each sentence and concatenates them to create features for a logistic regression classifier (first block in Table 3).

LSTMNs achieve better performance compared to LSTMs (with and without attention; 2nd block in Table 3). We also observe that fusion is generally beneficial, and that deep fusion slightly improves over shallow fusion. One explanation is that with deep fusion the inter-attention vectors are recurrently memorized by the decoder with a gating operation, which also improves the information flow of the network. With standard training, our deep fusion yields the state-of-the-art performance in this task. Although encouraging, this result should be interpreted with caution since our model has substantially more parameters compared to related systems. We could compare different models using the same number of total parameters. However, this would inevitably introduce other biases, e.g., the number of hyper-parameters would become different.

6 Conclusions

In this paper we proposed a machine reading simulator to address the limitations of recurrent neural networks when processing inherently structured input. Our model is based on a Long Short-Term Memory architecture embedded with a memory network, explicitly storing contextual representations of input tokens without recursively compressing them. More importantly, an intra-attention mechanism is employed for memory addressing, as a way to induce undirected relations among tokens. The attention layer is not optimized with a direct supervision signal but with the entire network in downstream tasks. Experimental results across three tasks show that our model yields performance comparable or superior to state of the art.

Although our experiments focused on LSTMs, the idea of building more structure aware neural models is general and can be applied to other types of networks. When direct supervision is provided, similar architectures can be adapted to tasks such as dependency parsing and relation extraction. In the future, we hope to develop more linguistically plausible neural architectures able to reason over nested structures and neural models that learn to discover compositionality with weak or indirect supervision.

Acknowledgments

We thank members of the ILCC at the School of Informatics and the anonymous reviewers for helpful comments. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 NAACL: HLT*, pages 1545–1554, San Diego, California.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2014 ICLR*, Banff, Alberta.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd ACL*, pages 655–665, Baltimore, Maryland.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 EMNLP*, pages 22–32, Lisbon, Portugal.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th ACL*, pages 1466–1477, Berlin, Germany.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 EMNLP*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd ICML*, pages 2067–2075, Lille, France.

Peter Clark, Phil Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 3rd Workshop on Automated KB Construction*, San Francisco, California.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Sreerupa Das, C. Lee Giles, and Guo zheng Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, pages 791–795. Morgan Kaufmann Publishers.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd ACL*, pages 334–343, Beijing, China.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd IJCAI*, pages 3–10, Barcelona, Spain.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 EMNLP*, pages 1535–1545, Edinburgh, Scotland, UK.

Fernanda Ferreira and John M. Henderson. 1991. Recovery from misanalyses of garden-path sentences. *Journal of Memory and Language*, 30:725–745.

Stefan L. Frank and Rens Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Pyschological Science*, 22(6):829–834.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1819–1827.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 EMNLP*, pages 1746–1751, Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 2015 ICLR*, San Diego, California.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd ACL*, pages 478–485, Barcelona, Spain.

Lars Konieczny. 2000. Locality and parsing complexity. *Journal of Psycholinguistics*, 29(6):627–645.

Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. 2014. A clockwork RNN. In *Proceedings of the 31st ICML*, pages 1863–1871, Beijing, China.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd ICML*, New York, NY.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st ICML*, pages 1188–1196, Beijing, China.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 EMNLP*, pages 1565–1575, Lisbon, Portugal.

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. In *Proceedings of ICLR-Workshop 2016*, San Juan, Puerto Rico.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of 11th Interspeech*, pages 1045–1048, Makuhari, Japan.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2015. Learning longer memory in recurrent neural networks. In *Proceedings of ICLR Workshop*, San Diego, California.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th ICML*, pages 1310–1318, Atlanta, Georgia.

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

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 296–305, Uppsala.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 2016 ICLR*, San Juan, Puerto Rico.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013a. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 EMNLP*, pages 1631–1642, Seattle, Washington.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 EMNLP*, pages 1631–1642, Seattle, Washingtton.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd ACL*, pages 1556–1566, Beijing, China.

Michael K. Tanenhaus, Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julue C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. In *Proceedings of the 15th NAACL*, San Diego, CA.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 NAACL: HLT*, pages 1442–1451, San Diego, California.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 2015 ICLR*, San Diego, USA.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd ICML*, New York, NY.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

On Generating Characteristic-rich Question Sets for QA Evaluation

Yu Su[1], **Huan Sun**[2], **Brian Sadler**[3], **Mudhakar Srivatsa**[4]
Izzeddin Gür[1], **Zenghui Yan**[1] **and Xifeng Yan**[1]
[1]University of California, Santa Barbara, Department of Computer Science
[2]The Ohio State University, Department of Computer Science and Engineering
[3]U.S. Army Research Lab, [4]IBM Research
{ysu,izzeddingur,zyan,xyan}@cs.ucsb.edu, sun.397@osu.edu
brian.m.sadler6.civ@mail.mil, msrivats@us.ibm.com

Abstract

We present a semi-automated framework for constructing factoid question answering (QA) datasets, where an array of question characteristics are formalized, including structure complexity, function, commonness, answer cardinality, and paraphrasing. Instead of collecting questions and manually characterizing them, we employ a reverse procedure, first generating a kind of graph-structured logical forms from a knowledge base, and then converting them into questions. Our work is the first to generate questions with explicitly specified characteristics for QA evaluation. We construct a new QA dataset with over 5,000 logical form-question pairs, associated with answers from the knowledge base, and show that datasets constructed in this way enable fine-grained analyses of QA systems. The dataset can be found in https://github.com/ysu1989/GraphQuestions.

1 Introduction

Factoid question answering (QA) has gained great attention recently, owing to the fast growth of large knowledge bases (KBs) such as DBpedia (Lehmann et al., 2014) and Freebase (Bollacker et al., 2008), which avail QA systems of comprehensive and precise knowledge of encyclopedic scope (Yahya et al., 2012; Berant et al., 2013; Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant and Liang, 2014; Fader et al., 2014; Reddy et al., 2014; Bao et al., 2014; Zou et al., 2014; Yao and Van Durme, 2014; Yih et al., 2015; Sun et al., 2015; Dong et al., 2015; Yao, 2015; Berant and Liang, 2015). With the blossoming of QA systems, evaluation is becoming an increasingly important problem. QA datasets, consisting of a set of questions with ground-truth answers, are critical for both comparing existing systems and gaining insights to develop new systems.

Questions have rich *characteristics*, constituting dimensions along which question difficulty varies. Some questions are difficult due to their complex semantic structure (*"Who was the coach when Michael Jordan stopped playing for the Chicago Bulls?"*), while some others may be difficult because they require a precise quantitative analysis over the answer space (*"What is the best-selling smartphone in 2015?"*). Many other characteristics shall be considered too, e.g., what topic a question is about (questions about common topics may be easier to answer) and how many answers there are (it is harder to achieve a high recall in case of multiple answers). Worse still, due to the flexibility of natural language, different people often describe the same question in different ways, i.e., paraphrasing. It is important for a QA system to be robust to paraphrasing.

A QA dataset explicitly specifying such question characteristics allows for fine-grained inspection of system performance. However, to the best of our knowledge, none of the existing QA datasets (Voorhees and Tice, 2000; Berant et al., 2013; Cai and Yates, 2013; Lopez et al., 2013; Bordes et al., 2015; Serban et al., 2016) provides question characteristics. In this work, we make the first attempt to generate questions with explicitly specified characteristics, and examine the impact of various question characteristics in QA.

We present a semi-automated framework (Figure 1) to construct QA datasets with characteristic

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 562–572,
Austin, Texas, November 1-5, 2016.

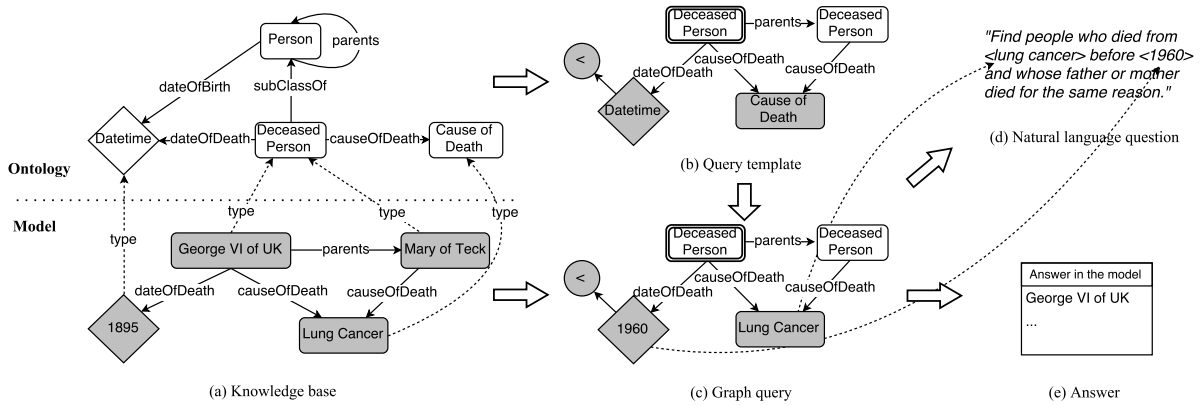


Figure 1: Running example of our framework. Graph queries are first generated from a knowledge base. After refinement (not shown), graph queries are sent to human annotators and converted into natural language questions. Answers are collected from the knowledge base.

specification from a knowledge base. The framework revolves around an intermediate *graph query* representation, which helps to formalize question characteristics and collect answers. We first automatically generate graph queries from a knowledge base, and then employ human annotators to convert graph queries into questions.

Automating graph query generation brings with it the challenge of assessing the quality of graph queries and filtering out bad ones. Our framework tackles the challenge by combining structured information in the knowledge base and statistical information from the Web. First, we identify *redundant components* in a graph query and develop techniques to remove them. Furthermore, based on the frequency of entities, classes, and relations mined from the Web, we quantify the *commonness* of a graph query and filter out too rare ones.

We employ a semi-automated approach for the conversion from graph query to natural language question, which provides two levels of paraphrasing: Common lexical forms of an entity (e.g., "*Queen Elizabeth*" and "*Her Majesty the Queen*" for `ElizabethII`) mined from the Web are used as entity paraphrases, and the remaining parts of a question are paraphrased by annotators. As a result, dozens of paraphrased questions can be produced for a single graph query.

To demonstrate the usefulness of question characteristics in QA evaluation, we construct a new dataset with over 5,000 questions based on Freebase using the proposed framework, and extensively evaluate several QA systems. A couple of new findings about system performance and question difficulty are discussed. For example, different from the results based on previous QA datasets (Yao et al., 2014), we find that semantic parsing in general works better than information extraction on our dataset. Information extraction based QA systems have trouble dealing with questions requiring aggregation or with multiple answers. A holistic understanding of the whole question is often needed for hard questions. The experiments point out an array of issues that future QA systems may need to solve.

2 Related Work

Early QA research has extensively studied problems like question taxonomy, answer type, and knowledge sources (Burger et al., 2001; Hirschman and Gaizauskas, 2001; Voorhees and Tice, 2000). This work mainly targets factoid questions with one or more answers that are guaranteed to exist in a KB.

A few KB-based QA datasets have been proposed recently. QALD (Lopez et al., 2013) and FREE917 (Cai and Yates, 2013) contain hundreds of hand-crafted questions. QALD also indicates whether a question requires aggregation. Both based on single Freebase triples, SIMPLEQUESTIONS (Bordes et al., 2015) employ human annotators to formulate questions, while Serban et al. (2016) use a recurrent neural network to automatically formulate questions. They are featured by a large size, but the questions only concern single triples, while our framework can generate ques-

tions involving multiple triples and various functions. Wang et al. (2015) generate question-answer pairs for closed domains like basketball. They also first generate logical forms (λ-DCS formulae (Liang, 2013) in their case), and then convert logical forms into questions via crowdsourcing. Logical forms are first converted into canonical questions to help crowdsourcing workers. Different from previous works, we put a particular focus on generating questions with diversified characteristics in a systematic way, and examining the impact of different question characteristics in QA.

Another attractive way for QA dataset construction is to collect questions from search engine logs (Bendersky and Croft, 2009). For example, WEBQUESTIONS (Berant et al., 2013) contains thousands of popular questions from Google search, and Yih et al. (2016) have manually annotated these questions with logical forms. However, automatic characterization of questions is hard, while manual characterization is costly and requires expertise. Moreover, users' search behavior is shaped by search engines (Aula et al., 2010). Due to the inadequacy of current search engines to answer advanced questions, users may adapt themselves accordingly and mostly ask simple questions. Thus questions collected in this way, to some extent, may still not well reflect the true distribution of user information needs, nor does it fully exploit the potential of KB-based QA. Collecting answers is yet another challenge for this approach. Yih et al. (2016) show that only 66% of the WEBQUESTIONS answers, which were collected via crowdsourcing, are completely correct. On the other hand, although questions generated from a KB may not follow the distribution of user information needs, it has the advantage of explicit question characteristics, and enables programmatic configuration of question generation. Also, answer collecting is automated without involving human labor and errors.

3 Background

3.1 Knowledge Base

In this work, we mainly concern knowledge bases storing knowledge about entities and relations in the form of triples (simply *knowledge bases* hereafter). Suppose $\mathcal{E}$ is a set of entities, $\mathcal{L}$ a set of literals ($\mathcal{I} = \mathcal{E} \cup \mathcal{L}$ is also called *individuals*), $\mathcal{C}$ a set of classes, and $\mathcal{R}$ a set of directed relations, a knowledge base $\mathcal{K}$ consists of two parts: an *ontology* $\mathcal{O} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$ and a *model* $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{L})$. In other words, an ontology specifies classes and relations between classes, and a model consists of *facts* about individuals. Such knowledge bases can be naturally represented as a directed graph, e.g., Figure 1(a). Literal classes such as `Datetime` are represented as diamonds, and other classes are rounded rectangles. Individuals are shaded. We assume relations are typed, i.e., each relation is associated with a set of *domain* and *range* classes. Facts of a relation must be compatible with its domain and range constraints. Without loss of generality, we use Freebase (June 2013 version) in this work for compatibility with the to-be-tested QA systems. It has 24K classes, 65K relations, 41M entities, and 596M facts.

3.2 Graph Query

Motivated by the graph-structured nature of knowledge bases, we adopt a graph-centric approach. We hinge on a formal representation named *graph query* (e.g., Figure 1(c)), developed on the basis of Yih et al. (2015) and influenced by λ-DCS (Liang, 2013).

Syntax. A graph query q is a connected directed graph built on a given knowledge base $\mathcal{K}$. It comprises three kinds of nodes: (1) *Question node* (double rounded rectangle), a free variable. (2) *Ungrounded node* (rounded rectangle or diamond), an existentially quantified variable. (3) *Grounded node* (shaded rounded rectangle or diamond), an individual. In addition, there are *functions* (shaded circle) such as $<$ and `count` applied on a node. Nodes are typed, each associated with a class. Nodes are connected by directed edges representing relations. Entities on the grounded nodes are called *topic entities*.

Semantics. Graph query is a strict subset of λ-calculus. For example, the graph query in Figure 1(c) can be written in λ-calculus (an existentially quantified variable is imposed by $<$):

$$\begin{aligned}\lambda x.\exists y.\exists z.&\texttt{type}(x, \texttt{DeceasedPerson})\\ &\wedge \texttt{type}(y, \texttt{DeceasedPerson})\\ &\wedge \texttt{type}(z, \texttt{Datetime}) \wedge \texttt{parents}(x, y)\\ &\wedge \texttt{causeOfDeath}(x, \texttt{LungCancer})\\ &\wedge \texttt{causeOfDeath}(y, \texttt{LungCancer})\\ &\wedge \texttt{dateOfDeath}(x, z) \wedge <(z, 1960).\end{aligned}$$

The *answer* of a graph query q, denoted as $[\![q]\!]_{\mathcal{K}}$, can be easily obtained from $\mathcal{K}$. For example, if $\mathcal{K}$ is stored in a RDF triplestore, then q can be automatically converted into a SPARQL query and run against $\mathcal{K}$ to get the answer. Compared with Yih et al. (2015), graph queries are not constrained to be tree-structured, which grants us a higher expressivity. For example, linguistic phenomena like anaphora (e.g., Figure 1(d)) become easier to model.

4 Automatic Graph Query Generation

Our framework proceeds as follows: (1) Generate *query templates* from a knowledge base, ground the templates to generate graph queries, and collect answers (this section). (2) Refine graph queries to retain high-quality ones (Section 5). (3) Convert graph queries into questions via crowdsourcing (Section 6).

We now describe an algorithm to generate the query template shown in Figure 1(b) (excluding the function for now). For simplicity, we will focus on the case of a single question node. Nevertheless, the proposed framework can be extended to generate graph queries with multiple question nodes. The algorithm takes as input an ontology (Figure 1(a)) and the desired number of edges. All the operations are conducted in a random manner to avoid systematic biases in query generation. The `DeceasedPerson` class is first selected as the question node. We then iteratively grow it by adding neighboring nodes and edges in the ontology. In each iteration, an existing node is selected, and a new edge, which might introduce a new node, is appended to it. For example, the relation `causeOfDeath`, whose domain includes `DeceasedPerson`, is first appended to the question node, and then one of its range classes, `CauseOfDeath`, is added as a new node. When a node with the class `CauseOfDeath` already exists, it is possible to add an edge without introducing a new node. The same relation or class can be added multiple times, e.g., "*parent of parent*".

Topic entities like `LungCancer` play an important role in a question. A query template contains some template nodes that can be grounded with different topic entities to generate different graph queries. We randomly choose a few nodes as template. It may cause problems. For example, grounding one node may make some others redundant. We conduct a formal study on this in Section 5.1.

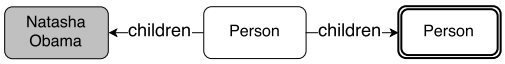


Figure 2: Mutual exclusivity example. Entities on different nodes should be different.

Functions such as counting and comparatives are pervasive in real-life questions, e.g., "*how many*", "*the most recent*", and "*people older than*", but are scarce in existing QA datasets. We incorporate functions as an important question characteristic, and consider nine common functions, grouped into three categories: counting (`count`), superlative (`max`, `min`, `argmax`, `argmin`), and comparative ($>$, $\geq$, $<$, $\leq$). More functions can be incorporated in the future. See Appendix A for examples. We randomly add functions to compatible nodes in query templates. In the running example, the $<$ function imposes the constraint that only people who passed away before a certain date should be considered. Each query will have at most one function.

We then ground the template nodes with individuals to generate graph queries. A grounding is valid if the individuals conform with the class of the corresponding template nodes, and the resulted answer is not empty. For example, by grounding `CauseOfDeath` with `LungCancer` and `Datetime` with `1960`, we get the graph query in Figure 1(c). A query template can render multiple groundings.

Finally, we convert a graph query into a SPARQL query and execute it using Virtuoso Open-Source 7 to collect answers. We further impose *mutual exclusivity* in SPARQL queries, that is, the entities on any two nodes in a graph query should be different. Consider the example in Figure 2, which is asking for the siblings of Natasha Obama. Wihout mutual exclusivity, however, Natash Obama herself will also be included as an answer, which is not desired.

5 Query Refinement

Since graph queries are randomly generated, some of them may not correspond to an interesting question. Next we study two query characteristics, *redundancy* and *commonness*, based on which we provide mechanisms for automatic query refinement.

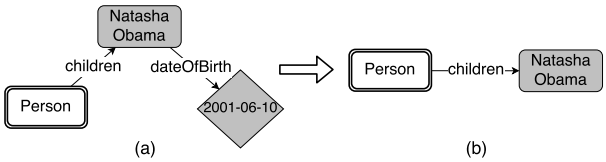


Figure 3: Query minimization example: (a) Graph query with redundant components. (b) Graph query after minimization.

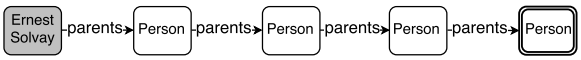


Figure 4: Uncommon query example. It is uncommon to ask for somebody's great-great-grandparents.

5.1 Query Redundancy and Minimization

Some components (nodes and edges) in a graph query may not effectively impose any constraint on the answer. The query in Figure 3(a) is to "*find the US president whose child is Natasha Obama, **and Natasha Obama was born on 2001-06-10***". Intuitively, the bold-faced clause does not change the answer of the question. Correspondingly, the `dateOfBirth` edge and the date node are redundant. As a comparison, removing any component from the query in Figures 3(b) will change the answer. Formally, given a knowledge base $\mathcal{K}$, a component in a graph query q is *redundant* iff. removing it does not change the answer $[\![q]\!]_{\mathcal{K}}$.

Redundancy can be desired or not. In a question, redundant information may be inserted to reduce ambiguity. In Figure 3(a), if one uses "*Natasha*" to refer to `NatashaObama`, there comes ambiguity since it may be matched with many other entities. The additional information "*who was born on 2001-06-10*" then helps. Next we describe an algorithm to remove redundancy from queries. One can choose to either only generate queries with no redundant component, or intentionally generate redundant queries and test QA systems in presence of redundancy.

We manage to generate *minimal* queries, for which there exists no sub-query having the same answer. An important theorem, as we prove in Appendix B, is the equivalency of minimality and non-redundancy: A query is minimal *iff.* it has no redundant component. This renders a simple algorithm for query minimization, which directly detects and removes the redundant components in a query. We first examine every edge (in an arbitrary order), and remove an edge if it is redundant. Redundant nodes will then become disconnected to the question node and are thus eliminated. It is easy to prove that the produced query (e.g., Figure 3(b)) is minimal, and has the same answer as the original query.

5.2 Commonness Checking

We now quantify the *commonness* of graph queries. The benefits of this study are two-fold. First, it provides a refinement mechanism to reduce too rare queries. Second, commonness is itself an important question characteristic. It is interesting to examine its impact on question difficulty. Consider the example in Figure 4, which asks for "*the great-great-grandparents of Ernest Solvay*". It is minimal and logically plausible. Few users, however, are likely to come up with it. Ernest Solvay is famous for the Solvay Conferences, but few people outside the science community may know him. Although `Person` and `parents` are common, asking for the great-great-grandparents is quite uncommon.

A query is more common if users would more likely come up with it. We define the commonness of a query q as its *probability* $p(q)$ of being picked among all possible queries from a knowledge base. The problem then boils down to estimating $p(q)$. It is hard, if not impossible, to exhaust the whole query space. We thus make the following simplification. We break down query commonness by components, assuming mutual independence between components, and omit functions:

$$p(q) = \prod_{i \in \mathcal{I}_q} p(i) \times \prod_{c \in \mathcal{C}_q} p(c) \times \prod_{r \in \mathcal{R}_q} p(r), \qquad (1)$$

where $\mathcal{I}_q, \mathcal{C}_q, \mathcal{R}_q$ are the *multi-set* of the individuals, classes, and relations in q, respectively. Repeating components are thus accumulated (c.f. Figure 4).

We propose a data-driven method, using statistical information from the Web, to estimate $p(i)$, $p(c)$, and $p(r)$. Other methods like domain-knowledge based estimation are also applicable if available. We start with entity probability $p(e)$ (excluding literals for now). If users mention an entity more frequently, its probability of being observed in a question should be higher. We use a large entity linking dataset, FACC1 (Gabrilovich et al., 2013), which identifies around 10 billion mentions of Freebase entities in over 1 billion web documents. The estimated linking precision and recall are 80-85% and 70-85%, re-

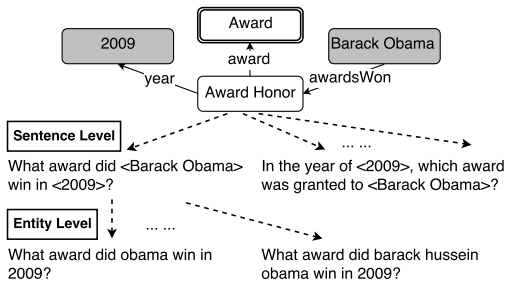


Figure 5: Question generation and paraphrasing.

spectively. Suppose entity e has $n(e)$ mentions, then $p(e) = \frac{n(e)}{\sum_{e' \in \mathcal{E}} n(e')}$. For a class c, probability $p(c)$ is higher if it has more frequently mentioned entities. If we use $e \in c$ to indicate e is an instance of c, then $p(c) = \frac{\sum_{e \in c} n(e)}{\sum_{c' \in \mathcal{C}} \sum_{e \in c'} n(e)}$. Estimating $p(r)$ requires relation extraction from texts, which is hard. We make the following simplification: If (e_1, r, e_2) is a fact in the knowledge base, we increase $n(r)$ by 1 if e_1 and e_2 co-occur in a document. This suffices to distinguish common relations from uncommon ones. We then define $p(r) = \frac{n(r)}{\sum_{r' \in \mathcal{R}} n(r')}$. Finally, we use frequency information from the knowledge base to smooth the probabilities, e.g., to avoid zero probabilities. The probability of literals are solely determined by the frequency information from the knowledge base. Refer to Appendix C for the resulted probability distributions.

6 Natural Language Conversion

In order to ensure naturalness and diversity, we employ human annotators to manually convert graph queries into natural language questions. We manage to provide two levels of paraphrasing (Figure 5). Each query is sent to multiple annotators for sentence-level paraphrasing. In addition, we use different lexical forms of an entity mined from FACC1 for entity-level paraphrasing. We provide a ranked list of common lexical forms and the corresponding frequency for each topic entity. For example, the lexical form list for `UnitedStatesOfAmerica` is "*us*" (108M), "*united states*" (44M), "*usa*" (22M), etc. Finally, graph queries are automatically translated into SPARQL queries to collect answers.

Natural language generation (NLG) (Wen et al., 2015; Serban et al., 2016; Dušek and Jurčíček, 2015) would be a good complement to our framework, the combination of which can lead to a fully-automated pipeline to generate QA datasets. For example, Serban et al. (2016) automatically convert Freebase triples into questions with a neural network. More sophisticated NLG techniques able to handle graph queries involving multiple relations and various functions are an interesting future direction.

7 Experiments

We have constructed a new QA dataset, named GRAPHQUESTIONS, using the proposed framework, and tested several QA systems to show that it enables fine-grained inspection of QA systems.

7.1 Dataset Construction

We first randomly generated a set of minimal graph queries, and removed the ones whose commonness is below a certain threshold. The remaining graph queries were then screened by graduate students, and a *canonical* question was generated for each query, with each being verified by at least two students. We recruited 160 crowdsourcing workers from Amazon MTurk to generate sentence-level paraphrases of the canonical questions. Trivial paraphrases (e.g., "*which city*" vs. "*what city*") were manually removed to retain a high diversity in paraphrasing. At most 3 entity-level paraphrases were used for each sentence-level paraphrase.

7.2 Dataset Analysis

GRAPHQUESTIONS contains 500 graph queries, 2,460 sentence-level paraphrases, and 5,166 questions[2]. The dataset presents a high diversity and covers a wide range of domains including `People`, `Astronomy`, `Medicine`, etc. Specifically, it contains 148, 506, 596, 376 and 3,026 distinct domains, classes, relations, topic entities, and words, respectively. We evenly split GRAPHQUESTIONS into a training set and a testing set. All the paraphrases of the same graph query are in the same set.

While there are other question characteristics derivable from graph query, we will focus on the following ones: *structure complexity*, *function*, *commonness*, *paraphrasing*, and *answer cardinality*. We

[2]For each query template, we only generate one graph query, but one can also generate multiple graph queries, and easily get the corresponding questions by replacing the topic entities. This will significantly increase the total number of questions, and can be helpful in training.

	# of edges			Function				$\log_{10}(p(q))$				$\|\mathbf{A}\|$	
	1	2	3	none	count	super.	comp.	$[-40, 30)$	$[-30, 20)$	$[-20, 10)$	$[-10, 0)$	1	> 1
# of graph queries	321	144	35	350	61	42	47	60	135	283	22	332	168
# of questions	3094	1648	424	3855	710	332	269	653	1477	2766	270	3487	1679

Table 1: Characteristic statistics. $|\mathbf{A}|$ is answer cardinality. Refer to Appendix D for paraphrase and other fine-grained distributions.

Question	Domain	Answer	# of edges	Function	$\log_{10}(p(q))$	$\|\mathbf{A}\|$
Find terrorist organizations involved in **September 11 attacks**.	Terrorism	alQaeda	1	none	-16.67	1
The **September 11 attacks** were carried out with the involvement of what terrorist organizations?						
Who did **nine eleven**?						
How many children of **Eddard Stark** were born in **Winterfell**?	Fictional Universe	3	2	count	-23.34	1
Winterfell is the home of how many of **Eddard Stark**'s children?						
What's the number of **Ned Stark**'s children whose birthplace is **Winterfell**?						
In which month does the average rainfall of **New York City** exceed **86** mm?	Travel	March, August ...	3	comp.	-37.84	7
Rainfall averages more than **86** mm in **New York City** during which months?						
List the calendar months when **NYC** averages in excess of **86** millimeters of rain?						

Table 2: Example questions and characteristics. Three sentence-level paraphrases are shown for each graph query, with the last one also involving entity-level paraphrasing. Topic entities are bold-faced. More examples can be found in Appendix D.

use the number of edges to quantify structure complexity, and limit it to at most 3. Commonness is limited to $\log_{10}(p(q)) \geq -40$ (c.f. Eq. 1). As shown in Section 7.4.2, such questions are already very hard for existing QA systems. Nevertheless, the proposed framework can be used to generate questions with different characteristic distributions. Some statistics are shown in Table 1 and more fine-grained statistics can be found in Appendix D.

Several example questions are shown in Table 2. Sentence-level paraphrasing requires to handle both commands (the first example) and "*Wh*" questions, light verbs ("*Who did nine eleven?*"), and changes of syntactic structure ("*The September 11 attacks were carried out with the involvement of what terrorist organizations?*"). Entity-level paraphrasing tests the capability of QA systems on abbreviation ("*NYC*" for `New York City`), world knowledge ("*Her Majesty the Queen*" for `ElizabethII`), or even common typos ("*Shakspeare*" for `WilliamShakespeare`). Numbers and dates are also common, e.g., "*Which computer operating system was released on Sept. the 20th, 2008?*"

We compare several QA datasets constructed from Freebase, shown in Table 3. Datasets focusing on single-relation questions are of a larger scale, but are also of a significant lack in question characteristics. Overall GRAPHQUESTIONS presents the highest diversity in question characteristics.

7.3 Setup

We evaluate three QA systems whose source code is publicly available: SEMPRE (Berant et al., 2013), PARASEMPRE (Berant and Liang, 2014), and JACANA (Yao and Van Durme, 2014). SEMPRE and PARASEMPRE follow the semantic parsing paradigm. SEMPRE conducts a bottom-up beam-based parsing on questions to find the best logical form. PARASEMPRE, in a reverse manner, enumerates a set of logical forms, generates a canonical utterance for each logical form, and ranks logical forms according to how well the canonical utterance paraphrases the input question. In contrast, JACANA follows the information extraction paradigm, and builds a classifier to directly predict whether an individual is the answer. They all use Freebase.

The main metric for answer quality is the average F1 score, following Berant and Liang (2014). Because a question can have more than one answer, individual precision, recall, and F1 scores are first computed on each question and then averaged. When a system generates no response for a question,

Dataset	# of Questions	# of Multi-relation	Function (count/super./comp.)	Commonness	Paraphrase	Multi-answer
GRAPHQUESTIONS (this work)	5166	2072	710 / 332 / 269	+	+	+
WEBQUESTIONSSP (Yih et al., 2016)[1]	4737	2075	2 / 168 / 334	-	-	+
FREE917 (Cai and Yates, 2013)	917	229	185 / 0 / 0	-	-	+
Serban et al. (2016)	30M	0	0 / 0 / 0	-	-	-
SIMPLEQUESTIONS (Bordes et al., 2015)	108K	0	0 / 0 / 0	-	-	-

Table 3: Comparison of QA datasets constructed from Freebase. GRAPHQUESTIONS is the richest in question characteristics.

System	F1	Time/s
SEMPRE	10.80	56.19
PARASEMPRE	12.79	18.43
JACANA	5.08	2.01

Table 4: Overall performance on GRAPHQUESTIONS.

precision is 1, recall is 0, and F1 is 0. Average runtime is used for efficiency. Results are shown in percentage. Systems are trained on the training set using the suggested configurations (Appendix E). We use student's t test at $p = 0.05$ for significance test.

7.4 Results

7.4.1 Overall Evaluation

Compared with the scores on WEBQUESTIONS (30%-40%), the scores on GRAPHQUESTIONS are lower (Table 4). This is because GRAPHQUESTIONS contains questions over a broader range of difficulty levels. For example, it is more diverse in topics (Appendix D); also the scores become much closer when excluding paraphrasing (Section 7.4.2).

JACANA achieves a comparable F1 score with SEMPRE and PARASEMPRE on WEBQUESTIONS (Yao et al., 2014). On GRAPHQUESTIONS, however, SEMPRE and PARASEMPRE significantly outperform JACANA (both $p < 0.0001$). The following experiments will give more insights about where the performance difference comes from. On the other hand, JACANA is much faster, showing an advantage of information extraction. The semantic parsing systems spend a lot of time on executing SPARQL queries. Bypassing SPARQL and directly working on the knowledge base may be a promising way to speed up semantic parsing on large knowledge bases (Yih et al., 2015).

[2]WEBQUESTIONSSP is WEBQUESTIONS with manually annotated logical forms. Only those with a full logical form are included (4737 / 5810).

7.4.2 Fine-grained Evaluation

With explicitly specified question characteristics, we are able to further inspect QA systems.

Structure Complexity. We first break down system performance by structure. Answer quality is in general sensitive to the complexity of question structure: As the number of edges increases, F1 score decreases (Figure 6(a)). The tested systems often fail to take into account auxiliary constraints in a question. For example, for "*How many children of Ned Stark were born in Winterfell?*" SEMPRE fails to identify the constraint "*born in Winterfell*", so it also considers Ned Stark's bastard son, Jon Snow, as an answer, who was not born in Winterfell. Answering questions involving multiple relations using large knowledge bases remain an open problem. The large size of knowledge bases prohibits exhaustive search, so smarter algorithms are needed to efficiently prune the answer space. Berant and Liang (2015) point out an interesting direction, leveraging agenda-based parsing with imitation learning for efficient search in the answer space.

Function. In terms of functions, while SEMPRE and PARASEMPRE perform well on `count` questions, all the tested systems perform poorly on questions with superlatives or comparatives (Figure 6(b)). JACANA has trouble dealing with functions because it does not conduct quantitative analysis over the answer space. SEMPRE and PARASEMPRE do not generate logical forms with superlatives and comparatives, so they cannot answer such questions well.

Commonness. Not surprisingly, more common questions are in general easier to answer (Figure 6(c)). An interesting observation is that SEMPRE's performance gets worse on the most common questions. The cause is likely rooted in how the QA systems construct their candidate answer sets. PARASEMPRE and JACANA exhaustively construct candidate sets, while SEMPRE employs a bottom-up beam search, making it more sensitive to the size of

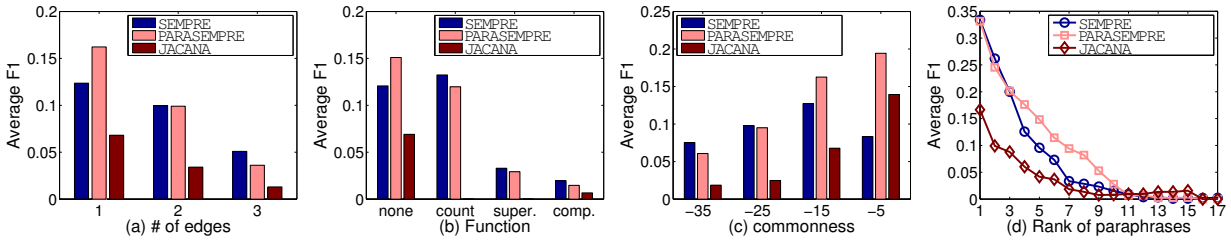


Figure 6: Performance breakdown by (a) structure complexity, (b) function, (c) commonness, and (d) paraphrase. Note that in (c) $x = -5$ indicates the commonness range $-10 \leq \log_{10}(p(q)) < 0$.

the candidate answer space. Common entities like `UnitedStatesOfAmerica` are often featured by a high degree in knowledge bases (e.g., 1 million neighboring entities), which dramatically increases the size of the candidate answer space. During SEMPRE's iterative beam search, many correct logical forms may have fallen off beam before getting into the final candidate set. We checked the percentage of questions for which the correct logical form is in the final candidate set, and found that it decreased from 19.8% to 16.7% when commonness increased from -15 to -5, providing an evidence for the intuition.

Paraphrasing. It is critical for a system to tolerate the wording varieties of users. We make the first effort to evaluate QA systems on paraphrasing. For each system, we rank, in descending order, all the paraphrases derived from the same graph query by their F1 score achieved by the system, and then compute the average F1 score of each rank. In Figure 6(d), the decreasing rate of a curve thus describes a system's robustness to paraphrasing; the higher, the less robust. All the systems achieve a reasonable score on the top-1 paraphrases, i.e., when a system can choose the paraphrase it can best answer. The F1 scores drop quickly in general. On the fourth-ranked paraphrases, the F1 score of SEMPRE, PARASEMPRE, and JACANA are respectively only 37.65%, 53.2%, and 36.2% of their score on the top-1 paraphrases. Leveraging paraphrasing in its model, PARASEMPRE does seem to be more robust. The results show that how to handle paraphrased questions is still a challenging problem.

Answer Cardinality. SEMPRE and JACANA get a significantly lower F1 score (both $p < 0.0001$) on multi-answer questions (Table 5), mainly coming from a decrease on recall. The decrease of PARASEMPRE is not significant (p=0.29). The particularly significant decrease of JACANA demonstrates the difficulty of training a classifier that can predict all of the answers correctly; semantic parsing is more robust in this case. The precision of SEMPRE is high because it generates no response for many questions. Note that under the current definition, the average F1 score is not the harmonic mean of the average precision and recall (c.f. Section 7.3).

System	$\|\mathbf{A}\|$	Prec.	Rec.	F1
SEMPRE	1	59.81	16.11	12.68
	> 1	62.38	9.17	6.78
PARASEMPRE	1	17.42	17.58	13.25
	> 1	19.65	17.23	11.82
JACANA	1	14.77	6.56	6.56
	> 1	11.80	1.43	1.98

Table 5: Performance breakdown by answer cardinality $|\mathbf{A}|$.

8 Conclusion

We proposed a framework to generate characteristic-rich questions for question answering (QA) evaluation. Using the proposed framework, we constructed a new and challenging QA dataset, and extensively evaluated several QA systems. The findings point out an array of issues that future QA research may need to solve.

9 Acknowledgements

This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053, NSF IIS 0954125, and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Anne Aula, Rehan M. Khan, and Zhiwei Guan. 2010. How does search behavior change as search becomes more difficult? In *Proceedings of CHI*.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of ACL*.

Michael Bendersky and W. Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Proceedings of the 2009 workshop on Web Search Click Data*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, et al. 2001. Issues, tasks and program structures to roadmap research in question & answering (q&a). In *Document Understanding Conferences Roadmapping Documents*, pages 1–35.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of ACL*.

Ondřej Dušek and Filip Jurčíček. 2015. Training a natural language generator from unaligned data. In *Proceedings of ACL*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of KDD*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). `http://lemurproject.org/clueweb09/` and `http://lemurproject.org/clueweb12/`.

Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of ACL*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of WWW*.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of SIGIR*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Deep answers for naturally asked questions on the web of data. In *Proceedings of WWW*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of ACL*.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *Proceedings of ACL*.

Xuchen Yao. 2015. Lean question answering over Freebase from scratch. In *Proceedings of NAACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of SIGMOD*.

Learning to Translate for Multilingual Question Answering

Ferhan Ture
Comcast Labs*
1110 Vermont Ave NW Ste 600
Washington, DC, 20005 USA
ferhan_ture@cable.comcast.com

Elizabeth Boschee
Raytheon BBN Technologies
10 Moulton St
Cambridge, MA, 02138 USA
eboschee@bbn.com

Abstract

In multilingual question answering, either the question needs to be translated into the document language, or vice versa. In addition to *direction*, there are multiple *methods* to perform the translation, four of which we explore in this paper: word-based, 10-best, context-based, and grammar-based. We build a feature for each combination of translation *direction* and *method*, and train a model that learns optimal feature weights. On a large forum dataset consisting of posts in English, Arabic, and Chinese, our novel *learn-to-translate* approach was more effective than a strong baseline ($p < 0.05$): translating all text into English, then training a classifier based only on English (original or translated) text.

1 Introduction

Question answering (QA) is a specific form of the information retrieval (IR) task, where the goal is to find relevant well-formed answers to a posed question. Most QA pipelines consist of three main stages: (a) preprocessing the question and collection, (b) retrieval of candidate answers in the collection, and (c) ranking answers with respect to their relevance to the question and return the top N answers. The types of questions can range from factoid (e.g., "What is the capital of France?") to causal (e.g., "Why are trees green?"), and opinion questions (e.g., "Should USA lower the drinking age?").

The most common approach to *multilingual QA* (MLQA) has been to translate all content into its most probable English translation via machine translation (MT) systems. This strong baseline, which we refer to as *one-best MT* (`1MT`), has been successful in prior work (Hartrumpf et al., 2009; Lin and Kuo, 2010; Shima and Mitamura, 2010). However, recent advances in cross-lingual IR (CLIR) show that one can do better by representing the translation space as a probability distribution (Ture and Lin, 2014). In addition, MT systems perform substantially worse with user-generated text, such as web forums (Van der Wees et al., 2015), which provide extra motivation to consider alternative translation approaches for higher recall. To our knowledge, it has yet to be shown whether these recent advancements in CLIR transfer to MLQA.

We introduce a novel answer ranking approach for MLQA (i.e., *Learning to Translate* or `L2T`), a model that learns the optimal translation of question and/or candidate answer, based on how well it discriminates between good and bad answers. We achieve this by introducing a set of features that encapsulate lexical and semantic similarities between a question and a candidate answer through various translation strategies (Section 3.1). The model then learns feature weights for each combination of translation *direction* and *method*, through a discriminative training process (Section 3.2). Once a model is trained, it can be used for MLQA, by sorting each candidate answer in the collection by model score. Instead of learning a single model to score candidate answers in any language, it might be meaningful to train a separate model that can learn to discriminate between good and bad answers in each language. This can let each model learn *feature weights custom to*

*This work was completed while author was an employee of Raytheon BBN Technologies.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 573–584,
Austin, Texas, November 1-5, 2016.

the language, therefore allowing a more fine-grained ranking (Section 3.4). We call this alternative approach *Learning to Custom Translate* (L2CT).

Experiments on the DARPA Broad Operational Language Technologies (BOLT) IR task[1] confirm that L2T yields statistically significant improvements over a strong baseline ($p < 0.05$), in three out of four experiments. L2CT outperformed the baseline as well, but was not more effective than L2T.

2 Related Work

For the last decade or so, research in QA has mostly been driven by annual evaluation campaigns like TREC,[2] CLEF,[3] and NTCIR.[4] Most earlier work relied on either rule-based approaches where a set of rules were manually crafted for each type of question, or IR-like approaches where each pair of question and candidate answer was scored using retrieval functions (e.g., BM25 (Robertson et al., 2004)). On the other hand, training a classifier for ranking candidate answers allows the exploitation of various features extracted from the question, candidate answer, and surrounding context (Madnani et al., 2007; Zhang et al., 2007). In fact, an explicit comparison at 2007 TREC confirmed the superiority of machine learning-based (ML-based) approaches (F-measure 35.9% vs 38.7%) (Zhang et al., 2007). Learning-to-rank approaches have also been applied to QA successfully (Agarwal et al., 2012).

Previous ML-based approaches have introduced useful features from many aspects of natural language, including lexical (Brill et al., 2001; Attardi et al., 2001), syntactic (Alfonseca et al., 2001; Katz et al., 2005), semantic (Cui et al., 2005; Katz et al., 2005; Alfonseca et al., 2001; Hovy et al., 2001), and discourse features, such as coreference resolution (Morton, 1999), or identifying temporal/spatial references (Saquete et al., 2005; Harabagiu and Bejan, 2005), which are especially useful for "why" and "how" questions (Kolomiyets and Moens, 2011). Additionally, semantic role labeling and dependency trees are other forms of semantic analysis used widely in NLP applications (Shen and Lapata, 2007; Cui et al., 2005).

When dealing with multilingual collections, most prior approaches translate all text into English beforehand, then treat the task as monolingual retrieval (previously referred to as 1MT). At recent evaluation campaigns like CLEF and NTCIR,[5] almost all teams simply obtained the one-best question translation, treating some online MT system as a black box (Adafre and van Genabith, 2009; Hartrumpf et al., 2009; Martinez-Gonzalez et al., 2009; Lin and Kuo, 2010; Shima and Mitamura, 2010), with few notable exceptions that took term importance (Ren et al., 2010), or semantics (Munoz-Terol et al., 2009) into account. Even for non-factoid MLQA, most prior work does not focus on the translation component (Luo et al., 2013; Chaturvedi et al., 2014).

Contributions. Ture and Lin described three methods for translating queries into the collection language in a probabilistic manner, improving *document retrieval* effectiveness over a one-best translation approach (2014). Extending this idea to MLQA appears as a logical next step, yet most prior work relies solely on the one-best translation of questions or answers (Ko et al., 2010b; García-Cumbreras et al., 2012; Chaturvedi et al., 2014), or selects the best translation out of few options (Sacaleanu et al., 2008; Mitamura et al., 2006). Mehdad et al. reported improvements by including the top ten translations (instead of the single best) and computing a distance-based entailment score with each (2010). While Espla-Gomis et al. argue that using MT as a black box is more convenient (and modular) (2012), there are potential benefits from a closer integration between statistical MT and multilingual retrieval (Nie, 2010). To the best of our knowledge, there is no prior work in the literature, where the ***optimal query and/or answer translation is learned via machine learning***. This is our main contribution, with which we outperform the state of the art.

In addition to learning the optimal translation, we ***learn the optimal subset of the training data for a given task***, where the criteria of whether we include a certain data instance is based on either the source language of the sentence, or the language in which the sentence was annotated. Training data selection strategies have not been studied extensively in the

[1] http://www.darpa.mil/Our_Work/I2O/Programs

[2] http://trec.nist.gov

[3] http://www.clef-initiative.eu

[4] http://research.nii.ac.jp/ntcir/index.html

[5] Most recent MLQA tracks were in 2008 (CLEF) and 2010 (NTCIR).

QA literature, therefore the effectiveness of our simple language-related criteria can provide useful insights to the community.

When there are multiple independent approaches for ranking question-answer pairs, it is required to perform a *post-retrieval merge*: each approach generates a ranked list of answers, which are then merged into a final ranked list. This type of system combination approach has been applied to various settings in QA research. Merging at the document-level is common in IR systems (e.g., (Tsai et al., 2008)), and has shown to improve multilingual QA performance as well (García-Cumbreras et al., 2012). Many QA systems combine answers obtained by different variants of the underlying model (e.g., (Brill et al., 2001) for monolingual, (Ko et al., 2010a; Ko et al., 2010b) for multilingual QA). We are not aware, however, of any prior work that has explored the ***merging of answers that are generated by language-specific ranking models***. Although this does not show increased effectiveness in our experiments, we believe that it brings a new perspective to the problem.

3 Approach

Our work is focused on a specific stage of the QA pipeline, namely *answer ranking*: Given a natural-language question q and k candidate answers $s_1, \ldots, s_k$, we score each answer in terms of its relevance to q. In our case, candidate answers are sentences extracted from all documents retrieved in the previous stage of the pipeline (using Indri (Metzler and Croft, 2005)). Hereafter, sentence and answer might be used interchangeably.

While our approach is not language-specific, we assume (for simplicity) that questions are in English, whereas sentences are in either English, Arabic, or Chinese. Non-English answers are translated back into English before returning to user.

Our approach is not limited to any question type, factoid or non-factoid. Our main motivation is to provide good QA quality on any multilingual Web collection. This entails finding answers to questions where there is no single answer, and for which human agreement is low. We aim to build a system that can successfully retrieve relevant information from open-domain and informal-language content. In this scenario, two assumptions made by many of the prior approaches fail:

1) We can accurately classify questions via template patterns (Chaturvedi et al. argue that this does not hold for non-factoid questions (2014))

2) We can accurately determine the relevance of an answer, based on its automatic translation into English (Wees et al. show how recall decreases when translating user-generated text (2015))

To avoid these assumptions, we opted for a more adaptable approach, in which question-answer relevance is modeled as a function of features, intended to capture the relationship between the question and sentence text. Also, instead of relying solely on a single potentially incorrect English translation, we increase our chances of a hit by translating both the question and the candidate answer, using four different translation methods. Our main features, described throughout this section, are based on lexical similarity computed using these translations. The classifier is trained on a large number of question-answer pairs, each labeled by a human annotator with a binary relevance label.[6]

3.1 Representation

In MLQA, since questions and answers are in different languages, most approaches translate both into an intermediary language (usually English). Due to the error-prone nature of MT, valuable information often gets "lost in translation". These errors are especially noticeable when translating informal text or less-studied languages (Van der Wees et al., 2015).

Translation Direction. We perform a *two-way translation* to better retain the original meaning: in addition to translating each non-English sentence into English, we also translate the English questions into Arabic and Chinese (using multiple translation methods, described below). For each question-answer pair, we have two "views": comparing translated question to the original sentence (i.e., *collection-language* (CL) view); and comparing original question to the translated sentence (i.e., *question-language* (QL) view).

Translation Method. When translating text for retrieval tasks like QA, including a variety of alterna-

[6] Annotators score each answer from 1 to 5. We label any score of 3 or higher as relevant.

tive translations is as important as finding the most accurate translation, especially for non-factoid questions, where capturing (potentially multiple) underlying topics is essential. Recent work in cross-language IR (CLIR) has shown that incorporating probabilities from the internal representations of an MT system to "translate" the question can accomplish this, outperforming standard one-best translation (Ture and Lin, 2014). We hypothesize that these improvements transfer to multilingual QA as well.

In addition to *translation directions*, we explored four *translation methods* for converting the English question into a probabilistic representation (in Arabic and Chinese). Each method builds a probability distribution for every question word, expressing the translation space in the collection language. More details of first three methods can be found in (Ture and Lin, 2014), while fourth method is a novel query translation method adapted from the neural network translation model described in (Devlin et al., 2014).

Word: In MT, a word alignment is a many-to-many mapping between source- and target-language words, learned without supervision, at the beginning of the training pipeline (Och, 2003). These alignments can be converted into word translation probabilities for CLIR (Darwish and Oard, 2003).

For example, in an English-Arabic parallel corpus, if an English word appears m times in total and is aligned to a certain Arabic word k times, we assign a probability of $\frac{k}{m}$ for this translation. This simple idea has performed greatly in IR for generating a probability distribution for query word translations.

Grammar: Probabilities are derived from a synchronous context-free grammar, which is a typical translation model found in MT systems (Ture and Lin, 2014). The grammar contains rules r that follow the form $\alpha \mid \beta \mid \mathcal{A} \mid \ell(r)$, stating that source-language word α can be translated into target-language word β with an associated likelihood value $\ell(r)$ ($\mathcal{A}$ represents word alignments). For each rule r that applies to the question, we identify each source word s_j. From the word alignment information included in the rule, we can find all target words that s_j is aligned to. By processing all the rules to accumulate likelihood values, we construct translation probabilities for each word in the question.

10-best: Statistical MT systems retrieve a ranked list of translations, not a single best. Ture and Lin exploited this to obtain word translation probabilities from the top 10 translations of the question (2014). For each question word w, we can extract which grammar rules were used to produce the translation – once we have the rules, word alignments allow us to find all target-language words that w translates into. By doing this for each question translation, we construct a probability distribution that defines the translation space of each question word.

Context: Neural network-based MT models learn context-dependent word translation probabilities – the probability of a target word is dependent on the source word it aligns to, as well as a 5-word window of context (Devlin et al., 2014). For example, if the Spanish word "placer" is aligned to the English word "pleasure", the model will not only learn from this word-to-word alignment but also consider the source sentence context (e.g., "Fue un placer conocerte y tenerte unos meses."). However, since short questions might lack full sentence context, our model should have the flexibility to translate under partial or no context. Instead of training the NN-base translation model on full, well-formed sentences, we custom-fit it for question translation: words in the context window are randomly masked by replacing it with a special filler token <F>. This teaches the model how to accurately translate with full, partial context, or no context. For the above example, we generate partial contexts such as "fue un placer <F> y" or "<F> <F> placer conocerte y". Since there are many possibilities, if the context window is large, we randomly sample a few of the possibilities (e.g., 4 out of 9) per training word.

In Figure 1, we display the probabilistic structure produced the *grammar-based* translation method, when implemented as described above. Each English word in the question is translated into a probabilistic structure, consisting of Chinese words and corresponding probabilities that represent how much weight the method decides to put on that specific word. Similar structures are learned with the other three translation methods.

We are not aware of any other MLQA approach that represents the question-answer pair based on their probabilistic translation space.

Figure 1: Probabilistic grammar-based translation of example question. The example question "Tell me about child labor in Africa" is simplified by our preprocessing engine to "child labor africa".

3.2 Features

Given two different translation directions (*CL* and *QL*), and four different translation methods (*Word*, *Grammar*, *10-best*, *Context*), our strategy is to leverage a machine learning process to determine how helpful each signal is with respect to the end task. For this, we introduced separate question-answer similarity features based on each combination of translation direction and method.

Collection-language Features. In order to compute a single real-valued vector to represent the question in the collection language (*LexCL*), we start with the probabilistic structure representing the question translation (e.g., Figure 1 is one such structure when the translation method is *grammar-based*). For each word in the collection-language vocabulary, we compute a weight by averaging its probability across the terms in the probabilistic structure.

$$v_{q\text{grammar}}(w) = \text{avg}_i \Pr(w|q_i) \quad (1)$$

where w is a non-Engish word and $\Pr(w|q_i)$ is the probability of w in the probability distribution corresponding to the i^{th} query term.

Figure 2 shows the real-valued vector computed based on the probabilistic question translation in Figure 1. The Chinese word translated as "child labor" has a weight of 0.32, 0.36, and 0 in the probability distributions of the three query terms, respectively. Averaging these three values results in the final weight of 0.23 in $v_{q\text{grammar}}$ in Figure 2. Notice that these weights are normalized by construction.

Similarly, a candidate answer s in Chinese is represented by normalized word frequencies:

$$v_s(w) = \frac{\text{freq}(w|s)}{\sum_{w'} \text{freq}(w'|s)} \quad (2)$$

Given the two vectors, we compute the cosine similarity. Same process is repeated for the other three translation methods. The four lexical collection-language similarity features are collectively called *LexCL*.

Feature category	Question repr. ($v_{q'}$)	Sentence repr. ($v_{s'}$)	Feature Value
LexCL	$v_{q\text{word}}$	v_s	cosine $(v_{q'}, v_{s'})$
	$v_{q\text{10best}}$	v_s	
	$v_{q\text{context}}$	v_s	
	$v_{q\text{grammar}}$	v_s	
LexQL	v_q	$v_{s\text{1best}}$	

Table 1: List of features used in `L2T`, and how the values are computed from vector representations.

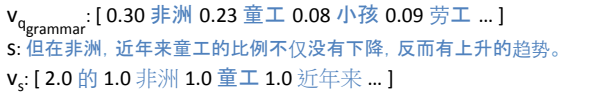


Figure 2: Vector representation of grammar-translated question ($q$grammar) and sentence (s).

Question-language Features. As mentioned before, we also obtain a similarity value by translating the sentence (s_{1best}) and computing the cosine similarity with the original question (q). v_q and $v_{s\text{1best}}$ are computed using Equation 2. Although it is possible to translate the sentence into English using the same four methods, we only used the one-best translation due to the computational cost. Hence, we have only one lexical similarity feature in the QL view (call *LexQL*).

The computation process for the five *lexical similarity* features is summarized in Table 1. After computation, feature weights are learned via a maximum-entropy model.[7] Although not included in the figure or table, we also include the same set of features from the sentence preceding the answer (within the corresponding forum post), in order to represent the larger discourse.

3.3 Data Selection

In order to train a machine learning model with our novel features, we need positive and negative examples of question-answer pairs (i.e., (q, s)). For this, for each training question, our approach is to hire

[7] Support vector machines yielded worse results.

human annotators to label sentences retrieved from the non-English collections used in our evaluation. It is possible to label the sentences in the source language (i.e., Arabic or Chinese) or in the question language (i.e., translated into English). In this section, we explore the question of whether it is useful to distinguish between these two independently created labels, and whether this redundancy can be used to improve the machine learning process.

We hypothesize two reasons why selecting training data based on language might benefit MLQA:
i) The translation of non-English candidate answers might lack in quality, so annotators are likely to judge some relevant answers as non-relevant. Hence, training a classifier on this data might lead to a tendency to favor English answers.
ii) For the question-answer pairs that were annotated in both languages, we can remove noisy (i.e., labeled inconsistently by annotators) instances from the training set.

The question of annotation is an unavoidable part of evaluation of MLQA systems, so finding the optimal subset for training is a relevant problem. In order to explore further, we generated six subsets with respect to (a) the original *language* of the answer, or (b) the language of *annotation* (i.e., based on original text or its English translation):
en: Sentences from the English corpus.
ar/ch: Sentences from the Arabic / Chinese corpus (regardless of how it was judged).
consist: All sentences except those that were judged inconsistently.
src+: Sentences judged only in original text, or judged in both consistently.
en+: Sentences that are either judged only in English, or judged in both original and English translation consistently.
all: All sentences.

These subsets were determined based on linguistically motivated heuristics, but choosing the most suitable one (for a given task) is done via machine learning (see Section 4).

3.4 Language-specific Ranking

Scoring Arabic sentences with respect to a question is inherently different than scoring English (or Chinese) sentences. The quality of resources, grammar, etc., as well as other internal dynamics might differ greatly across languages. We hypothesize that there is no one-size-fits-all model, so the parameters that work best for English retrieval might not be as useful when scoring sentences in Arabic, and/or Chinese.

Our proposed solution is to apply a separate classifier, custom-tuned to each collection, and retrieve three *single-language ranked lists* (i.e., in English, Arabic, and Chinese). In addition to comparing each custom-tuned, language-specific classifier to a single, language-independent one, we also use this idea to propose an approach for MLQA:
L2CT(n) Retrieve answers from each language using separate classifiers (call these lists English-only, Arabic-only, and Chinese-only), take the best answers from each language, then merge them into a mixed-language set of n answers.

We compare this to the standard approach:
L2T(n) Retrieve up to n mixed-language answers using a single classifier.

Four heuristics were explored for merging lists in the L2CT approach.[8] Two common approaches are *uniform* and *alternate* merging (Savoy, 2004):
Uniform: A straightforward merge can be achieved by using the classifier scores (i.e., probability of answer relevance, given question) to sort all answers, across all languages, and include the top n in the final list of answers. Classifier scores are normalized into the [0,1] range for comparability.
Alternate: We alternate between the lists, picking one answer at a time from each, stopping when the limit n has been reached.

Since answers are expected in English, there is a natural preference for answers that were originally written English, avoiding noisy text due to translation errors. However, it is also important not to restrict answers entirely to English sources, since that would defeat the purpose of searching in a multilingual collection. We implemented the following methods to account for language preferences:
English first: We keep all sufficiently-confident (i.e., normalized score above a fixed threshold) answers from the English-only list first, and start including answers from Arabic- and Chinese-only lists only if the limit of n answers has not been reached.

[8] In addition to these heuristics, the optimal merge could be learned from training data, as a "learning to rank" problem. This is out of the scope of this paper, but we plan to explore the idea in the future.

Weighted: Similar to *Uniform*, but we weight the normalized scores before sorting. The optimal weights can be learned by using a grid-search procedure and a cross-validation split.

4 Evaluation

In order to perform controlled experiments and gain more insights, we split our evaluation into four separate tasks: three tasks focus on retrieving answers from posts written in a specified language (*English-only*, *Arabic-only*, or *Chinese-only*) [9], and the last task is not restricted to any language (*Mixed-language*). All experiments were conducted on the DARPA BOLT-IR task. The collection consists of 12.6M Arabic, 7.5M Chinese, and 9.6M English Web forum posts. All runs use a set of 45 non-factoid (mostly opinion and causal) English questions, from a range of topics. All questions and forum posts were processed with an information extraction (IE) toolkit (Boschee et al., 2005), which performs sentence-splitting, named entity recognition, coreference resolution, parsing, and part-of-speech tagging.

All non-English posts were translated into English (one-best only), and all questions were translated into Arabic and Chinese (probabilistic translation methods from Section 3.1). For all experiments, we used the same state-of-the-art English↔Arabic (En-Ar) and English↔Chinese (En-Ch) MT systems (Devlin et al., 2014). Models were trained on parallel corpora from NIST OpenMT 2012, in addition to parallel forum data collected as part of the BOLT program (10M En-Ar words; 30M En-Ch words). Word alignments were learned with GIZA++ (Och and Ney, 2003) (five iterations of IBM Models 1–4 and HMM).

After all preprocessing, features were computed using the original post and question text, and their translations. Training data were created by having annotators label all sentences of the top 200 documents retrieved by Indri from each collection (for each question). Due to the nature of retrieval tasks, training data usually contains an unbalanced portion of negative examples. Hence, we split the data into balanced subsets (each sharing the same set of positively labeled data) and train multiple classifiers, then take a majority vote when predicting.

For testing, we froze the set of candidate answers and applied the trained classifier to each question-answer pair, generating a ranked list of answers for each question. This ranked list was evaluated by average precision (AP).[10] Due to the size and redundancy of the collections, we sometimes end up with over 1000 known relevant answers for a question. So it is neither reasonable nor meaningful to compute AP until we reach 100% recall (e.g., 11-point AP) for these cases. Instead, we computed AP-k, by accumulating precision values at every relevant answer until we get k relevant answers.[11] In order to provide a single metric for the test set, it is common to report the mean average precision (MAP), which in this case is the average of the AP-k values across all questions.

Baseline. As described earlier, the baseline system computes similarity between question text and the one-best translation of the candidate answer (we run the sentence through our state-of-the-art MT system). After translation, we compute similarity via scoring the match between the parse of the question text and the parse of the candidate answer, using our finely-tuned IE toolkit [reference removed for anonymization]. This results in three different similarity features: matching the tree node similarity, edge similarity, and full tree similarity. Feature weights are then learned by training this classifier discriminatively on the training data described above. This already performs competitively, outperforming the simpler baseline where we compute a single similarity score between question and translated text, and matching the performance of the system by Chaturvedi et al. on the BOLT evaluation (2014). Baseline MAP values are reported on the leftmost column of Table 2.

Data effect. In the baseline approach, we do not perform any data selection, and use all available data for training the classifier. In order to test our hypothesis that selecting a linguistically-motivated subset of the training data might help, we used 10-fold cross-validation to choose the optimal data set

[9] Shortened as *Eng*, *Arz*, and *Cmn*, respectively.

[10] Many other metrics (e.g., NDCG, R-precision) were explored during BOLT, and results were very similar.

[11] k was fixed to 20 in our evaluation, although we verified that conclusions do not change with varying k.

(among seven options described in Section 3.3). Results indicate that including English or Arabic sentences when training a classifier for Chinese-only QA is a bad idea, since effectiveness increases when restricted to Chinese sentences (`lang=ch`). On the other hand, for the remaining three tasks, the most effective training data set is `annot=en+consist`. These selections are consistent across all ten folds, and the difference is statistically significant for all but Arabic-only. The second column in Table 2 displays the MAP achieved when data selection is applied before training the baseline model.

Feature effect. To measure the impact of our novel features, we trained classifiers using either *LexCL*, *LexQL*, or *both* feature sets (Section 3.2). In these experiments, the data is fixed to the optimal subset found earlier. Results are summarized on right side of Table 2. Statistically significants improvements over *Baseline*/*Baseline+Data selection* are indicated with single/double underlining.

For Arabic-only QA, adding *LexQL* features yields greatest improvements over the baseline, while the same statement holds for *LexCL* features for the Chinese-only task. For the English-only and mixed-language tasks, the most significant increase in MAP is observed with all of our probabilistic bilingual features. For all but Arabic-only QA, the MAP is statistically significantly better ($p < 0.05$) than the baseline; for Chinese-only and mixed-language tasks, it also outperforms baseline plus data selection ($p < 0.05$).[12] All of this indicates the effectiveness of our probabilistic question translation, as well as our data selection strategy.

Task	Base	+Data	+Feats
Cmn	0.416	0.425 (*ch*)	0.451 (*LexCL*)
Arz	0.421	0.423 (*en+*)	0.425 (*LexQL*)
Eng	0.637	0.657 (*en+*)	0.660 (*all*)
Mixed	0.665	0.675 (*en+*)	0.681 (*all*)

Table 2: `L2T` evaluated using MAP with 10-fold cross-validation for each task. A statistically significant increase over Baseline/Base+Data is shown by single/double underlining ($p < 0.05$).

Understanding the contribution of each of the four *LexCL* features is also important. To gain insight, we trained a classifier using all *LexCL* features (using the optimal data subset learned earlier for each task), and then incrementally removed one of the features, and tested on the same task. This controlled experiment revealed that the *word* translation feature is most useful for Chinese-only QA (i.e., removing it produces largest drop in MAP, 0.6 points), whereas *context* translation appears to be most useful (by a slighter margin) in Arabic-only QA. In the former case, the diversity provided by word translation might be useful at increasing recall in retrieving Chinese answers. In retrieving Arabic answers, using context to disambiguate the translation might be useful at increasing precision. This result further emphasizes the importance of a customized translation approach for MLQA.

Furthermore, to test the effectiveness of the probabilistic translation approach (Section 3.1), we replaced all *LexCL* features with a single lexical similarity feature computed from the one-best question translation. This resulted in lower MAP: 0.427 to 0.423 for Arabic-only, and 0.451 to 0.425 for Chinese-only task ($p < 0.01$), supporting the hypothesis that *probabilistic translation is more effective than the widely-used one-best translation*. In fact, almost all gains in Chinese-only QA seems to be coming from the probabilistic translation.

For a robustness test, we let cross-validation select the best combination of (*data*, *feature*), mimicking a less controlled, real-world setting. In this case, the best MAP for the Arabic-, Chinese-, English-only, and Mixed-language tasks are 0.403, 0.448, 0.657, and 0.679, respectively. In all but Arabic-only, these are statistically significantly better ($p < 0.05$) than not tuning the feature set or training data (i.e., Baseline). This result suggests that our approach can be used for any MLQA task out of the box, and provide improvements.

Learning to Custom Translate (`L2CT`). We took the ranked list of answers output by each language-specific model, and merged all of them into a ranked list of mixed-language answers. For the *weighted* heuristic, we tried three values for the weight. In Table 3, we see that training separate classifiers for each subtask does not bring overall improvements to the end task. Amongst merging strategies,

[12] Note that bilingual features are not expected to help on the English-only task, and the improvements come solely from data selection.

the most effective were *weighted* (weights for each query learned by performing a grid-search on other queries) and *English first* – however, both are statistically indistinguishable from the single classifier baseline. In the latter case, the percentage of English answers is highest (88%), which might not be desirable. Depending on the application, the ratio of languages can be adjusted with an appropriate merging method. For instance, *alternate* and *norm* heuristics tend to represent languages almost equally.

Approach			(En-Ch-Ar) %	MAP
L2T			64-19-16	0.681
L2CT	Uniform		24-35-41	0.548
	Alt.		32-34-34	0.574
	Eng. First		88-6-6	0.668
	Weight	2	37-30-34	0.599
		5	51-24-25	0.654
		10	61-20-19	0.669

Table 3: L2T vs. L2CT for multilingual QA.

Even though we get lower MAP in the overall task, Table 2 suggests that it is worthwhile customizing classifiers for each subtask (e.g., the Chinese responses in the ranked list of L2CT are more relevant than *Single*.). The question of how to effectively combine the results into a mixed-language list, however, remains an open question.

5 Conclusions

We introduced L2T, a novel approach for MLQA, inspired from recent success in CLIR research. To our knowledge, this is the first use of probabilistic translation methods for this task, and the first attempt at using machine learning to learn the optimal question translation.

We also proposed L2CT, which uses language-specific classifiers to treat the ranking of English, Arabic, and Chinese answers as three separate subtasks, by applying a separate classifier for each language. While post-retrieval merging has been studied in the past, we have not come across any work that applies this idea specifically to create a language-aware ranking for MLQA.

Our experimental analysis shows the importance of data selection when dealing with annotations on source and translated text, and the effect of combining translation methods. L2T improved answer ranking effectiveness significantly for Chinese-only, English-only, and mixed-language QA.

Although results did not support the hypothesis that *learning a custom classifier* for the retrieval of each language would outperform the *single classifier* baseline, we think that more research is needed to fully understand how language-specific modeling can benefit MLQA. More sophisticated merging of multiple ranked lists of answers need to be explored. Learning to rank between answers from different languages might be more effective than heuristics. This would allow us to predict the final language ratio, based on many features (e.g., general collection statistics, quality of candidate answers, question category and complexity, MT system confidence levels) to merge question-answer pairs.

An even more comprehensive use of machine learning would be to learn word-level translation scores, instead of relying on translation probabilities from the bilingual dictionary, resulting in a fully customized translation. Similar approaches have appeared in learning-to-rank literature for monolingual IR (Bendersky et al., 2010), but not for multilingual retrieval. Another extension of this work would be to apply the same translation for translating answers into the question language (in addition to question translation). By doing this, we would be able to capture the semantics of each answer much better, since we have discussed that one-best translation discards a lot of potentially useful information.

Finally, since one of the take-away messages of our work is that a deeper understanding of linguistic context can improve QA effectiveness via more sophisticated question translation, we are hoping to see even more improvements by creating features based on word embeddings. One potential next step is to learn bilingual embeddings directly for the task of QA, for which we have started adapting some related work (Bai et al., 2010).

Acknowledgements

Jacob Devlin has provided great help in the design and implementation of the context-based question translation approach. We would also like to thank the anonymous reviewers for their helpful feedback.

References

SisayFissaha Adafre and Josef van Genabith. 2009. Dublin city university at qaclef 2008. In Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, GarethJ.F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo Peñas, and Vivien Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *Lecture Notes in Computer Science*, pages 353–360. Springer Berlin Heidelberg.

Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D Lawrence, David C Gondek, and James Fan. 2012. Learning to Rank for Robust Question Answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 833–842, New York, NY, USA. ACM.

Enrique Alfonseca, Marco De Boni, José-Luis Jara-Valencia, and Suresh Manandhar. 2001. A prototype question answering system using syntactic and semantic information for answer retrieval. In *TREC*.

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. 2001. Piqasso: Pisa question answering system. In *TREC*.

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Q. Weinberger. 2010. Learning to rank with (a lot of) word features. *Inf. Retr.*, 13(3):291–314.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 31–40, New York, NY, USA. ACM.

Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*, volume 71.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 393–400.

Snigdha Chaturvedi, Vittorio Castelli, Radu Florian, Ramesh M Nallapati, and Hema Raghavan. 2014. Joint Question Clustering and Relevance Prediction for Open Domain Non-factoid Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 503–514, New York, NY, USA. ACM.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 400–407, New York, NY, USA. ACM.

Kareem Darwish and Douglas W. Oard. 2003. Probabilistic structured query methods. In *SIGIR*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA*, pages 1370–1380.

Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel L Forcada. 2012. UAlacant: Using Online Machine Translation for Cross-lingual Textual Entailment. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 472–476, Stroudsburg, PA, USA. Association for Computational Linguistics.

M Á García-Cumbreras, F Martínez-Santiago, and L A Ureña López. 2012. Architecture and Evaluation of BRUJA, a Multilingual Question Answering System. *Inf. Retr.*, 15(5):413–432, October.

Sanda Harabagiu and Cosmin Adrian Bejan. 2005. Question answering based on temporal inference. In *Proceedings of the AAAI-2005 workshop on inference for textual question answering*, pages 27–34.

Sven Hartrumpf, Ingo GlÃckner, and Johannes Leveling. 2009. Efficient question answering with question decomposition and multiple answer streams. In Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, GarethJ.F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo PeÃ±as, and Vivien Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *Lecture Notes in Computer Science*, pages 421–428. Springer Berlin Heidelberg.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the first international conference on Human language technology research*, pages 1–7. Association for Computational Linguistics.

Boris Katz, Gary Borchardt, and Sue Felshin. 2005. Syntactic and semantic decomposition strategies for question answering from multiple resources. In *Proceedings of the AAAI 2005 workshop on inference for textual question answering*, pages 35–41.

Jeongwoo Ko, Luo Si, and Eric Nyberg. 2010a. Combining Evidence with a Probabilistic Framework for

Answer Ranking and Answer Merging in Question Answering. *Inf. Process. Manage.*, 46(5):541–554, September.

Jeongwoo Ko, Luo Si, Eric Nyberg, and Teruko Mitamura. 2010b. Probabilistic Models for Answer-ranking in Multilingual Question-answering. *ACM Trans. Inf. Syst.*, 28(3):16:1—-16:37, July.

Oleksandr Kolomiyets and Marie-Francine Moens. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434.

Chuan-Jie Lin and Yu-Min Kuo. 2010. Description of the ntou complex qa system. In *Proceedings of NTCIR-8 Workshop*.

Xiaoqiang Luo, Hema Raghavan, Vittorio Castelli, Sameer Maskey, and Radu Florian. 2013. Finding what matters in questions. In *Proceedings of NAACL-HLT'13*.

N Madnani, Jimmy Lin, and Bonnie J Dorr. 2007. TREC 2007 ciQA Task: University of Maryland. *Proceedings of TREC*.

Ãngel Martinez-Gonzalez, Cesar de Pablo-Sanchez, Concepcion Polo-Bayo, MarÃaTeresa Vicente-Diez, Paloma Martinez-Fernandez, and Jose Luis Martinez-Fernandez. 2009. The miracle team at the clef 2008 multilingual question answering track. In Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, GarethJ.F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo PeÃ±as, and Vivien Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *Lecture Notes in Computer Science*, pages 409–420. Springer Berlin Heidelberg.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2010. Towards Cross-lingual Textual Entailment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 321–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

Donald Metzler and W Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 472–479, New York, NY, USA. ACM.

Teruko Mitamura, Mengqiu Wang, Hideki Shima, and Frank Lin. 2006. Keyword translation accuracy and cross-lingual question answering in chinese and japanese. In *Proceedings of the Workshop on Multilingual Question Answering*, MLQA '06, pages 31–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas S Morton. 1999. Using Coreference for Question Answering. In *Proceedings of the Workshop on Coreference and Its Applications*, CorefApp '99, pages 85–89, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rafael Munoz-Terol, Marcel Puchol-Blasco, Maria Pardino, Jose Manuel Gomez, Sandra Roger, Katia Vila, Antonio Ferrandez, Jesus Peral, and Patricio Martinez-Barco. 2009. Integrating logic forms and anaphora resolution in the aliqan system. In *Evaluating Systems for Multilingual and Multimodal Information Access*, LNCS, pages 438–441. Springer Berlin Heidelberg.

Jian-Yun Nie. 2010. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Han Ren, Donghong Ji, and Jing Wan. 2010. Whu question answering system at ntcir-8 aclia task. In *Proceedings of NTCIR-8 Workshop*.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple {BM25} extension to multiple weighted fields. In *Proc. CIKM*, pages 42–49.

Bogdan Sacaleanu, Günter Neumann, and Christian Spurk. 2008. Dfki-lt at qaclef 2008. In Carol Peters and et al., editors, *CLEF 2008 Working Notes*, Working Notes. Springer Verlag.

E Saquete, J L Vicedo, P Martínez-Barco, R Muñoz, and F Llopis. 2005. Evaluation of Complex Temporal Questions in CLEF-QA. In *Proceedings of the 5th Conference on Cross-Language Evaluation Forum: Multilingual Information Access for Text, Speech and Images*, CLEF'04, pages 591–596, Berlin, Heidelberg. Springer-Verlag.

Jacques Savoy. 2004. Combining multiple strategies for effective monolingual and cross-language retrieval. *Information Retrieval*, 7(1-2):121–148.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21. Citeseer.

Hideki Shima and Teruko Mitamura. 2010. Bootstrap pattern learning for open-domain clqa. In *Proceedings of NTCIR-8 Workshop*.

Ming-Feng Tsai, Yu-Ting Wang, and Hsin-Hsi Chen. 2008. A study of learning a merge model for multilingual information retrieval. In *Proceedings of the 31st*

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pages 195–202, New York, NY, USA. ACM.

Ferhan Ture and Jimmy Lin. 2014. Exploiting representations from statistical machine translation for cross-language information retrieval. *ACM Trans. Inf. Syst.*, 32(4):19:1–19:32, October.

Marlies Van der Wees, Arianna Bisazza, Wouter Weerkamp, and Christof Monz. 2015. What's in a domain? analyzing genre and topic differences in statistical machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 560–566, Beijing, China, July. Association for Computational Linguistics.

Chen Zhang, Matthew Gerber, Tyler Baldwin, Steven Emelander, Joyce Chai, and Rong Jin. 2007. Michigan State University at the 2007 TREC ciQA Task. In *Proceedings of the Sixteenth Text Retrieval Conference*, Gaithersburg, Maryland, November.

A Semiparametric Model for Bayesian Reader Identification

Ahmed Abdelwahab[1] and **Reinhold Kliegl**[2] and **Niels Landwehr**[1]
[1] Department of Computer Science, Universität Potsdam
August-Bebel-Straße 89, 14482 Potsdam, Germany
{ahmed.abdelwahab, niels.landwehr}@uni-potsdam.de
[2] Department of Psychology, Universität Potsdam
Karl-Liebknecht-Straße 24/25, 14476 Potsdam OT/Golm
kliegl@uni-potsdam.de

Abstract

We study the problem of identifying individuals based on their characteristic gaze patterns during reading of arbitrary text. The motivation for this problem is an unobtrusive biometric setting in which a user is observed during access to a document, but no specific challenge protocol requiring the user's time and attention is carried out. Existing models of individual differences in gaze control during reading are either based on simple aggregate features of eye movements, or rely on parametric density models to describe, for instance, saccade amplitudes or word fixation durations. We develop flexible semiparametric models of eye movements during reading in which densities are inferred under a Gaussian process prior centered at a parametric distribution family that is expected to approximate the true distribution well. An empirical study on reading data from 251 individuals shows significant improvements over the state of the art.

1 Introduction

Eye-movement patterns during skilled reading consist of brief fixations of individual words in a text that are interleaved with quick eye movements called *saccades* that change the point of fixation to another word. Eye movements are driven both by low-level visual cues and high-level linguistic and cognitive processes related to text understanding; as a reflection of the interplay between vision, cognition, and motor control during reading they are frequently studied in cognitive psychology (Kliegl et al., 2006; Rayner, 1998). Computational models (Engbert et al., 2005; Reichle et al., 1998) as well as models based on machine learning (Matties and Søgaard, 2013; Hara et al., 2012) have been developed to study how gaze patterns arise based on text content and structure, facilitating the understanding of human reading processes.

A central observation in these and earlier psychological studies (Huey, 1908; Dixon, 1951) is that eye movement patterns strongly differ between individuals. Holland et al. (2012) and Landwehr et al. (2014) have developed models of individual differences in eye movement patterns during reading, and studied these models in a biometric problem setting where an individual has to be identified based on observing her eye movement patterns while reading arbitrary text. Using eye movements during reading as a biometric feature has the advantage that it suffices to observe a user during a routine access to a device or document, without requiring the user to react to a specific challenge protocol. If the observed eye movement sequence is unlikely to be generated by an authorized individual, access can be terminated or an additional verification requested. This is in contrast to approaches where biometric identification is based on eye movements in response to an artificial visual stimulus, for example a moving (Kasprowski and Ober, 2004; Komogortsev et al., 2010; Rigas et al., 2012b; Zhang and Juhola, 2012) or fixed (Bednarik et al., 2005) dot on a computer screen, or a specific image stimulus (Rigas et al., 2012a).

The model studied by Holland & Komogortsev (2012) uses aggregate features (such as average fixation duration) of the observed eye movements. Landwehr et al. (2014) showed that readers can be identified more accurately with a model that captures aspects of individual-specific distributions over

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 585–594,
Austin, Texas, November 1-5, 2016.

eye movements, such as the distribution over fixation durations or saccade amplitudes for word refixations, regressions, or next-word movements. Some of these distributions need to be estimated from very few observations; a key challenge is thus to design models that are flexible enough to capture characteristic differences between readers yet robust to sparse data. Landwehr et al. (2014) used a fully parametric approach where all densities are assumed to be in the gamma family; gamma distributions were shown to approximate the true distribution of interest well for most cases (see Figure 1). This model is robust to sparse data, but might not be flexible enough to capture all differences between readers.

The model we study in this paper follows ideas developed by Landwehr et al. (2014), but employs more flexible semiparametric density models. Specifically, we place a Gaussian process prior over densities that concentrates probability mass on densities that are close to the gamma family. Given data, a posterior distribution over densities is derived. If data is sparse, the posterior will still be sharply peaked around distributions in the gamma family, reducing the effective capacity of the model and minimizing overfitting. However, given enough evidence in the data, the model will also deviate from the gamma-centered prior—depending on the kernel function chosen for the GP prior, any density function can in principle be represented. Integrating over the space of densities weighted by the posterior yields a marginal likelihood for novel observations from which predictions are inferred. We empirically study this model in the same setting as studied by Landwehr et al. (2014), but using an order of magnitude more individuals. Identification error is reduced by more than a factor of three compared to the state of the art.

The rest of the paper is organized as follows. After defining the problem setting in Section 2, Section 3 presents the semiparametric probabilistic model. Section 4 discusses inference, Section 5 presents an empirical study on reader identification.

2 Problem Setting

Assume R different readers, indexed by $r \in \{1, \ldots, R\}$, and let $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ denote a set of texts. Each $r \in \mathcal{R}$ generates a set of eye movement patterns $\mathcal{S}^{(r)} = \{\mathbf{S}_1^{(r)}, \ldots, \mathbf{S}_n^{(r)}\}$ on $\mathcal{X}$, by

$$\mathbf{S}_i^{(r)} \sim p(\mathbf{S}|\mathbf{X}_i, r, \mathbf{\Gamma})$$

where $p(\mathbf{S}|\mathbf{X}_i, r, \mathbf{\Gamma})$ is a reader-specific distribution over eye movement patterns given a text $\mathbf{X}_i$. Here, r is a variable indicating the reader generating the sequence, and $\mathbf{\Gamma}$ is a true but unknown model that defines all reader-specific distributions. We assume that $\mathbf{\Gamma}$ can be broken down into reader-specific models, $\mathbf{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_k)$, such that the distribution

$$p(\mathbf{S}|\mathbf{X}_i, r, \mathbf{\Gamma}) = p(\mathbf{S}|\mathbf{X}_i, \boldsymbol{\gamma}_r) \tag{1}$$

is defined by the partial model $\boldsymbol{\gamma}_r$. We aggregate the observations of all readers on the training data into a variable $\mathcal{S}^{(1:R)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(R)})$.

We follow a Bayesian approach, defining a prior $p(\mathbf{\Gamma})$ over the joint model that factorizes into priors over reader-specific models, $p(\mathbf{\Gamma}) = \prod_{r=1}^{R} p(\boldsymbol{\gamma}_r)$. At test time, we observe novel eye movement patterns $\bar{\mathcal{S}} = \{\bar{\mathbf{S}}_1, \ldots, \bar{\mathbf{S}}_m\}$ on a novel set of texts $\bar{\mathcal{X}} = \{\bar{\mathbf{X}}_1, \ldots, \bar{\mathbf{X}}_m\}$ generated by an unknown reader $r \in \mathcal{R}$. We assume a uniform prior over readers, that is, each $r \in \mathcal{R}$ is equally likely to be observed at test time. The goal is to infer the most likely reader to have generated the novel eye movement patterns. In a Bayesian setting, this means inferring the most likely reader given the training observations $(\mathcal{X}, \mathcal{S}^{(1:R)})$ and test observation $(\bar{\mathcal{X}}, \bar{\mathcal{S}})$:

$$r_* = \arg\max_{r \in \mathcal{R}} p(r|\bar{\mathcal{X}}, \bar{\mathcal{S}}, \mathcal{X}, \mathcal{S}^{(1:R)}). \tag{2}$$

We can rewrite Equation 2 to

$$\begin{aligned} r_* &= \arg\max_{r \in \mathcal{R}} p(\bar{\mathcal{S}}|r, \bar{\mathcal{X}}, \mathcal{X}, \mathcal{S}^{(1:R)}) && (3) \\ &= \arg\max_{r \in \mathcal{R}} \int p(\bar{\mathcal{S}}|r, \bar{\mathcal{X}}, \mathbf{\Gamma}) p(\mathbf{\Gamma}|\mathcal{X}, \mathcal{S}^{(1:R)}) \mathrm{d}\mathbf{\Gamma} \\ &= \arg\max_{r \in \mathcal{R}} \int p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \boldsymbol{\gamma}_r) p(\boldsymbol{\gamma}_r|\mathcal{X}, \mathcal{S}^{(r)}) \mathrm{d}\boldsymbol{\gamma}_r && (4) \end{aligned}$$

where

$$p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \boldsymbol{\gamma}_r) = \prod_{i=1}^{m} p(\bar{\mathbf{S}}_i|\bar{\mathbf{X}}_i, \boldsymbol{\gamma}_r) \tag{5}$$

$$p(\boldsymbol{\gamma}_r|\mathcal{X}, \mathcal{S}^{(r)}) \propto p(\boldsymbol{\gamma}_r) \prod_{i=1}^{n} p(\mathbf{S}_i^{(r)}|\mathbf{X}_i, \boldsymbol{\gamma}_r). \tag{6}$$

In Equation 3 we exploit that readers are uniformly chosen at test time, and in Equation 4 we exploit the factorization $p(\boldsymbol{\Gamma}) = \prod_{r=1}^{R} p(\boldsymbol{\gamma}_r)$ of the prior, which together with Equation 1 entails a factorization $p(\boldsymbol{\Gamma}|\mathcal{X}, \mathcal{S}^{(1:R)}) = \prod_{r=1}^{R} p(\boldsymbol{\gamma}_r|\mathcal{X}, \mathcal{S}^{(r)})$ of the posterior. Note that Equation 4 states that at test time we predict the reader r for which the marginal likelihood (that is, after integrating out the reader-specific model γ_r) of the test observations is highest. The next section discusses the reader-specific models $p(\mathbf{S}|\mathbf{X}, \boldsymbol{\gamma}_r)$ and prior distributions $p(\boldsymbol{\gamma}_r)$.

3 Probabilistic Model

The probabilistic model we employ follows the general structure proposed by Landwehr et al. (2014), but employs semiparametric density models and allows for fully Bayesian inference. To reduce notational clutter, let $\boldsymbol{\gamma} \in \{\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_R\}$ denote a particular reader-specific model, and let $\mathbf{X} \in \mathcal{X}$ denote a text. An eye movement pattern is a sequence $\mathbf{S} = ((s_1, d_1), \ldots, (s_T, d_T))$ of gaze fixations, consisting of a fixation position s_t (position in text that was fixated) and duration $d_t \in \mathbb{R}$ (length of fixation in milliseconds). In our experiments, individual sentences are presented in a single line on screen, thus we only model a horizontal gaze position $s_t \in \mathbb{R}$. We model $p(\mathbf{S}|\mathbf{X}, \boldsymbol{\gamma})$ as a dynamic process that successively generates fixation positions s_t and durations d_t in $\mathbf{S}$, reflecting how a reader generates a sequence of saccades in response to a text stimulus $\mathbf{X}$:

$$p(\mathbf{S}|\mathbf{X}, \boldsymbol{\gamma}) = p(s_1, d_1|\mathbf{X}, \boldsymbol{\gamma}) \prod_{t=2}^{T} p(s_t, d_t|s_{t-1}, \mathbf{X}, \boldsymbol{\gamma}),$$

where $p(s_t, d_t|s_{t-1}, \mathbf{X}, \boldsymbol{\gamma})$ models the generation of the next fixation position and duration given the old fixation position s_{t-1}. In the psychological literature, four different *saccade types* are distinguished: a reader can refixate the current word (*refixation*), fixate the next word in the text (*next word movement*), move the fixation to a word after the next word, that is, skip one or more words (*forward skip*), or regress to fixate a word occurring earlier in the text (*regression*), see, e.g., Heister et al. (2012). We observe empirically that for each saccade type, there is a characteristic distribution over saccade amplitudes and fixation durations, and that both approximately follow gamma distributions—see Figure 1. We therefore model $p(s_t, d_t|s_{t-1}, \mathbf{X}, \boldsymbol{\gamma})$ using a mixture over distributions for the four different saccade types. At each time t, the model first draws a saccade type $u_t \in \{1, 2, 3, 4\}$, and then draws a saccade amplitude a_t and fixation duration d_t from type-specific distributions $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\gamma})$ and $p(d|u_t, \boldsymbol{\gamma})$. More formally,

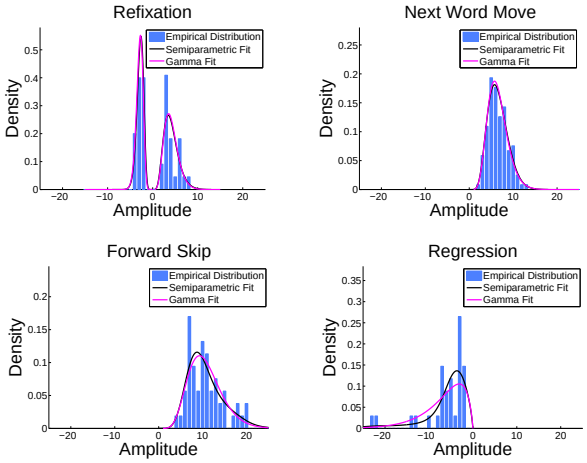


Figure 1: Empirical distributions of saccade amplitudes in training data for first individual, with fitted Gamma distributions and semiparametric distribution fits.

$$u_t \sim p(u|\boldsymbol{\pi}) \tag{7}$$
$$a_t \sim p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) \tag{8}$$
$$d_t \sim p(d|u_t, \boldsymbol{\delta}), \tag{9}$$

where $\boldsymbol{\gamma} = (\boldsymbol{\pi}, \boldsymbol{\alpha}, \boldsymbol{\delta})$ is decomposed into components $\boldsymbol{\pi}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\delta}$. Afterwards, the model updates the fixation position according to $s_t = s_{t-1} + a_t$, concluding the definition of $p(s_t, d_t|s_{t-1}, \mathbf{X}, \boldsymbol{\gamma})$. Figure 2 shows a slice in the dynamical model.

The distribution $p(u|\boldsymbol{\pi})$ over saccade types (Equation 7) is multinomial with parameter vector $\boldsymbol{\pi} \in \mathbb{R}^4$. The distributions over amplitudes and durations (Equations 8 and 9) are modeled semiparametrically as discussed in the following subsections.

3.1 Model of Saccade Amplitudes

We first discuss the amplitude model $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha})$ (Equation 8). We first define a distribution $p(a|u_t, \boldsymbol{\alpha})$ over amplitudes for saccade type u_t, and subsequently discuss conditioning on the text $\mathbf{X}$ and old fixation position s_{t-1},

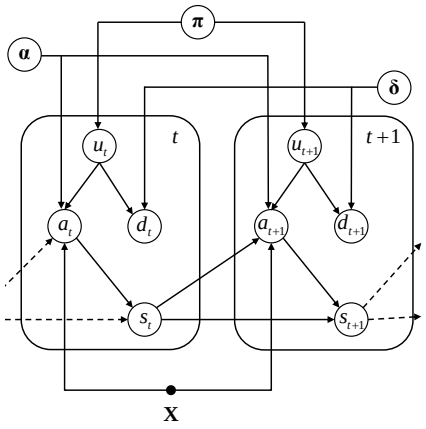


Figure 2: Plate notation of of a slice in the dynamic model.

leading to $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha})$. We define

$$p(a|u_t = 1, \boldsymbol{\alpha}) = \begin{cases} \mu\alpha_1(a) & : a > 0 \\ (1-\mu)\bar{\alpha}_1(-a) & : a \leq 0 \end{cases} \quad (10)$$

where μ is a mixture weight and α_1, $\bar{\alpha}_1$ are densities defining the distribution over positive and negative amplitudes for the saccade type *refixation*, and

$$p(a|u_t = 2, \boldsymbol{\alpha}) = \alpha_2(a) \quad (11)$$

$$p(a|u_t = 3, \boldsymbol{\alpha}) = \alpha_3(a) \quad (12)$$

$$p(a|u_t = 4, \boldsymbol{\alpha}) = \alpha_4(-a) \quad (13)$$

where $\alpha_2(a)$, $\alpha_3(a)$, and $\alpha_4(a)$ are densities defining the distribution over amplitudes for the remaining saccade types. Finally, the distribution

$$p(s_1|\mathbf{X}, \boldsymbol{\alpha}) = \alpha_0(s_1) \quad (14)$$

over the initial fixation position is given by another density function α_0. The variables μ, α_0, α_1, $\bar{\alpha}_1$, α_2, α_3, and α_4 are aggregated into model component $\boldsymbol{\alpha}$. For resolving the most likely reader at test time (Equation 4), densities in $\boldsymbol{\alpha}$ will be integrated out under a prior based on Gaussian processes (Section 3.3) using MCMC inference (Section 4).

Given the old fixation position s_{t-1}, the text $\mathbf{X}$, and the chosen saccade type u_t, the amplitude is constrained to fall within a specific interval. For instance, for a refixation the amplitude has to be chosen such that the novel fixation position lies within the beginning and the end of the currently fixated word; a regression implies an amplitude that is negative and makes the novel fixation position lie before the beginning of the currently fixated word. These constraints imposed by the text structure define the conditional distribution $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha})$. More formally, $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha})$ is the distribution $p(a|u_t, \boldsymbol{\alpha})$ conditioned on $a \in [l, r]$, that is,

$$p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) = p(a|a \in [l, r], u_t, \boldsymbol{\alpha}),$$

where l and r are the minimum and maximum amplitude consistent with the constraints. Recall that for a distribution over a continuous variable x given by density $\alpha(x)$, the distribution over x conditioned on $x \in [l, r]$ is given by the truncated density

$$\alpha(x|x \in [l, r]) = \begin{cases} \frac{\alpha(x)}{\int_l^r \alpha(\bar{x})\mathrm{d}\bar{x}} & : x \in [l, r] \\ 0 & : x \notin [l, r]. \end{cases} \quad (15)$$

We derive $p(a|u_t, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha})$ by truncating the distributions given by Equations 10 to 13 to the minimum and maximum amplitude consistent with the current fixation position s_{t-1} and text $\mathbf{X}$. Let w_l° (w_r°) denote the position of the left-most (right-most) character of the currently fixated word, and let w_l^+, w_r^+ denote these positions for the next word in $\mathbf{X}$. Let furthermore $l^\circ = w_l^\circ - s_{t-1}$, $r^\circ = w_r^\circ - s_{t-1}$, $l^+ = w_l^+ - s_{t-1}$, and $r^+ = w_r^+ - s_{t-1}$. Then

$$p(a|u_t = 1, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) = \begin{cases} \mu\alpha_1(a|a \in [0, r^\circ]) : a > 0 \\ (1-\mu)\bar{\alpha}_1(-a|a \in [l^\circ, 0]) : a \leq 0 \end{cases} \quad (16)$$

$$p(a|u_t = 2, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) = \alpha_2(a|a \in [l^+, r^+]) \quad (17)$$

$$p(a|u_t = 3, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) = \alpha_3(a|a \in (r^+, \infty)) \quad (18)$$

$$p(a|u_t = 4, s_{t-1}, \mathbf{X}, \boldsymbol{\alpha}) = \alpha_4(-a|a \in (-\infty, l^\circ)) \quad (19)$$

defines the appropriately truncated distributions.

3.2 Model of Fixation Durations

The model for fixation durations (Equation 9) is similarly specified by saccade type-specific densities,

$$p(d|u_t = u, \boldsymbol{\delta}) = \delta_u(d) \quad \text{for } u \in \{1, 2, 3, 4\} \quad (20)$$

and a density for the initial fixation durations

$$p(d_1|\mathbf{X}, \boldsymbol{\delta}) = \delta_0(d_1) \quad (21)$$

where $\delta_0, ..., \delta_4$ are aggregated into model component $\boldsymbol{\delta}$. Unlike saccade amplitude, the fixation duration is not constrained by the text structure and accordingly densities are not truncated. This concludes the definition of the model $p(\mathbf{S}|\mathbf{X}, \boldsymbol{\gamma})$.

3.3 Prior Distributions

The prior distribution over the entire model $\boldsymbol{\gamma}$ factorizes over the model components as

$$p(\boldsymbol{\gamma}|\lambda,\rho,\kappa) = \tag{22}$$
$$p(\boldsymbol{\pi}|\lambda)p(\mu|\rho)p(\bar{\alpha}_1|\kappa)\prod_{i=0}^{4}p(\alpha_i|\kappa)\prod_{i=0}^{4}p(\delta_i|\kappa)$$

where $p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\lambda)$ is a symmetric Dirichlet prior and $p(\mu) = \mathrm{Beta}(\mu|\rho)$ is a Beta prior. The key challenge is to develop appropriate priors for the densities defining saccade amplitude $(p(\bar{\alpha}_1|\kappa), p(\alpha_i|\kappa))$ and fixation duration $(p(\delta_i|\kappa))$ distributions. Empirically, we observe that amplitude and duration distributions tend to be close to gamma distributions—see the example in Figure 1.

Our goal is to exploit the prior knowledge that distributions tend to be closely approximated by gamma distributions, but allow the model to deviate from the gamma assumption in case there is enough evidence in the data. To this end, we define a prior over densities that concentrates probability mass around the gamma family. For all densities $f \in \{\bar{\alpha}_1, \alpha_0, ..., \alpha_4, \delta_0, ..., \delta_4\}$, we employ identical prior distributions $p(f|\kappa)$. Intuitively, the prior is given by first drawing a density function from the gamma family and then drawing the final density from a Gaussian process (with covariance function κ) centered at this function. More formally, let

$$\mathcal{G}(x|\boldsymbol{\eta}) = \frac{\exp(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x))}{\int \exp(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x'))\mathrm{d}x'} \tag{23}$$

denote the gamma distribution in exponential family form, with sufficient statistics $\mathbf{u}(x) = (\log(x), x)^\mathsf{T}$ and parameters $\boldsymbol{\eta} = (\eta_1, \eta_2)$. Let $p(\boldsymbol{\eta})$ denote a prior over the gamma parameters, and define

$$p(f|\kappa) = \int p(\boldsymbol{\eta})p(f|\boldsymbol{\eta},\kappa)\mathrm{d}\boldsymbol{\eta} \tag{24}$$

where $p(f|\boldsymbol{\eta},\kappa)$ is given by drawing

$$g \sim \mathcal{GP}(0,\kappa) \tag{25}$$

from a Gaussian process prior $\mathcal{GP}(0,\kappa)$ with mean zero and covariance function κ, and letting

$$f(x) = \frac{\exp(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x) + g(x))}{\int \exp(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x') + g(x'))\mathrm{d}x'}. \tag{26}$$

Note that decreasing the variance of the Gaussian process means regularizing $g(x)$ towards zero, and therefore Equation 26 towards Equation 23. This concludes the specification of the prior $p(\boldsymbol{\gamma}|\lambda,\rho,\kappa)$.

The density model defined by Equations 24 to 26 draws on ideas from the large body of literature on GP-based density estimation, for example by Adams et al. (2009), Leonard (1978), or Tokdar et al. (2010), and semiparametric density estimation, e.g. as discussed by Yang (2009), Lenk (2003) or Hjort & Glad (1995). However, note that existing density estimation approaches are not applicable off-the-shelf as in our domain distributions are truncated differently at each observation due to constraints that arise from the way eye movements interact with the text structure (Equations 16 to 19).

4 Inference

To solve Equation 4, we need to integrate for each $r \in \mathcal{R}$ over the reader-specific model $\boldsymbol{\gamma}_r$. To reduce notational clutter, let $\boldsymbol{\gamma} \in \{\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_R\}$ denote a reader-specific model, and let $\mathcal{S} \in \{\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(R)}\}$ denote the eye movement observations of that reader on the training texts $\mathcal{X}$. We approximate

$$\int p(\bar{\mathcal{S}}|\bar{\mathcal{X}},\boldsymbol{\gamma})p(\boldsymbol{\gamma}|\mathcal{X},\mathcal{S})\mathrm{d}\boldsymbol{\gamma} \approx \frac{1}{K}\sum_{k=1}^{K}p(\bar{\mathcal{S}}|\bar{\mathcal{X}},\boldsymbol{\gamma}^{(k)})$$

by a sample $\boldsymbol{\gamma}^{(1)}, \ldots, \boldsymbol{\gamma}^{(K)}$ of models drawn by

$$\boldsymbol{\gamma}^{(k)} \sim p(\boldsymbol{\gamma}|\mathcal{X},\mathcal{S},\lambda,\rho,\kappa),$$

where $p(\boldsymbol{\gamma}|\mathcal{X},\mathcal{S},\lambda,\rho,\kappa)$ is the posterior as given by Equation 6 but with the dependence on the prior hyperparameters λ,ρ,κ made explicit. Note that with $\mathcal{X}$ and $\mathcal{S}$, all saccade types u_t are observed. Together with the factorizing prior (Equation 22), this means that the posterior factorizes according to

$$p(\boldsymbol{\gamma}|\mathcal{X},\mathcal{S},\lambda,\rho,\kappa) = p(\boldsymbol{\pi}|\mathcal{X},\mathcal{S},\lambda)p(\mu|\mathcal{X},\mathcal{S},\rho)$$
$$\cdot\, p(\bar{\alpha}_1|\mathcal{X},\mathcal{S},\kappa)\prod_{i=0}^{4}p(\alpha_i|\mathcal{X},\mathcal{S},\kappa)\prod_{i=0}^{4}p(\delta_i|\mathcal{X},\mathcal{S},\kappa)$$

as is easily seen from the graphical model in Figure 2. Obtaining samples $\boldsymbol{\pi}^{(k)} \sim p(\boldsymbol{\pi}|\mathcal{X},\mathcal{S})$ and $\mu^{(k)} \sim p(\mu|\mathcal{X},\mathcal{S})$ is straightforward because their prior distributions are conjugate to the likelihood terms. Let now $f \in \{\bar{\alpha}_1, \alpha_0, ..., \alpha_4, \delta_0, ..., \delta_4\}$

denote a particular density in the model. The posterior $p(f|\mathcal{X}, \mathcal{S}, \kappa)$ is proportional to the prior $p(f|\kappa)$ (Equation 24) multiplied by the likelihood of all observations that are generated by this density, that is, that are generated according to Equation 14, 16, 17, 18, 19, 20, or 21. Let $\mathbf{y} = (y_1, \ldots, y_{|\mathbf{y}|})^\mathsf{T} \in \mathbb{R}^{|\mathbf{y}|}$ denote the vector of all observations generated from density f, and let $\mathbf{l} = (l_1, \ldots, l_{|\mathbf{l}|})^\mathsf{T} \in \mathbb{R}^{|\mathbf{l}|}$, $\mathbf{r} = (r_1, \ldots, r_{|\mathbf{r}|})^\mathsf{T} \in \mathbb{R}^{|\mathbf{r}|}$ denote the corresponding left and right boundaries of the truncation intervals (again see Equations 14 to 21), where for densities that are not truncated we take $l_i = 0$ and $r_i = \infty$ throughout. Then the likelihood of the observations generated from f is

$$p(\mathbf{y}|f, \mathbf{l}, \mathbf{r}) = \prod_{i=1}^{|\mathbf{y}|} f(y_i | y_i \in [l_i, r_i]) \qquad (27)$$

and the posterior over f is given by

$$p(f|\mathcal{X}, \mathcal{S}, \kappa) \propto p(f|\kappa) p(\mathbf{y}|f, \mathbf{l}, \mathbf{r}). \qquad (28)$$

Note that $\mathbf{y}$, $\mathbf{l}$ and $\mathbf{r}$ are observable from $\mathcal{X}$, $\mathcal{S}$.

We obtain samples from the posterior given by Equation 28 from a Metropolis-Hastings sampler that explores the space of densities $f : \mathbb{R} \to \mathbb{R}$, generating density samples $f^{(1)}, ..., f^{(K)}$. A density f is given by a combination of gamma parameters $\boldsymbol{\eta} \in \mathbb{R}^2$ and function $g : \mathbb{R} \to \mathbb{R}$; specifically, f is obtained by multiplying the gamma distribution with parameters $\boldsymbol{\eta}$ by $\exp(g)$ and normalizing appropriately (Equation 26). During sampling, we explicitly represent a density sample $f^{(k)}$ by its gamma parameters $\boldsymbol{\eta}^{(k)}$ and function $g^{(k)}$. The proposal distribution of the Metropolis-Hastings sampler is

$$q(\boldsymbol{\eta}^{(k+1)}, g^{(k+1)} | \boldsymbol{\eta}^{(k)}, g^{(k)}) = p(g^{(k+1)}|\kappa) \mathcal{N}(\boldsymbol{\eta}^{(k+1)} | \boldsymbol{\eta}^{(k)}, \sigma^2 \mathbf{I})$$

where $p(g^{(k+1)}|\kappa)$ is the probability of $g^{(k+1)}$ according to the GP prior $\mathcal{GP}(0, \kappa)$ (Equation 25), and $\mathcal{N}(\boldsymbol{\eta}^{(k+1)} | \boldsymbol{\eta}^{(k)}, \sigma^2 \mathbf{I})$ is a symmetric proposal that randomly perturbs the old state $\boldsymbol{\eta}^{(k)}$ according to a Gaussian. In every iteration k a proposal $\boldsymbol{\eta}^\star, g^\star \sim q(\boldsymbol{\eta}, g | \boldsymbol{\eta}^{(k)}, g^{(k)})$ is drawn based on the old state $(\boldsymbol{\eta}^{(k)}, g^{(k)})$. The acceptance probability is $A(\boldsymbol{\eta}^\star, g^\star | \boldsymbol{\eta}^{(k)}, g^{(k)}) = \min(1, Q)$ with

$$Q = \frac{q(\boldsymbol{\eta}^{(k)}, g^{(k)} | \boldsymbol{\eta}^\star, g^\star) p(\boldsymbol{\eta}^\star) p(g^\star|\kappa) p(\mathbf{y}|f^\star, \mathbf{l}, \mathbf{r})}{q(\boldsymbol{\eta}^\star, g^\star | \boldsymbol{\eta}^{(k)}, g^{(k)}) p(\boldsymbol{\eta}^{(k)}) p(g^{(k)}|\kappa), p(\mathbf{y}|f^{(k)}, \mathbf{l}, \mathbf{r})}.$$

Here, $p(\boldsymbol{\eta}^\star)$ is the prior probability of gamma parameters $\boldsymbol{\eta}^\star$ (Section 3.3) and $p(\mathbf{y}|f^\star, \mathbf{l}, \mathbf{r})$ is given by Equation 27 where $f^\star$ is obtained from $\boldsymbol{\eta}^\star$, $g^\star$ according to Equation 26.

To compute the likelihood terms $p(\mathbf{y}|f^{(k)}, \mathbf{l}, \mathbf{r})$ (Equation 27) and also to compute the likelihood of test data under a model (Equation 5), the density $f : \mathbb{R} \to \mathbb{R}$ needs to be evaluated. According to Equation 26, f is represented by parameter vector $\boldsymbol{\eta}$ together with the nonparametric function $g : \mathbb{R} \to \mathbb{R}$. As usual when working with distributions over functions in a Gaussian process framework, the function g only needs to be represented at those points for which we need to evaluate it. Clearly, this includes all observations of saccade amplitudes and fixation durations observed in the training and test set. However, we also need to evaluate the normalizer in Equation 26, and (for $f \in \{\alpha_1, \bar{\alpha}_1, \alpha_2, \alpha_3, \alpha_4\}$) the additional normalizer required when truncating the distribution (see Equation 15). As these integrals are one-dimensional, they can be solved relatively accurately using numerical integration; we use 2-point Newton-Cotes quadrature. Newton-Cotes integration requires the evaluation (and thus representation) of g at an additional set of equally spaced supporting points.

When the set of test observations $\bar{\mathcal{S}}, \bar{\mathcal{X}}$ is large, the need to evaluate $p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \boldsymbol{\gamma}^{(k)})$ for all $\boldsymbol{\gamma}_k$ and all test observations leads to computational challenges. In our experiments, we use a heuristic to reduce computational load. While generating samples, densities are only represented at the training observations and the supporting points needed for Newton-Cotes integration. We then estimate the mean of the posterior by $\hat{\boldsymbol{\gamma}} = \frac{1}{K} \sum_{k=1}^K \boldsymbol{\gamma}^{(k)}$, and approximate $\frac{1}{K} \sum_{k=1}^K p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \boldsymbol{\gamma}^{(k)}) \approx p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \hat{\boldsymbol{\gamma}})$. To evaluate $p(\bar{\mathcal{S}}|\bar{\mathcal{X}}, \hat{\boldsymbol{\gamma}})$, we infer the approximate value of the density $\hat{\boldsymbol{\gamma}}$ at a test observation by linearly interpolating based on the available density values at the training observations and supporting points.

5 Empirical Study

We conduct a large-scale study of biometric identification performance using the same setup as discussed by Landwehr et al. (2014) but a much larger set of individuals (251 rather than 20).

Eye movement records for 251 individuals are

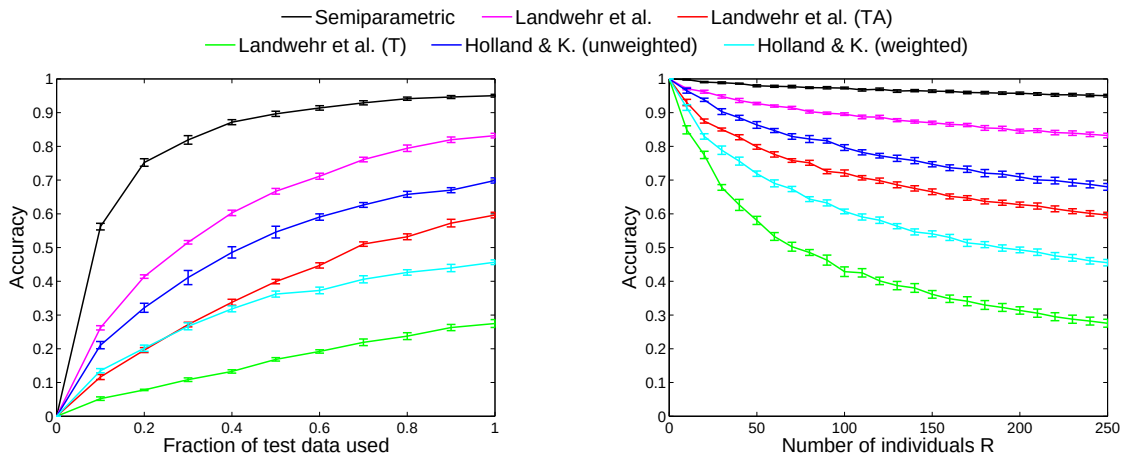


Figure 3: Multiclass accuracy over number of test observations (left) and number of individuals R (right) with standard errors.

Method	Accuracy
Semiparametric	0.9502 ± 0.0130
Semiparametric (TD)	0.8853 ± 0.0142
Semiparametric (TA)	0.7717 ± 0.0361
Landwehr et al.	0.8319 ± 0.0218
Landwehr et al. (TA)	0.5964 ± 0.0262
Landwehr et al. (T)	0.2749 ± 0.0369
Holland & K. (unweighted)	0.6988 ± 0.0241
Holland & K. (weighted)	0.4566 ± 0.0220

Table 1: Multiclass identification accuracy $\pm$ standard error.

obtained from an EyeLink II system with a 500-Hz sampling rate (SR Research, Ongoode, Ontario, Canada) while reading sentences from the *Potsdam Sentence Corpus* (Kliegl et al., 2006). There are 144 sentences in the corpus, which we split into equally sized sets of training and test sentences. Individuals read between 100 and 144 sentences, the training (testing) observations for one individual are the observations on those sentences in the training (testing) set of sentences that the individual has read. Results are averaged over 10 random train-test splits. Each sentence is shown as a single line on the screen.

We study the semiparametric model discussed in Section 3 with MCMC inference as presented in Section 4 (denoted *Semiparametric*[1]). We employ a squared exponential covariance function $\kappa(x, x') = \alpha \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$, where the multiplicative constant α is tuned on the training data by cross-validation and the bandwidth σ is set to the average distance between points in the training data. The Beta and Dirichlet parameters λ and ρ are set to one (Laplace smoothing), the prior $p(\boldsymbol{\eta})$ for the Gamma parameters is uninformative. We use backoff-smoothing as discussed by Landwehr et al. (2014). We initialize the sampler with the maximum-likelihood Gamma fit and perform 10000 sampling iterations, 5000 of which are burn-in iterations. As a baseline, we study the model by Landwehr et al. (2014) (*Landwehr et al.*) and simplified versions proposed by them that only use saccade type and amplitude (*Landwehr et al. (TA)*) or saccade type (*Landwehr et al. (T)*). We also study the weighted and unweighted version of the feature-based model of Holland & Komogortsev (2012) with a feature set adapted to the Potsdam Sentence Corpus data as described in Landwehr et al. (2014).

We note that there are two recent extensions of the feature-based model (by Rigas et al. (2016) and Abdulin & Komogortsev (2015)) that are unfortunately not applicable in our empirical setting but might yield improved results in other scenarios. Rigas et al. (2016) study a model that is focused on representing reader-specific differences in saccadic vigor and acceleration, which are both derived from the dynamics of saccadic velocity. In the preprocessed data set that we use, saccadic velocities are not available, therefore we do not make use of velocities in our model and cannot easily compare against their model. Abdulin & Komogortsev (2015) study a model that is based on features that relate eye move-

[1] An implementation is available at `github.com/abdelwahab/SemiparametricIdentification`

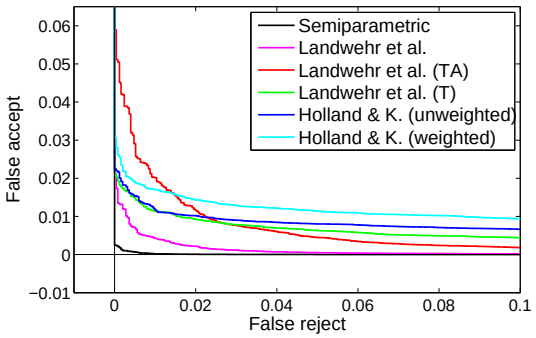


Figure 4: False-accept over false-reject rate when varying τ.

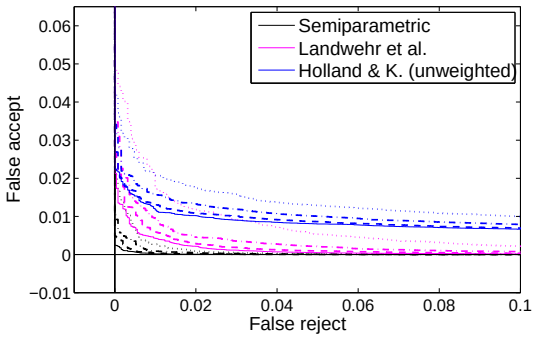


Figure 5: False-accept over false-reject rate when using 40% (dotted), 60% (dashed-dotted), 80% (dashed), and 100% (solid) of test observations, for selected subset of methods.

ments to the 2D text structure, that is, to the way words are arranged into lines in a text. As in our empirical study each sentence is presented as a single line on screen, this 2D structure does not exist. Moreover, Abdulin & Komogortsev (2015) only report accuracy improvements for their method in a setting where individuals have to be identified in the future based on data collected in the past (*aging test*), which is not the focus of our study.

We first study multiclass identification accuracy. All test observations of one particular individual constitute one test example; the task is to infer the individual that has generated these test observations. Multiclass identification accuracy is the fraction of cases in which the correct individual is identified. Table 1 shows multiclass identification accuracy for all methods, including variants of *Semiparametric* discussed below. We observe that *Semiparametric* outperforms *Landwehr et al.*, reducing the error by more than a factor of three. Consistent with results reported in Landwehr et al. (2014), *Holland & K. (unweighted)* is less accurate than *Landwehr et al.*, but more accurate than the simplified variants. We next study how the amount of data available at test time—that is, the amount of time we can observe a reader before having to make a decision—influences accuracy. Figure 3 (left) shows identification accuracy as a function of the fraction of test data available, obtained by randomly removing a fraction of sentences from the test set. We observe that identification accuracy steadily improves with more test observations for all methods. Figure 3 (right) shows identification accuracy when varying the number R of individuals that need to be distinguished. We randomly draw a subset of R individuals from the set

Method	Area under curve
Semiparametric	0.0000119
Semiparametric (TD)	0.0000821
Semiparametric (TA)	0.0001833
Landwehr et al.	0.0001743
Landwehr et al. (TA)	0.0010371
Landwehr et al. (T)	0.0017040
Holland & K. (unweighted)	0.0027853
Holland & K. (weighted)	0.0039978

Table 2: Area under the curve in binary classification setting.

of 251 individuals, and perform identification based on only these individuals. Results are averaged over 10 such random draws. As expected, accuracy improves if fewer individuals need to be distinguished.

We next study a binary setting in which for each individual and each set of test observations a decision has to be made whether or not the test observations have been generated by that individual. This setting more closely matches typical use cases for the deployment of a biometric system. Let $\bar{\mathcal{X}}$ denote the text being read at test time, and let $\bar{\mathcal{S}}$ denote the observed eye movement sequences. Our model infers for each reader $r \in \mathcal{R}$ the marginal likelihood $p(\bar{\mathcal{S}}|r, \bar{\mathcal{X}}, \mathcal{X}, \mathcal{S}^{(1:R)})$ of the eye movement observations under the reader-specific model (Equation 3). The binary decision is made by dividing this marginal likelihood by the average marginal likelihood assigned to the observations by all reader-specific models, and comparing the result to a threshold τ. Figure 4 shows the fraction of false accepts as a function of false rejects as the threshold τ is varied, averaged over all individuals. The *Landwehr et al.* model and variants also assign a

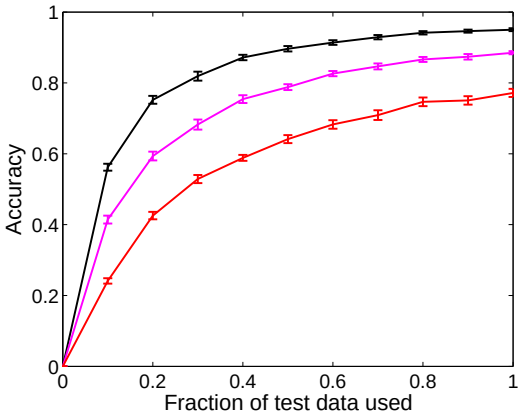


Figure 6: Multiclass accuracy over number of test observations with standard errors for *Semiparametric* variants.

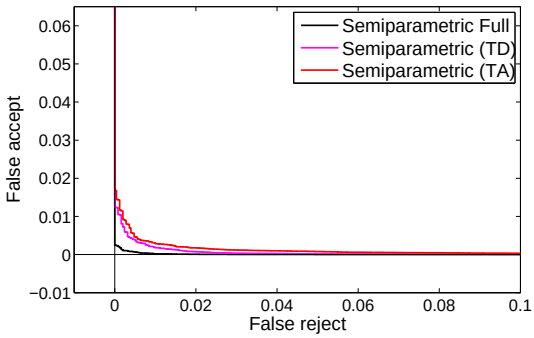


Figure 7: False-accept over false-reject rate when varying τ for the *Semiparametric* variants.

reader-specific likelihood to novel test observations; we compute the same statistics again by normalizing the likelihood and comparing to a threshold τ. Finally, *Holland & K. (unweighted)* and *Holland & K. (weighted)* compute a similarity measure for each combination of individual and set of test observations, which we normalize and threshold analogously. We observe that *Semiparametric* accomplishes a false-reject rate of below 1% at virtually no false accepts; *Landwehr et al.* and variants tend to perform better than *Holland & K. (unweighted)* and *Holland & K. (weighted)*. Table 2 shows the error under the curve for the experiment shown in Figure 4, as well as for variants of *Semiparametric* discussed below.

We finally study the contribution of the individual model components for saccade type, saccade amplitude, and fixation duration (see Figure 2) by removing the corresponding model components, as in Landwehr et al. (2014). By *Semiparametric (TD)* we denote a variant of *Semiparametric* in which the variable a_t and the corresponding distribution is removed, that is, only the distribution over the saccade type and duration is modeled. *Semiparametric (TA)* denotes a variant in which the variable d_t and the corresponding distribution is removed. Figure 6 shows identification accuracy as a function of the fraction of test data available for model variants *Semiparametric (TD)* and *Semiparametric (TA)* in comparison to *Semiparametric*; results for these variants are also included in Table 1. Figure 7 shows the fraction of false accepts as a function of false rejects in the binary classification setting discussed above for these two model variants; Table 2 includes area under the curve results for the experiment shown in Figure 7. We observe that accuracy is substantially reduced when removing any model component. Note that if both the amplitude and duration components of the model are removed, it becomes identical to the model *Landwehr et al. (T)*.

Training the joint model for all 251 individuals takes 46 hours on a single eight-core CPU (Intel Xeon E5520, 2.27GHz); predicting the most likely individual to have generated a set of 72 test sentences takes less than 2 seconds.

6 Conclusions

We have studied the problem of identifying readers unobtrusively during reading of arbitrary text. For fitting reader-specific distributions, we employ a Bayesian semiparametric approach that infers densities under a Gaussian process prior centered at the gamma family of distributions, striking a balance between robustness to sparse data and modeling flexibility. In an empirical study with 251 individuals, the model was shown to reduce identification error by more than a factor of three compared to earlier approaches to reader identification proposed by Landwehr et al. (2014) and Holland & Komogortsev (2012).

Acknowledgements

We gratefully acknowledge support from the German Research Foundation (DFG), grant LA 3270/1-1.

References

Evgeniy Abdulin and Oleg Komogortsev. 2015. Person verification via eye movement-driven text reading model. In *Proceedings of the Sixth International Conference on Biometrics: Theory, Applications and Systems*.

Ryan P. Adams, Iain Murray, and David J.C. MaxKay. 2009. Gaussian process density sampler. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*.

Roman Bednarik, Tomi Kinnunen, Andrei Mihaila, and Pasi Fränti. 2005. Eye-movements as a biometric. In *Proceedings of the 14th Scandinavian Conference on Image Analysis*.

W. Robert Dixon. 1951. Studies in the psychology of reading. In W. S. Morse, P. A. Ballantine, and W. R. Dixon, editors, *Univ. of Michigan Monographs in Education No. 4*. Univ. of Michigan Press.

Ralf Engbert, Antje Nuthmann, Eike M. Richter, and Reinhold Kliegl. 2005. SWIFT: A dynamical model of saccade generation during reading. *Psychological Review*, 112(4):777–813.

Tadayoshi Hara, Daichi Mochihashi, Yoshino Kano, and Akiko Aizawa. 2012. Predicting word fixations in text with a CRF model for capturing general reading strategies among readers. In *Proceedings of the First Workshop on Eye-Tracking and Natural Language Processing*.

Julian Heister, Kay-Michael Würzner, and Reinhold Kliegl. 2012. Analysing large datasets of eye movements during reading. In James S. Adelman, editor, *Visual word recognition. Vol. 2: Meaning and context, individuals and development*, pages 102–130.

Nils L. Hjort and Ingrid K. Glad. 1995. Nonparametric density estimation with a parametric start. *The Annals of Statistics*, 23(3):882–904.

Corey Holland and Oleg V. Komogortsev. 2012. Biometric identification via eye movement scanpaths in reading. In *Proceedings of the 2011 International Joint Conference on Biometrics*.

Edmund B. Huey. 1908. *The psychology and pedagogy of reading*. Cambridge, Mass.: MIT Press.

Pawel Kasprowski and Jozef Ober. 2004. Eye movements in biometrics. In *Proceedings of the 2004 International Biometric Authentication Workshop*.

Reinhold Kliegl, Antje Nuthmann, and Ralf Engbert. 2006. Tracking the mind during reading: The influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General*, 135(1):12–35.

Oleg V. Komogortsev, Sampath Jayarathna, Cecilia R. Aragon, and Mechehoul Mahmoud. 2010. Biometric identification via an oculomotor plant mathematical model. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*.

Niels Landwehr, Sebastian Arzt, Tobias Scheffer, and Reinhold Kliegl. 2014. A model of individual differences in gaze control during reading. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*.

Peter J. Lenk. 2003. Bayesian semiparametric density estimation and model verification using a logistic-Gaussian process. *Journal of Computational and Graphical Statistics*, 12(3):548–565.

Tom Leonard. 1978. Density estimation, stochastic processes and prior information. *Journal of the Royal Statistical Society*, 40(2):113–146.

Franz Matties and Anders Søgaard. 2013. With blinkers on: robust prediction of eye movements across readers. In *Proceedings of the 2013 Conference on Empirical Natural Language Processing*.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Erik D. Reichle, Alexander Pollatsek, Donald L. Fisher, and Keith Rayner. 1998. Toward a model of eye movement control in reading. *Psychological Review*, 105(1):125–157.

Ioannis Rigas, George Economou, and Spiros Fotopoulos. 2012a. Biometric identification based on the eye movements and graph matching techniques. *Pattern Recognition Letters*, 33(6).

Ioannis Rigas, George Economou, and Spiros Fotopoulos. 2012b. Human eye movements as a trait for biometrical identification. In *Proceedings of the IEEE 5th International Conference on Biometrics: Theory, Applications and Systems*.

Ioannis Rigas, Oleg Komogortsev, and Reza Shadmehr. 2016. Biometric recognition via eye movements: Saccadic vigor and acceleration cues. *ACM Transaction on Applied Perception*, 13(2):1–21.

Surya T. Tokdar, Yu M. Zhuy, and Jayanta K. Ghoshz. 2010. Bayesian density regression with logistic gaussian process and subspace projection. *Bayesian Analysis*, 5(2):319–344.

Ying Yang. 2009. Penalized semiparametric density estimation. *Statistics and Computing*, 19(1):355–366.

Youming Zhang and Martti Juhola. 2012. On biometric verification of a user by means of eye movement data mining. In *Proceedings of the 2nd International Conference on Advances in Information Mining and Management*.

Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora

William L. Hamilton, Kevin Clark, Jure Leskovec, Dan Jurafsky
Department of Computer Science, Stanford University, Stanford CA, 94305
wleif,kevclark,jure,jurafsky@stanford.edu

Abstract

A word's sentiment depends on the domain in which it is used. Computational social science research thus requires sentiment lexicons that are specific to the domains being studied. We combine domain-specific word embeddings with a label propagation framework to induce accurate domain-specific sentiment lexicons using small sets of seed words, achieving state-of-the-art performance competitive with approaches that rely on hand-curated resources. Using our framework we perform two large-scale empirical studies to quantify the extent to which sentiment varies across time and between communities. We induce and release historical sentiment lexicons for 150 years of English and community-specific sentiment lexicons for 250 online communities from the social media forum Reddit. The historical lexicons show that more than 5% of sentiment-bearing (non-neutral) English words completely switched polarity during the last 150 years, and the community-specific lexicons highlight how sentiment varies drastically between different communities.

1 Introduction

Inducing domain-specific sentiment lexicons is crucial to computational social science (CSS) research. Sentiment lexicons allow us to analyze key subjective properties of texts like opinions and attitudes (Taboada et al., 2011). But lexical sentiment is hugely influenced by context. The word *soft* has a very different sentiment in an online sports community than it does in one dedicated to toy animals (Figure 1). *Terrific* once had a highly negative connotation; now it is essentially synonymous with *good* (Figure 2). Without domain-specific lexicons, social scientific analyses can be misled by sentiment assignments biased towards domain-general contexts, neglecting factors like genre, community-specific vernacular, or demographic variation (Deng et al., 2014; Hovy, 2015; Yang and Eisenstein, 2015).

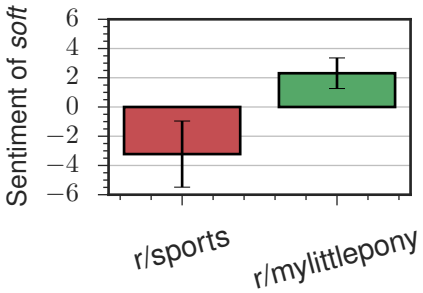


Figure 1: The sentiment of *soft* in different online communities. Sentiment values computed using SENTPROP (Section 3) on comments from Reddit communities illustrate how sentiment depends on social context. Bootstrap-sampled standard deviations provide a measure of confidence with the scores.

Using experts or crowdsourcing to construct domain-specific sentiment lexicons is expensive and often time-consuming (Mohammad and Turney, 2010; Fast et al., 2016), and is especially problematic when non-standard language (as in historical documents or obscure social media forums) prevents annotators from understanding the sociolinguistic context of the data.

Web-scale sentiment lexicons can be automatically induced for large socially-diffuse domains, such as the internet-at-large (Velikovich et al., 2010) or all of Twitter (Tang et al., 2014). However, to study sentiment in domain-specific cases—financial documents, historical texts, or tight-knit social me-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 595–605,
Austin, Texas, November 1-5, 2016.

dia forums—such generic lexicons may be inaccurate, and even introduce harmful biases (Loughran and McDonald, 2011).[1] Researchers need a principled and accurate framework for inducing lexicons that are specific to their domain of study.

To meet these needs, we introduce SENTPROP, a framework to learn accurate sentiment lexicons from small sets of seed words and domain-specific corpora. SENTPROP combines the well-known method of label propagation with advances in word embeddings, and unlike previous approaches, is designed to be accurate even when using modestly-sized domain-specific corpora ($\sim 10^7$ tokens). Our framework also provides *confidence scores* along with the learned lexicons, which allows researchers to quantify uncertainty in a principled manner.

The key contributions of this work are:

1. A simple state-of-the-art sentiment induction algorithm, combining high-quality word vector embeddings with a label propagation approach.
2. A novel bootstrap-sampling framework for inferring confidence scores with the sentiment values.
3. Two large-scale studies that reveal how sentiment depends on both social and historical context.
   (a) We induce community-specific sentiment lexicons for the largest 250 "subreddit" communities on the social-media forum Reddit, revealing substantial variation in word sentiment between communities.
   (b) We induce historical sentiment lexicons for 150 years of English, revealing that $>5\%$ of words switched polarity during this time.

To the best of our knowledge, this is the first work to systematically analyze the domain-dependency of sentiment at a large-scale, across hundreds of years and hundreds of user-defined online communities.

All of the inferred lexicons along with code for SENTPROP and all methods evaluated are made available in the SOCIALSENT package released with this paper.[2] The SOCIALSENT package provides a benchmark toolkit for inducing sentiment lexicons, including implementations of previously published algorithms (Velikovich et al., 2010; Rothe et al., 2016), which are not otherwise publicly available.

[1] http://brandsavant.com/brandsavant/the-hidden-bias-of-social-media-sentiment-analysis

[2] http://nlp.stanford.edu/projects/socialsent

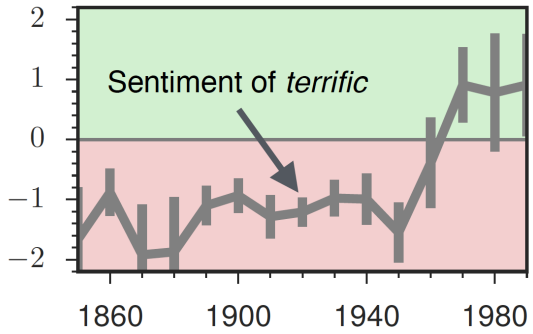


Figure 2: The sentiment of *terrific* changed from negative to positive over the last 150 years. Sentiment values and bootstrapped confidences were computed using SENTPROP on historical data (see Section 6).

2 Related work

Our work builds upon a wealth of previous research on inducing sentiment lexicons, along two threads:

Corpus-based approaches use seed words and patterns in unlabeled corpora to induce domain-specific lexicons. These patterns may rely on syntactic structures (Hatzivassiloglou and McKeown, 1997; Jijkoun et al., 2010; Rooth et al., 1999; Thelen and Riloff, 2002; Widdows and Dorow, 2002), which can be domain-specific and brittle (e.g., in social media lacking usual grammatical structures). Other models rely on general co-occurrence (Igo and Riloff, 2009; Riloff and Shepherd, 1997; Turney and Littman, 2003). Often corpus-based methods exploit distant-supervision signals (e.g., review scores, emoticons) specific to certain domains (Asghar et al., 2015; Blair-Goldensohn et al., 2008; Bravo-Marquez et al., 2015; Choi and Cardie, 2009; Severyn and Moschitti, 2015; Speriosu et al., 2011; Tang et al., 2014). An effective corpus-based approach that does not require distant-supervision—which we adapt here—is to construct lexical graphs using word co-occurrences and then to perform some form of label propagation over these graphs (Huang et al., 2014; Velikovich et al., 2010). Recent work has also learned transformations of word-vector representations in order to induce sentiment lexicons (Rothe et al., 2016). Fast et al. (2016) combine word vectors with crowdsourcing to produce domain-independent topic lexicons.

Dictionary-based approaches use hand-curated lexical resources—usually WordNet (Fellbaum, 1998)—in order to propagate sentiment from seed

Domain	Positive seed words	Negative seed words
Standard English	good, lovely, excellent, fortunate, pleasant, delightful, perfect, loved, love, happy	bad, horrible, poor, unfortunate, unpleasant, disgusting, evil, hated, hate, unhappy
Finance	successful, excellent, profit, beneficial, improving, improved, success, gains, positive	negligent, loss, volatile, wrong, losses, damages, bad, litigation, failure, down, negative
Twitter	love, loved, loves, awesome, nice, amazing, best, fantastic, correct, happy	hate, hated, hates, terrible, nasty, awful, worst, horrible, wrong, sad

Table 1: Seed words. The seed words were manually selected to be context insensitive (without knowledge of the test lexicons).

labels (Esuli and Sebastiani, 2006; Hu and Liu, 2004; Kamps et al., 2004; Rao and Ravichandran, 2009; San Vicente et al., 2014; Takamura et al., 2005; Tai and Kao, 2013). There is an implicit consensus that dictionary-based approaches will generate higher-quality lexicons, due to their use of these clean, hand-curated resources; however, they are not applicable in domains lacking such a resource (e.g., most historical texts).

Most previous work seeks to enrich or enlarge existing lexicons (Qiu et al., 2009; San Vicente et al., 2014; Velikovich et al., 2010), emphasizing recall over precision. This recall-oriented approach is motivated by the need for massive polarity lexicons in tasks like web-advertising (Velikovich et al., 2010). In contrast to these previous efforts, the goal of this work is to induce high-quality lexicons that are accurate to a specific social context.

Algorithmically, our approach is inspired by Velikovich et al. (2010). We extend Velikovich et al. (2010) by incorporating high-quality word vector embeddings, a new graph construction approach, an alternative label propagation algorithm, and a bootstrapping method to obtain confidence values. Together these improvements, especially the high-quality word vectors, allow our corpus-based method to even outperform the state-of-the-art dictionary-based approach.

3 Framework

Our framework, SENTPROP, is designed to meet four key desiderata:

1. **Resource-light:** Accurate performance without massive corpora or hand-curated resources.
2. **Interpretable**: Uses small seed sets of "paradigm" words to maintain interpretability and avoid ambiguity in sentiment values.
3. **Robust**: Bootstrap-sampled standard deviations provide a measure of confidence.
4. **Out-of-the-box**: Does not rely on signals that are specific to only certain domains.

SENTPROP involves two steps: constructing a lexical graph from unlabeled corpora and propagating sentiment labels over this graph.

3.1 Constructing a lexical graph

Lexical graphs are constructed from distributional word embeddings learned on unlabeled corpora.

Distributional word embeddings

The first step in our approach is to build high-quality semantic representations for words using a vector space model (VSM). We embed each word $w_i \in \mathcal{V}$ as a vector $\mathbf{w}_i$ that captures information about its co-occurrence statistics with other words (Landauer and Dumais, 1997; Turney and Pantel, 2010). This VSM approach has a long history in NLP and has been highly successful in recent applications (see Levy et al., 2015 for a survey).

When recreating known lexicons, we used a number of publicly available embeddings (Section 4).

In the cases where we learned embeddings ourselves, we employed an SVD-based method to construct the word-vectors. First, we construct a matrix $\mathbf{M}^{PPMI} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ with entries given by

$$\mathbf{M}^{PPMI}_{i,j} = \max\left\{\log\left(\frac{\hat{p}(w_i, w_j)}{\hat{p}(w)\hat{p}(w_j)}\right), 0\right\}, \quad (1)$$

where $\hat{p}$ denotes smoothed empirical probabilities of word (co-)occurrences within fixed-size sliding windows of text.[3] $\mathbf{M}^{PPMI}_{i,j}$ is equal to a smoothed variant of the positive pointwise mutual information between words w_i and w_j (Levy et al., 2015). Next, we compute $\mathbf{M}^{PPMI} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, the truncated singular value decomposition of $\mathbf{M}^{PPMI}$. The vector

[3] We use contexts of size four on each side and context-distribution smoothing with $c = 0.75$ (Levy et al., 2015).

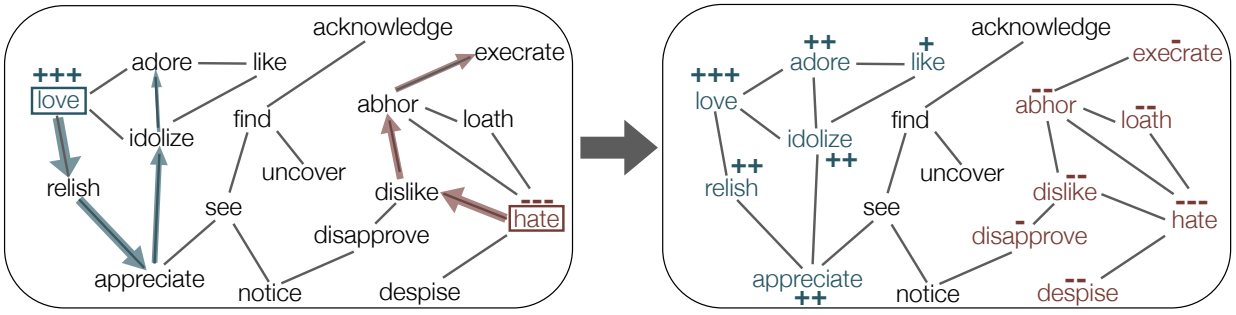


Figure 3: Visual summary of the SENTPROP algorithm.

embedding for word w_i is then given by

$$\mathbf{w}_i^{\text{SVD}} = (\mathbf{U})_i \,. \tag{2}$$

Excluding the singular value weights, Σ, has been shown known to dramatically improve embedding quality (Turney and Pantel, 2010; Bullinaria and Levy, 2012). Following standard practices, we learn embeddings of dimension 300.

We found that this SVD-based method significantly outperformed word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) on preliminary experiments with the domain-specific data we used (see Section 4).

Defining the graph edges

Given a set of word embeddings, a weighted lexical graph is constructed by connecting each word with its nearest k neighbors within the semantic space (according to cosine-similarity). The weights of the edges are set as

$$\mathbf{E}_{i,j} = \arccos\left(-\frac{\mathbf{w_i}^\top \mathbf{w_j}}{\|\mathbf{w_i}\|\|\mathbf{w_j}\|}\right). \tag{3}$$

3.2 Propagating polarities from a seed set

Once a weighted lexical graph is constructed, we propagate sentiment labels over this graph using a random walk method (Zhou et al., 2004). A word's polarity score for a seed set is proportional to the probability of a random walk from the seed set hitting that word (Figure 3).

Let $\mathbf{p} \in \mathbb{R}^{|\mathcal{V}|}$ be a vector of word-sentiment scores constructed using seed set $\mathcal{S}$ (e.g., ten negative words); $\mathbf{p}$ is initialized to have $\frac{1}{|\mathcal{V}|}$ in all entries. And let $\mathbf{E}$ be the matrix of edge weights given by equation (3). First, we construct a symmetric transition matrix from $\mathbf{E}$ by computing $\mathbf{T} = \mathbf{D}^{\frac{1}{2}}\mathbf{E}\mathbf{D}^{\frac{1}{2}}$, where $\mathbf{D}$ is a matrix with the column sums of $\mathbf{E}$ on the diagonal. Next, using $\mathbf{T}$ we iteratively update $\mathbf{p}$ until numerical convergence:

$$\mathbf{p}^{(t+1)} = \beta\mathbf{T}\mathbf{p}^{(t)} + (1-\beta)\mathbf{s}, \tag{4}$$

where $\mathbf{s}$ is a vector with values set to $\frac{1}{|\mathcal{S}|}$ in the entries corresponding to the seed set $\mathcal{S}$ and zeros elsewhere. The β term controls the extent to which the algorithm favors local consistency (similar labels for neighbors) vs. global consistency (correct labels on seed words), with lower βs emphasizing the latter.

To obtain a final polarity score for a word w_i, we run the walk using both positive and negative seed sets, obtaining positive ($\mathbf{p}^P(w_i)$) and negative ($\mathbf{p}^N(w_i)$) label scores. We then combine these values into a positive-polarity score as $\bar{\mathbf{p}}^P(w_i) = \frac{\mathbf{p}^P(w_i)}{\mathbf{p}^P(w_i)+\mathbf{p}^N(w_i)}$ and standardize the final scores to have zero mean and unit variance (within a corpus).

3.3 SENTPROP variants

Many variants of the random walk approach and related label propagation techniques exist in the literature (San Vicente et al., 2014; Velikovich et al., 2010; Zhou et al., 2004; Zhu and Ghahramani, 2002; Zhu et al., 2003). For example, there are differences in how to normalize the transition matrix in the random walks (Zhou et al., 2004) and variants of label propagation, e.g. where the labeled seeds are clamped to the correct values (Zhu and Ghahramani, 2002) or where only shortest-paths through the graph are used for propagation (Velikovich et al., 2010).

We experimented with a number of these approaches and found little difference in their performance. We opted to use the random walk method because it had a slight edge in terms of performance

in preliminary experiments[4] and because it produces well-behaved distributions over label scores, whereas Zhu and Ghahramani (2002)'s method and its variants produce extremely peaked distributions. We do note report in detail on all the label propagation variants here, but the SOCIALSENT package contains a full suite of these methods.

3.4 Bootstrap-sampling for robustness

Propagated sentiment scores are inevitably influenced by the seed set, and it is important for researchers to know the extent to which polarity values are simply the result of corpus artifacts that are correlated with these seeds words. We address this issue by using a bootstrap-sampling approach to obtain confidence regions over our sentiment scores. We bootstrap by running our propagation over B random equally-sized subsets of the positive and negative seed sets. Computing the standard deviation of the bootstrap-sampled polarity scores provides a measure of confidence and allows the researcher to evaluate the robustness of the assigned polarities. We set $B = 50$ and used 7 words per random subset (full seed sets are size 10; see Table 1).

4 Recreating known lexicons

We validate our approach by recreating known sentiment lexicons in the three domains: Standard English, Twitter, and Finance. Table 1 lists the seed words used in each domain.

Standard English: To facilitate comparison with previous work, we focus on the well-known General Inquirer lexicon (Stone et al., 1966). We also use the continuous valence (i.e., polarity) scores collected by Warriner et al. (2013) in order to evaluate the fine-grained performance of our framework. We test our framework's performance using two different embeddings: off-the-shelf Google news embeddings constructed from 10^{11} tokens[5] and embeddings we constructed from the 2000s decade of the Corpus of Historical American English (COHA), which contains $\sim 2 \times 10^7$ words in each decade, from 1850 to 2000 (Davies, 2010). The COHA corpus allows us to test how the algorithms deal with this smaller historical corpus, which is important since we will use the COHA corpus to infer historical sentiment lexicons (Section 6).

Finance: Previous work found that general purpose sentiment lexicons performed very poorly on financial text (Loughran and McDonald, 2011), so a finance-specific sentiment lexicon (containing binary labels) was hand-constructed for this domain (ibid.). To test against this lexicon, we constructed embeddings using a dataset of $\sim 2 \times 10^7$ tokens from financial 8K documents (Lee et al., 2014).

Twitter: Numerous works attempt to induce Twitter-specific sentiment lexicons using supervised approaches and features unique to that domain (e.g., follower graphs; Speriosu et al., 2011). Here, we emphasize that we can induce an accurate lexicon using a simple domain-independent and resource-light approach, with the implication that lexicons can easily be induced for related social media domains without resorting to complex supervised frameworks. We evaluate our approach using the test set from the 2015 SemEval task 10E competition (Rosenthal et al., 2015), and we use the embeddings constructed by Rothe et al. (2016).[6]

4.1 Baselines and state-of-the-art comparisons

We compare SENTPROP against standard baselines and state-of-the-art approaches. The PMI baseline of Turney and Littman (2003) computes the pointwise mutual information between the seeds and the targets without using propagation. The baseline method of Velikovich et al. (2010) is similar to our method but uses an alternative propagation approach and raw co-occurrence vectors instead of learned embeddings. Both these methods require raw corpora, so they function as baselines in cases where we do not use off-the-shelf embeddings. We also compare against DENSIFIER, a state-of-the-art method that learns orthogonal transformations of word vectors instead of propagating labels (Rothe et al., 2016). Lastly, on standard English we compare against a state-of-the-art WordNet-based method, which performs label propagation over a WordNet-derived graph (San Vicente et al., 2014). Several variant baselines, all of which SENTPROP

[4] $>2\%$ improvement across metrics on the standard and historical English datasets described in Section 4.

[5] `https://code.google.com/p/word2vec/`

[6] The official SemEval task 10E involved fully-supervised learning, so we do not use their evaluation setup.

Method	AUC	Ternary F1	τ
SENTPROP	90.6	58.6	0.44
DENSIFIER	**93.3**	**62.1**	**0.50**
WordNet	89.5	58.7	0.34
Majority	–	24.8	–

(a) Corpus methods outperform WordNet on standard English. Using word-vector embeddings learned on a massive corpus (10^{11} tokens), we see that both corpus-based methods outperform the WordNet-based approach overall.

Method	AUC	Ternary F1	τ
SENTPROP	86.0	**60.1**	0.50
DENSIFIER	**90.1**	59.4	**0.57**
Sentiment140	86.2	57.7	0.51
Majority	–	24.9	–

(b) Corpus approaches are competitive with a distantly supervised method on Twitter. Using Twitter embeddings learned from $\sim 10^9$ tokens, we see that the semi-supervised corpus approaches using small seed sets perform very well.

Method	AUC	Ternary F1
SENTPROP	**91.6**	**63.1**
DENSIFIER	80.2	50.3
PMI	86.1	49.8
Velikovich et al. (2010)	81.6	51.1
Majority	–	23.6

(c) SENTPROP performs best with domain-specific finance embeddings. Using embeddings learned from financial corpus ($\sim 2 \times 10^7$ tokens) , SENTPROP significantly outperforms the other methods.

Method	AUC	Ternary F1	τ
SENTPROP	**83.8**	**53.0**	**0.28**
DENSIFIER	77.4	46.6	0.19
PMI	70.6	41.9	0.16
Velikovich et al. (2010)	52.7	32.9	0.01
Majority	–	24.3	–

(d) SENTPROP performs well on standard English even with 1000x reduction in corpus size. SENTPROP maintains strong performance even when using embeddings learned from the 2000s decade of COHA (only $2 \times \sim 10^7$ tokens).

Table 2: Results on recreating known lexicons.

outperforms, are omitted for brevity (e.g., using word-vector cosines in place of PMI in Turney and Littman (2003)'s framework). Code for all these variants is available in the SOCIALSENT package.

4.2 Evaluation setup

We evaluate the approaches according to (i) their binary classification accuracy (ignoring the neutral class, as is common in previous work), (ii) ternary classification performance (positive vs. neutral vs. negative)[7], and (iii) Kendall τ rank-correlation with continuous human-annotated polarity scores.

For all methods in the ternary-classification condition, we use the class-mass normalization method (Zhu et al., 2003) to label words as positive, neutral, or negative. This method assumes knowledge of the label distribution—i.e., how many positive/negative vs. neutral words there are—and simply assigns labels to best match this distribution.

4.3 Evaluation results

Tables 2a-2d summarize the performance of our framework along with baselines and other state-of-the-art approaches. Our framework significantly outperforms the baselines on all tasks, outperforms a state-of-the-art approach that uses WordNet on standard English (Table 2a), and is competitive with Sentiment140 on Twitter (Table 2b), a distantly-supervised approach that uses signals from emoticons (Mohammad and Turney, 2010). DENSIFIER also performs extremely well, outperforming SENTPROP when off-the-shelf embeddings are used (Tables 2a and 2b). However, SENTPROP significantly outperforms all other approaches when using the domain-specific embeddings (Tables 2c and 2d).

Overall our results show that SENTPROP— a relatively simple method, which combines high-quality word vectors embeddings with standard label propagation — can perform at a state-of-the-art level, even performing competitively with methods relying on hand-curated lexical graphs. Unlike previous published approaches, SENTPROP is able to maintain high accuracy even when modest-sized domain-specific corpora are used. In cases where very large corpora are available and where there is an abundance of training data, DENSIFIER performs extremely well, since it was designed for this sort of setting (Rothe et al., 2016).

[7]Only GI contains words explicitly marked neutral, so for ternary evaluations in Twitter and Finance we sample neutral words from GI to match its neutral-vs-not distribution.

We found that the baseline method of Velikovich et al. (2010), which our method is closely related to, performed relatively poorly with these domain-specific corpora. This indicates that using high-quality word-vector embeddings can have a drastic impact on performance. However, it is worth noting that Velikovich et al. (2010)'s method was designed for high recall with massive corpora, so its poor performance in our regime is not surprising.

Lastly, we found that the choice of embedding method could have a drastic impact. Preliminary experiments on the COHA data showed that using word2vec SGNS vectors (with default settings) instead of our SVD-based embeddings led to a $>40\%$ performance drop for SENTPROP across all measures and a $>10\%$ performance drop for DENSIFIER. It is possible that certain settings of word2vec could perform better, but previous work has shown that SVD-based methods have superior results on smaller datasets and rare-word similarity tasks (Levy et al., 2015; Hamilton et al., 2016), so this result is not surprising.

5 Inducing community-specific lexicons

As a first large-scale study, we investigate how sentiment depends on the social context in which a word is used. It is well known that there is substantial sociolinguistic variation between different communities, whether these communities are defined geographically (Trudgill, 1974) or via underlying sociocultural differences (Labov, 2006). However, no previous work has systematically investigated community-specific variation in word sentiment at a large scale. Yang and Eisenstein (2015) exploit social network structure in Twitter to infer a small number (1-10) of communities and analyzed sentiment variation via a supervised framework. Our analysis extends this line of work by analyzing the sentiment across hundreds of user-defined communities using only unlabeled corpora and a small set of "paradigm" seed words (the Twitter seed words outlined in Table 1).

In our study, we induced sentiment lexicons for the top-250 (by comment-count) subreddits from the social media forum Reddit.[8] We used all the 2014 comment data to induce the lexicons, with words lower cased and comments from bots and deleted users removed.[9] Sentiment was induced for the top-5000 non-stop words in each subreddit (again, by comment-frequency).

5.1 Examining the lexicons

Analysis of the learned lexicons reveals the extent to which sentiment can differ across communities. Figure 4 highlights some words with opposing sentiment in two communities: in `r/TwoXChromosomes` (`r/TwoX`), a community dedicated to female perspectives and gender issues, the words *crazy* and *insane* have negative polarity, which is not true in the `r/sports` community, and, vice-versa, words like *soft* are positive in `r/TwoX` but negative in `r/sports`.

To get a sense of how much sentiment differs across communities in general, we selected a random subset of 1000 community pairs and examined the correlation in their sentiment values for highly sentiment-bearing words (Figure 5). We see that the distribution is noticeably skewed, with many community pairs having highly uncorrelated sentiment values. The 1000 random pairs were selected such that each member of the pair overlapped in at least half of their top-5000 word vocabulary. We then computed the correlation between the sentiments in these community-pairs. Since sentiment is noisy and relatively uninteresting for neutral words, we compute $\tau_{25\%}$, the Kendall-τ correlation over the top-25% most sentiment bearing words shared between the two communities.

Analysis of individual pairs reveals some interesting insights about sentiment and inter-community dynamics. For example, we found that the sentiment correlation between `r/TwoX` and `r/TheRedPill` ($\tau_{25\%} = 0.58$), two communities that hold conflicting views and often attack each other[10], was actually higher than the sentiment correlation between `r/TwoX` and `r/sports` ($\tau_{25\%} = 0.41$), two communities that are entirely unrelated. This result suggests that conflicting communities may have

[8] Subreddits are user-created topic-specific forums.

[9] https://archive.org/details/2015_reddit_comments_corpus

[10] This conflict is well-known on Reddit; for example, both communities mention each others' names along with *fuck*-based profanity in the same comment far more than one would expect by chance ($\chi^2_1 > 6.8$, $p < 0.01$ for both). `r/TheRedPill` is dedicated to male empowerment.

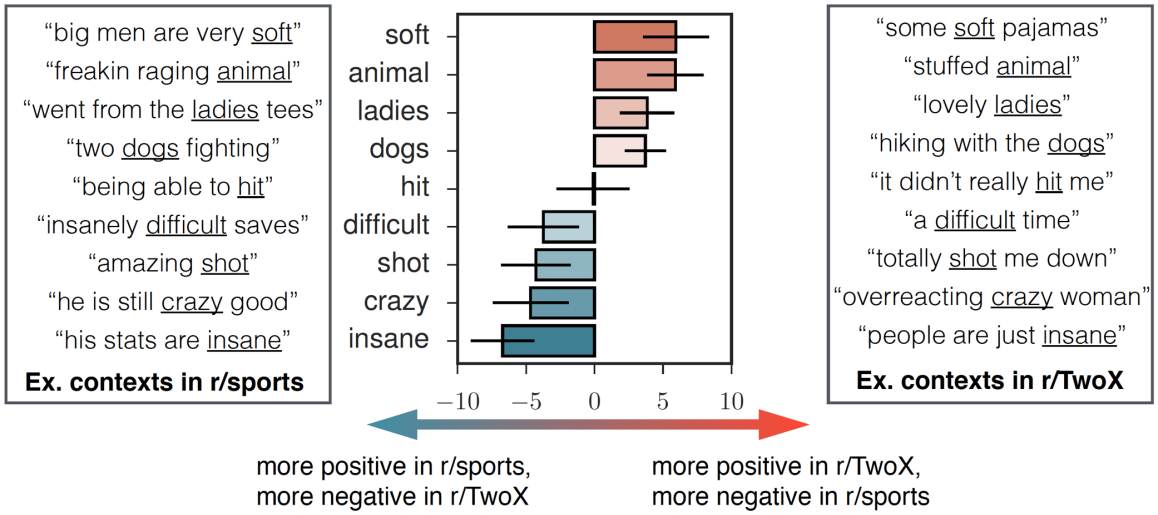


Figure 4: Word sentiment differs drastically between a community dedicated to sports (`r/sports`) and one dedicated to female perspectives and gender issues (`r/TwoX`). Words like *soft* and *animal* have positive sentiment in `r/TwoX` but negative sentiment in `r/sports`, while the opposite holds for words like *crazy* and *insane*.

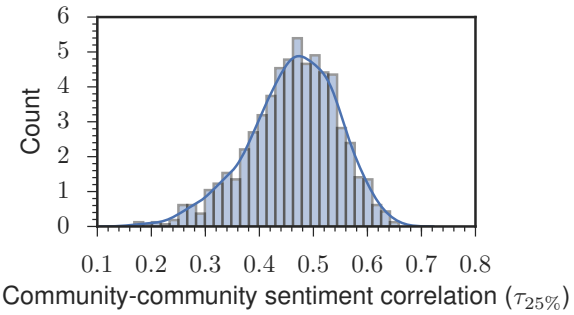


Figure 5: There is a long tail of communities with very different word sentiments. Some communities have very similar sentiment (e.g., `r/sports` and `r/hockey`), while other community pairs differ drastically (e.g., `r/sports` and `r/TwoX`).

more similar sentiment in their language compared to communities that are entirely unrelated.

6 Inducing diachronic sentiment lexicons

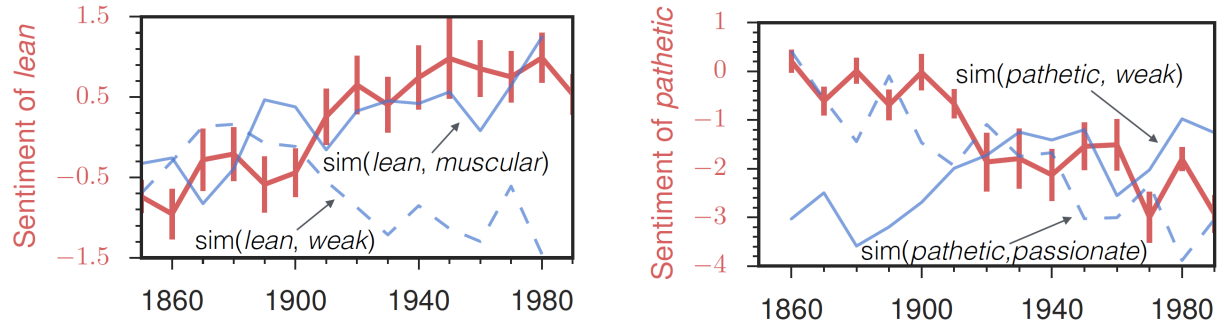
Sentiment also depends on the historical time-period in which a word is used. To investigate this dependency, we use our framework to analyze how word polarities have shifted over the last 150 years. The phenomena of *amelioration* (words becoming more positive) and *pejoration* (words becoming more negative) are well-discussed in the linguistic literature (Traugott and Dasher, 2001); however, no comprehensive polarity lexicons exist for historical data (Cook and Stevenson, 2010). Such lexicons are crucial to the growing body of work on NLP analyses of historical text (Piotrowski, 2012) which are informing diachronic linguistics (Hamilton et al., 2016), the digital humanities (Muralidharan and Hearst, 2012), and history (Hendrickx et al., 2011).

Our work is inspired by the only previous work on automatically inducing historical sentiment lexicons, Cook and Stevenson (2010); they use the PMI method and a full modern sentiment lexicon as their seed set, which relies on the assumption that all these words have not changed in sentiment. In contrast, in addition to our different algorithm, we use a small seed set of words that were manually selected based on having strong and stable sentiment over the last 150 years (Table 1; confirmed via historical entries in the Oxford English Dictionary).

6.1 Examining the lexicons

We constructed lexicons from COHA, since it was carefully constructed to be genre balanced (e.g., compared to the Google N-Grams; Pechenick et al., 2015). We built lexicons for all adjectives with counts above 100 in a given decade and also for the top-5000 non-stop words within each year. In both these cases we found that $>5\%$ of sentiment-bearing (positive/negative) words completely switched polarity during this 150-year time-period and $>25\%$ of all words changed their sentiment label (including switches to/from neutral).[11] The prevalence of

[11] We defined the thresholds for polar vs. neutral using the class-mass normalization method and compared scores aver-



(a) *Lean* becomes more positive. *Lean* underwent amelioration, becoming more similar to *muscular* and less similar to *weak*.

(b) *Pathetic* becomes more negative. *Pathetic* underwent pejoration, becoming similar to *weak* and less similar to *passionate*.

Figure 6: Examples of amelioration and pejoration.

full polarity switches highlights the importance of historical sentiment lexicons for work on diachronic linguistics and cultural change.

Figure 6a shows an example amelioration detected by this method: the word *lean* lost its negative connotations associated with "weakness" and instead became positively associated with concepts like "muscularity" and "fitness". Figure 6b shows an example pejoration, where *pathetic*, which used to be more synonymous with *passionate*, gained stronger negative associations with the concepts of "weakness" and "inadequacy" (Simpson et al., 1989). In both these cases, semantic similarities computed using our learned historical word vectors were used to contextualize the shifts.

Some other well-known examples of sentiment changes captured by our framework include the semantic bleaching of *sorry*, which shifted from negative and serious ("he was in a sorry state") to uses as a neutral discourse marker ("sorry about that") and *worldly*, which used to have negative connotations related to materialism and religious impurity ("sinful worldly pursuits") but now is frequently used to indicate sophistication ("a cultured, worldly woman") (Simpson et al., 1989). Our hope is that the full lexicons released with this work will spur further examinations of such historical shifts in sentiment, while also facilitating CSS applications that require sentiment ratings for historical text.

7 Conclusion

SentProp allows researchers to easily induce robust and accurate sentiment lexicons that are relevant to their particular domain of study. Such lexicons are crucial to CSS research, as evidenced by our two studies showing that sentiment depends strongly on both social and historical context.

Our methodological comparisons show that simply combining label propagation with high-quality word vector embeddings can achieve state-of-the-art performance competitive with methods that rely on hand-curated dictionaries, and the code package released with this work contains a full benchmark toolkit for this area, including implementations of several variants of SentProp. We hope these tools will facilitate future quantitative studies on the domain-dependency of sentiment.

Of course, the sentiment lexicons induced by SentProp are not perfect, which is reflected in the uncertainty associated with our bootstrap-sampled estimates. However, we believe that these user-constructed, domain-specific lexicons, which quantify uncertainty, provide a more principled foundation for CSS research compared to domain-general sentiment lexicons that contain unknown biases. In the future our method could also be integrated with supervised domain-adaption (e.g.,Yang and Eisenstein, 2015) to further improve these domain-specific results.

Acknowledgements

The authors thank P. Liang for his helpful comments. This research has been supported in part by NSF CNS-1010921, IIS-1149837, IIS-1514268 NIH BD2K, ARO MURI, DARPA XDATA, DARPA SIMPLEX, Stanford Data Science Initiative, SAP Stanford Graduate Fellowship, NSERC PGS-D, Boeing, Lightspeed, and Volkswagen.

aged over 1850-1880 to those averaged over 1970-2000.

References

Muhammad Zubair Asghar, Aurangzeb Khan, Shakeel Ahmad, Imran Ali Khan, and Fazal Masud Kundi. 2015. A Unified Framework for Creating Domain Dependent Polarity Lexicons from User Generated Reviews. *PLOS ONE*, 10(10):e0140204, October.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A. Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*.

Felipe Bravo-Marquez, Eibe Frank, and Bernhard Pfahringer. 2015. From Unlabelled Tweets to Twitter-specific Opinion Words. In *SIGIR*.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior Research Methods*, 44(3):890–907, September.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *EMNLP*.

Paul Cook and Suzanne Stevenson. 2010. Automatically Identifying Changes in the Semantic Orientation of Words. In *LREC*.

Mark Davies. 2010. The Corpus of Historical American English: 400 million words, 1810-2009. http://corpus.byu.edu/coha/.

Lingjia Deng, Janyce Wiebe, and Yoonjung Choi. 2014. Joint inference and disambiguation of implicit sentiments via implicature constraints. In *COLING*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*.

Ethan Fast, Binbin Chen, and Michael S. Bernstein. 2016. Empath: Understanding Topic Signals in Large-Scale Text. In *CHI*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *ACL*.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *EACL*.

Iris Hendrickx, Michel Gnreux, and Rita Marquilhas. 2011. Automatic pragmatic text segmentation of historical letters. In *Language Technology for Cultural Heritage*, pages 135–153. Springer.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *ACL*.

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

Sheng Huang, Zhendong Niu, and Chongyang Shi. 2014. Automatic construction of domain-specific sentiment lexicon based on constrained label propagation. *Knowledge-Based Systems*, 56:191–200.

Sean P. Igo and Ellen Riloff. 2009. Corpus-based semantic lexicon induction with web-based corroboration. In *ACL Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*.

Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. 2010. Generating focused topic-specific sentiment lexicons. In *ACL*.

Jaap Kamps, M. J. Marx, Robert J. Mokken, and M. de Rijke. 2004. Using wordnet to measure semantic orientations of adjectives. In *LREC*.

William Labov. 2006. *The social stratification of English in New York City*. Cambridge University Press.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychol. Rev.*, 104(2):211.

Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. 2014. On the Importance of Text Analysis for Stock Price Prediction. In *LREC*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Trans. Assoc. Comput. Ling.*, 3.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, 66(1):35–65.

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

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *NAACL*.

Aditi Muralidharan and Marti A Hearst. 2012. Supporting exploratory text analysis in literature study. *Literary and Linguistic Computing*.

Eitan Adam Pechenick, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Characterizing the Google Books corpus: Strong limits to inferences of socio-cultural and linguistic evolution. *PLoS ONE*, 10(10).

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

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis Lectures on Human Language Technologies*, 5(2):1–157.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding Domain Sentiment Lexicon through Double Propagation. In *IJCAI*.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *EACL*.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. *arXiv preprint cmp-lg/9706013*.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *ACL*.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in Twitter. *SemEval-2015*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schutze. 2016. Ultradense Word Embeddings by Orthogonal Transformation. In *NAACL-HLT*.

Inaki San Vicente, Rodrigo Agerri, German Rigau, and Donostia-San Sebastin. 2014. Simple, Robust and (almost) Unsupervised Generation of Polarity Lexicons for Multiple Languages. In *EACL*.

Aliaksei Severyn and Alessandro Moschitti. 2015. On the automatic learning of sentiment lexicons. In *NAACL-HLT*.

John Andrew Simpson, Edmund SC Weiner, et al. 1989. *The Oxford English Dictionary*, volume 2. Clarendon Press Oxford, Oxford, UK.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *ACL Workshop on Unsupervised Learning in NLP*.

Philip J Stone, Dexter C Dunphy, and Marshall S Smith. 1966. The General Inquirer: A Computer Approach to Content Analysis.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Ling.*, 37(2):267–307.

Yen-Jen Tai and Hung-Yu Kao. 2013. Automatic domain-specific sentiment lexicon generation with label propagation. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 53. ACM.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL*.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014. Building Large-Scale Twitter-Specific Sentiment Lexicon: A Representation Learning Approach. In *COLING*.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP*.

Elizabeth Closs Traugott and Richard B Dasher. 2001. *Regularity in Semantic Change*. Cambridge University Press, Cambridge, UK.

Peter Trudgill. 1974. Linguistic change and diffusion: Description and explanation in sociolinguistic dialect geography. *Language in Society*, 3(2):215–246.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Sys.*, 21(4):315–346.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.*, 37(1):141–188.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *NAACL-HLT*.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior research methods*, 45(4):1191–1207.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *COLING*.

Yi Yang and Jacob Eisenstein. 2015. Putting Things in Context: Community-specific Embedding Projections for Sentiment Analysis. *arXiv preprint arXiv:1511.06052*.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Scholkopf. 2004. Learning with local and global consistency. In *NIPS*.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report.

Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, and others. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*.

Attention-based LSTM for Aspect-level Sentiment Classification

Yequan Wang and **Minlie Huang** and **Li Zhao*** and **Xiaoyan Zhu**
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
*Microsoft Research Asia
wangyequan@live.cn, aihuang@tsinghua.edu.cn
lizo@microsoft.com, zxy-dcs@tsinghua.edu.cn

Abstract

Aspect-level sentiment classification is a fine-grained task in sentiment analysis. Since it provides more complete and in-depth results, aspect-level sentiment analysis has received much attention these years. In this paper, we reveal that the sentiment polarity of a sentence is not only determined by the content but is also highly related to the concerned aspect. For instance, "*The appetizers are ok, but the service is slow.*", for aspect *taste*, the polarity is positive while for *service*, the polarity is negative. Therefore, it is worthwhile to explore the connection between an aspect and the content of a sentence. To this end, we propose an Attention-based Long Short-Term Memory Network for aspect-level sentiment classification. The attention mechanism can concentrate on different parts of a sentence when different aspects are taken as input. We experiment on the SemEval 2014 dataset and results show that our model achieves state-of-the-art performance on aspect-level sentiment classification.

1 Introduction

Sentiment analysis (Nasukawa and Yi, 2003), also known as opinion mining (Liu, 2012), is a key NLP task that receives much attention these years. Aspect-level sentiment analysis is a fine-grained task that can provide complete and in-depth results. In this paper, we deal with aspect-level sentiment classification and we find that the sentiment polarity of a sentence is highly dependent on both content and aspect. For example, the sentiment polarity of "*Staffs are not that friendly, but the taste covers all.*" will be positive if the aspect is *food* but negative when considering the aspect *service*. Polarity could be opposite when different aspects are considered.

Neural networks have achieved state-of-the-art performance in a variety of NLP tasks such as machine translation (Lample et al., 2016), paraphrase identification (Yin et al., 2015), question answering (Golub and He, 2016) and text summarization (Rush et al., 2015). However, neural network models are still in infancy to deal with aspect-level sentiment classification. In some works, target dependent sentiment classification can be benefited from taking into account target information, such as in Target-Dependent LSTM (TD-LSTM) and Target-Connection LSTM (TC-LSTM) (Tang et al., 2015a). However, those models can only take into consideration the target but not aspect information which is proved to be crucial for aspect-level classification.

Attention has become an effective mechanism to obtain superior results, as demonstrated in image recognition (Mnih et al., 2014), machine translation (Bahdanau et al., 2014), reasoning about entailment (Rocktäschel et al., 2015) and sentence summarization (Rush et al., 2015). Even more, neural attention can improve the ability to read comprehension (Hermann et al., 2015). In this paper, we propose an attention mechanism to enforce the model to attend to the important part of a sentence, in response to a specific aspect. We design an aspect-to-sentence attention mechanism that can concentrate

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 606–615,
Austin, Texas, November 1-5, 2016.

on the key part of a sentence given the aspect.

We explore the potential correlation of aspect and sentiment polarity in aspect-level sentiment classification. In order to capture important information in response to a given aspect, we design an attention-based LSTM. We evaluate our approach on a benchmark dataset (Pontiki et al., 2014), which contains restaurants and laptops data.

The main contributions of our work can be summarized as follows:

- We propose attention-based Long Short-Term memory for aspect-level sentiment classification. The models are able to attend different parts of a sentence when different aspects are concerned. Results show that the attention mechanism is effective.
- Since aspect plays a key role in this task, we propose two ways to take into account aspect information during attention: one way is to concatenate the aspect vector into the sentence hidden representations for computing attention weights, and another way is to additionally append the aspect vector into the input word vectors.
- Experimental results indicate that our approach can improve the performance compared with several baselines, and further examples demonstrate the attention mechanism works well for aspect-level sentiment classification.

The rest of our paper is structured as follows: Section 2 discusses related works, Section 3 gives a detailed description of our attention-based proposals, Section 4 presents extensive experiments to justify the effectiveness of our proposals, and Section 5 summarizes this work and the future direction.

2 Related Work

In this section, we will review related works on aspect-level sentiment classification and neural networks for sentiment classification briefly.

2.1 Sentiment Classification at Aspect-level

Aspect-level sentiment classification is typically considered as a classification problem in the literature. As we mentioned before, aspect-level sentiment classification is a fine-grained classification task. The majority of current approaches attempt to detecting the polarity of the entire sentence, regardless of the entities mentioned or aspects. Traditional approaches to solve those problems are to manually design a set of features. With the abundance of sentiment lexicons (Rao and Ravichandran, 2009; Perez-Rosas et al., 2012; Kaji and Kitsuregawa, 2007), the lexicon-based features were built for sentiment analysis (Mohammad et al., 2013). Most of these studies focus on building sentiment classifiers with features, which include bag-of-words and sentiment lexicons, using SVM (Mullen and Collier, 2004). However, the results highly depend on the quality of features. In addition, feature engineering is labor intensive.

2.2 Sentiment Classification with Neural Networks

Since a simple and effective approach to learn distributed representations was proposed (Mikolov et al., 2013), neural networks advance sentiment analysis substantially. Classical models including Recursive Neural Network (Socher et al., 2011; Dong et al., 2014; Qian et al., 2015), Recursive Neural Tensor Network (Socher et al., 2013), Recurrent Neural Network (Mikolov et al., 2010; Tang et al., 2015b), LSTM (Hochreiter and Schmidhuber, 1997) and Tree-LSTMs (Tai et al., 2015) were applied into sentiment analysis currently. By utilizing syntax structures of sentences, tree-based LSTMs have been proved to be quite effective for many NLP tasks. However, such methods may suffer from syntax parsing errors which are common in resource-lacking languages.

LSTM has achieved a great success in various NLP tasks. TD-LSTM and TC-LSTM (Tang et al., 2015a), which took target information into consideration, achieved state-of-the-art performance in target-dependent sentiment classification. TC-LSTM obtained a target vector by averaging the vectors of words that the target phrase contains. However, simply averaging the word embeddings of a target phrase is not sufficient to represent the semantics of the target phrase, resulting a suboptimal performance.

Despite the effectiveness of those methods, it is still challenging to discriminate different sentiment polarities at a fine-grained aspect level. Therefore, we are motivated to design a powerful neural network which can fully employ aspect information for sentiment classification.

3 Attention-based LSTM with Aspect Embedding

3.1 Long Short-term Memory (LSTM)

Recurrent Neural Network(RNN) is an extension of conventional feed-forward neural network. However, standard RNN has the gradient vanishing or exploding problems. In order to overcome the issues, Long Short-term Memory network (LSTM) was developed and achieved superior performance (Hochreiter and Schmidhuber, 1997). In the LSTM architecture, there are three gates and a cell memory state. Figure 1 illustrates the architecture of a standard LSTM.

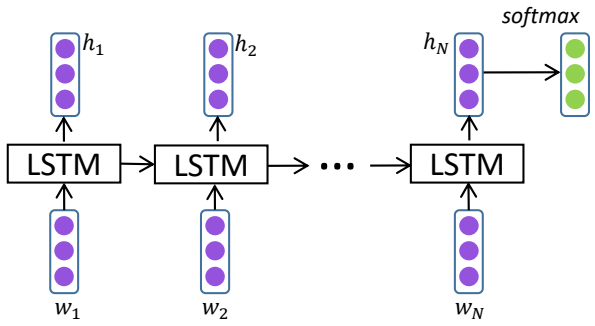


Figure 1: The architecture of a standard LSTM. $\{w_1, w_2, \ldots, w_N\}$ represent the word vector in a sentence whose length is N. $\{h_1, h_2, \ldots, h_N\}$ is the hidden vector.

More formally, each cell in LSTM can be computed as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \quad (1)$$

$$f_t = \sigma(W_f \cdot X + b_f) \quad (2)$$

$$i_t = \sigma(W_i \cdot X + b_i) \quad (3)$$

$$o_t = \sigma(W_o \cdot X + b_o) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot X + b_c) \quad (5)$$

$$h_t = o_t \odot tanh(c_t) \quad (6)$$

where $W_i, W_f, W_o \in \mathbb{R}^{d \times 2d}$ are the weighted matrices and $b_i, b_f, b_o \in \mathbb{R}^d$ are biases of LSTM to be learned during training, parameterizing the transformations of the input, forget and output gates respectively. σ is the sigmoid function and $\odot$ stands for element-wise multiplication. x_t includes the inputs of LSTM cell unit, representing the word embedding vectors w_t in Figure 1. The vector of hidden layer is h_t.

We regard the last hidden vector h_N as the representation of sentence and put h_N into a $softmax$ layer after linearizing it into a vector whose length is equal to the number of class labels. In our work, the set of class labels is {*positive*, *negative*, *neutral*}.

3.2 LSTM with Aspect Embedding (AE-LSTM)

Aspect information is vital when classifying the polarity of one sentence given aspect. We may get opposite polarities if different aspects are considered. To make the best use of aspect information, we propose to learn an embedding vector for each aspect.

Vector $v_{a_i} \in \mathbb{R}^{d_a}$ is represented for the embedding of aspect i, where d_a is the dimension of aspect embedding. $A \in \mathbb{R}^{d_a \times |A|}$ is made up of all aspect embeddings. To the best of our knowledge, it is the first time to propose aspect embedding.

3.3 Attention-based LSTM (AT-LSTM)

The standard LSTM cannot detect which is the important part for aspect-level sentiment classification. In order to address this issue, we propose to design an attention mechanism that can capture the key part of sentence in response to a given aspect. Figure 2 represents the architecture of an Attention-based LSTM (AT-LSTM).

Let $H \in \mathbb{R}^{d \times N}$ be a matrix consisting of hidden vectors $[h_1, \ldots, h_N]$ that the LSTM produced, where d is the size of hidden layers and N is the length of the given sentence. Furthermore, v_a represents the embedding of aspect and $e_N \in \mathbb{R}^N$ is a vector of 1s. The attention mechanism will produce an attention weight vector α and a weighted hidden

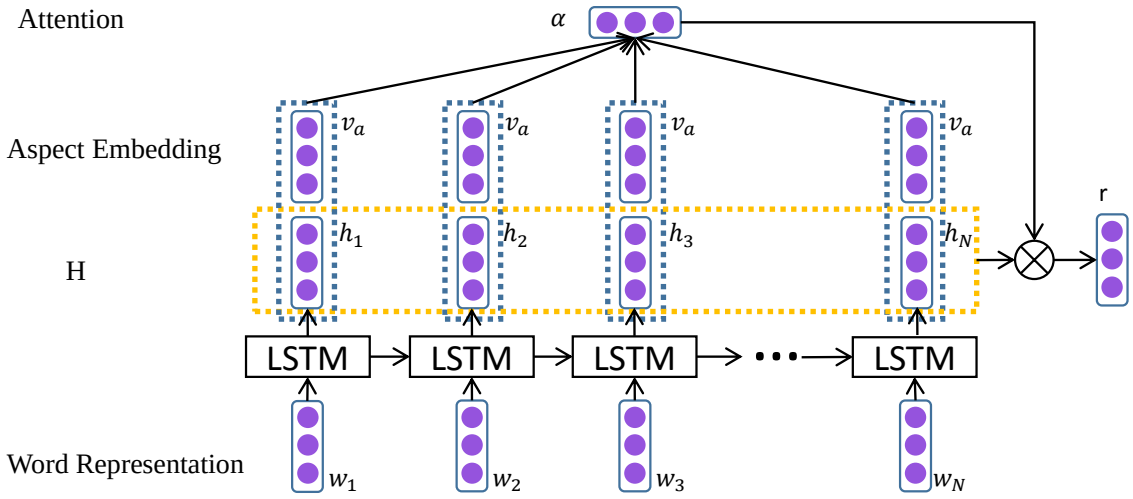


Figure 2: The Architecture of Attention-based LSTM. The aspect embeddings have been used to decide the attention weights along with the sentence representations. $\{w_1, w_2, \ldots, w_N\}$ represent the word vector in a sentence whose length is N. v_a represents the aspect embedding. α is the attention weight. $\{h_1, h_2, \ldots, h_N\}$ is the hidden vector.

representation r.

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix}) \quad (7)$$

$$\alpha = softmax(w^T M) \quad (8)$$

$$r = H\alpha^T \quad (9)$$

where, $M \in \mathbb{R}^{(d+d_a)\times N}$, $\alpha \in \mathbb{R}^N$, $r \in \mathbb{R}^d$. $W_h \in \mathbb{R}^{d\times d}$, $W_v \in \mathbb{R}^{d_a \times d_a}$ and $w \in \mathbb{R}^{d+d_a}$ are projection parameters. α is a vector consisting of attention weights and r is a weighted representation of sentence with given aspect. The operator in 7 (a circle with a multiplication sign inside, OP for short here) means: $v_a \otimes e_N = [v; v; \ldots; v]$, that is, the operator repeatedly concatenates v for N times, where e_N is a column vector with N 1s. $W_v v_a \otimes e_N$ is repeating the linearly transformed v_a as many times as there are words in sentence.

The final sentence representation is given by:

$$h^* = tanh(W_p r + W_x h_N) \quad (10)$$

where, $h^* \in \mathbb{R}^d$, W_p and W_x are projection parameters to be learned during training. We find that this works practically better if we add $W_x h_N$ into the final representation of the sentence, which is inspired by (Rocktäschel et al., 2015).

The attention mechanism allows the model to capture the most important part of a sentence when different aspects are considered.

h^* is considered as the feature representation of a sentence given an input aspect. We add a linear layer to convert sentence vector to e, which is a real-valued vector with the length equal to class number $|C|$. Then, a $softmax$ layer is followed to transform e to conditional probability distribution.

$$y = softmax(W_s h^* + b_s) \quad (11)$$

where W_s and b_s are the parameters for $softmax$ layer.

3.4 Attention-based LSTM with Aspect Embedding (ATAE-LSTM)

The way of using aspect information in AE-LSTM is letting aspect embedding play a role in computing the attention weight. In order to better take advantage of aspect information, we append the input aspect embedding into each word input vector. The structure of this model is illustrated in 3. In this way, the output hidden representations $(h_1, h_2, ..., h_N)$ can have the information from the input aspect (v_a). Therefore, in the following step that compute the attention weights, the inter-

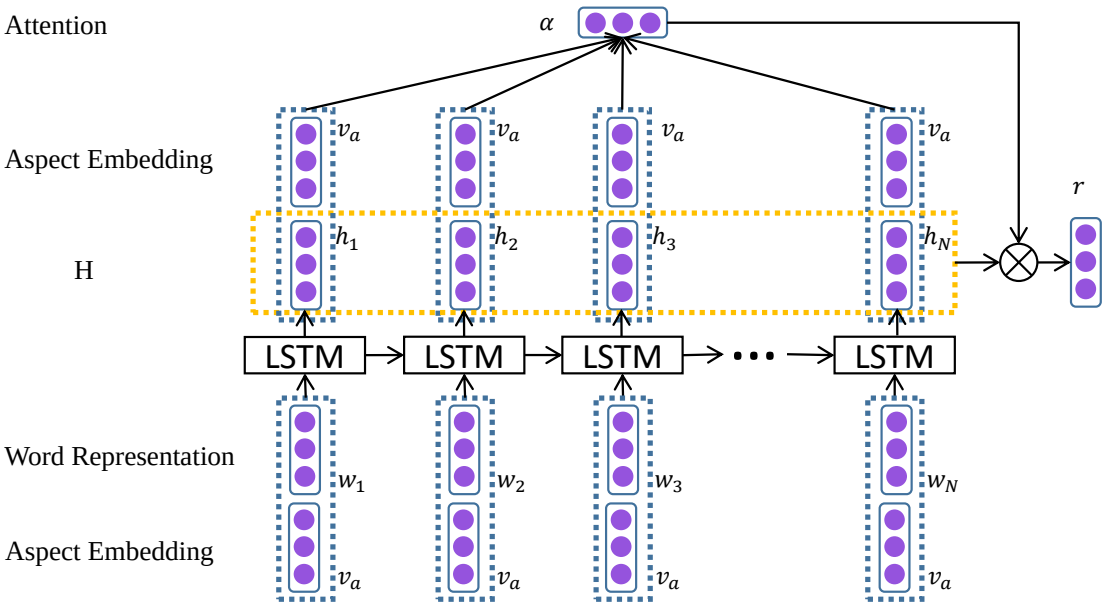


Figure 3: The Architecture of Attention-based LSTM with Aspect Embedding. The aspect embeddings have been take as input along with the word embeddings. $\{w_1, w_2, \ldots, w_N\}$ represent the word vector in a sentence whose length is N. v_a represents the aspect embedding. α is the attention weight. $\{h_1, h_2, \ldots, h_N\}$ is the hidden vector.

dependence between words and the input aspect can be modeled.

3.5 Model Training

The model can be trained in an end-to-end way by backpropagation, where the objective function (loss function) is the cross-entropy loss. Let y be the target distribution for sentence, $\hat{y}$ be the predicted sentiment distribution. The goal of training is to minimize the cross-entropy error between y and $\hat{y}$ for all sentences.

$$loss = -\sum_i \sum_j y_i^j log\hat{y}_i^j + \lambda||\theta||^2 \qquad (12)$$

where i is the index of sentence, j is the index of class. Our classification is three way. λ is the L_2 - regularization term. θ is the parameter set.

Similar to standard LSTM, the parameter set is $\{W_i, b_i, W_f, b_f, W_o, b_o, W_c, b_c, W_s, b_s\}$. Furthermore, word embeddings are the parameters too. Note that the dimension of W_i, W_f, W_o, W_c changes along with different models. If the aspect embeddings are added into the input of the LSTM cell unit, the dimension of W_i, W_f, W_o, W_c will be enlarged correspondingly. Additional parameters are listed as follows:

AT-LSTM: The aspect embedding A is added into the set of parameters naturally. In addition, W_h, W_v, W_p, W_x, w are the parameters of attention. Therefore, the additional parameter set of AT-LSTM is $\{A, W_h, W_v, W_p, W_x, w\}$.

AE-LSTM: The parameters include the aspect embedding A. Besides, the dimension of W_i, W_f, W_o, W_c will be expanded since the aspect vector is concatenated. Therefore, the additional parameter set consists of $\{A\}$.

ATAE-LSTM: The parameter set consists of $\{A, W_h, W_v, W_p, W_x, w\}$. Additionally, the dimension of W_i, W_f, W_o, W_c will be expanded with the concatenation of aspect embedding.

The word embedding and aspect embedding are optimized during training. The percentage of out-of-vocabulary words is about 5%, and they are randomly initialized from $U(-\epsilon, \epsilon)$, where $\epsilon = 0.01$.

In our experiments, we use AdaGrad (Duchi et al., 2011) as our optimization method, which has

improved the robustness of SGD on large scale learning task remarkably in a distributed environment (Dean et al., 2012). AdaGrad adapts the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent parameters.

4 Experiment

We apply the proposed model to aspect-level sentiment classification. In our experiments, all word vectors are initialized by Glove[1] (Pennington et al., 2014). The word embedding vectors are pre-trained on an unlabeled corpus whose size is about 840 billion. The other parameters are initialized by sampling from a uniform distribution $U(-\epsilon, \epsilon)$. The dimension of word vectors, aspect embeddings and the size of hidden layer are 300. The length of attention weights is the same as the length of sentence. Theano (Bastien et al., 2012) is used for implementing our neural network models. We trained all models with a batch size of 25 examples, and a momentum of 0.9, L_2-regularization weight of 0.001 and initial learning rate of 0.01 for AdaGrad.

4.1 Dataset

We experiment on the dataset of SemEval 2014 Task 4[2] (Pontiki et al., 2014). The dataset consists of customers reviews. Each review contains a list of aspects and corresponding polarities. Our aim is to identify the aspect polarity of a sentence with the corresponding aspect. The statistics is presented in Table 1.

4.2 Task Definition

Aspect-level Classification Given a set of pre-identified aspects, this task is to determine the polarity of each aspect. For example, given a sentence, "*The restaurant was too expensive.*", there is an aspect *price* whose polarity is negative. The set of aspects is {*food*, *price*, *service*, *ambience*, *anecdotes/miscellaneous*}. In the dataset of SemEval 2014 Task 4, there is only restaurants data that has aspect-specific polarities. Table 2 illustrates the comparative results.

[1]Pre-trained word vectors of Glove can be obtained from `http://nlp.stanford.edu/projects/glove/`

[2]The introduction about SemEval 2014 can be obtained from `http://alt.qcri.org/semeval2014/`

Asp.	Positive		Negative		Neural	
	Train	Test	Train	Test	Train	Test
Fo.	867	302	209	69	90	31
Pr.	179	51	115	28	10	1
Se.	324	101	218	63	20	3
Am.	263	76	98	21	23	8
An.	546	127	199	41	357	51
Total	2179	657	839	222	500	94

Table 1: Aspects distribution per sentiment class. {*Fo., Pr., Se, Am., An.*} refer to {*food, price, service, ambience, anecdotes/miscellaneous*}. "*Asp.*" refers to aspect.

Models	Three-way	Pos./Neg.
LSTM	82.0	88.3
TD-LSTM	82.6	89.1
TC-LSTM	81.9	89.2
AE-LSTM	82.5	88.9
AT-LSTM	83.1	89.6
ATAE-LSTM	**84.0**	**89.9**

Table 2: Accuracy on aspect level polarity classification about restaurants. *Three-way* stands for 3-class prediction. *Pos./Neg.* indicates binary prediction where ignoring all neutral instances. Best scores are in bold.

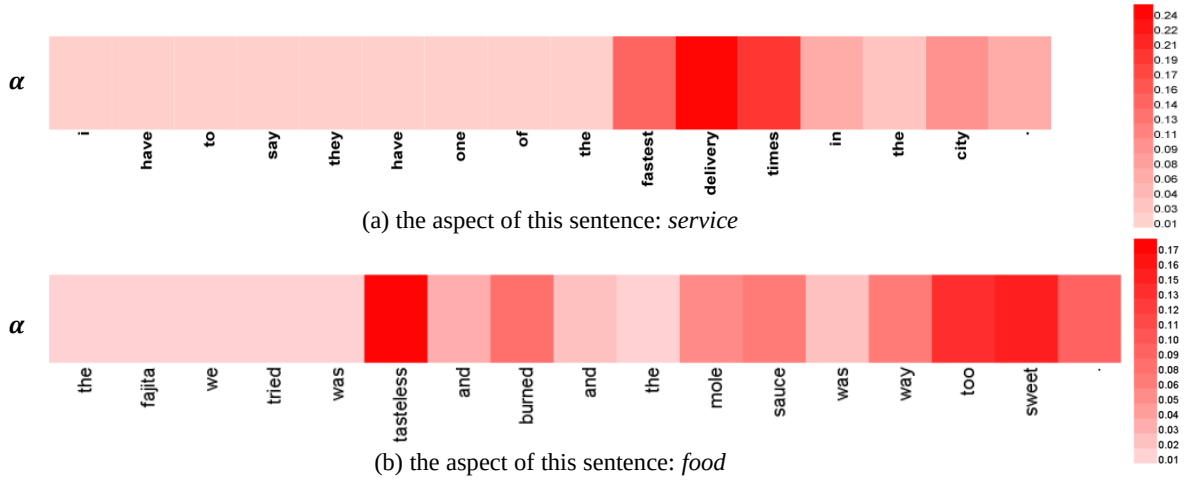
Aspect-Term-level Classification For a given set of aspects term within a sentence, this task is to determine whether the polarity of each aspect term is positive, negative or neutral. We conduct experiments on the dataset of SemEval 2014 Task 4. In the sentences of both restaurant and laptop datasets, there are the location and sentiment polarity for each occurrence of an aspect term. For example, there is an aspect term *fajitas* whose polarity is negative in sentence "*I loved their fajitas.*".

Experiments results are shown in Table 3 and Table 4. Similar to the experiment on aspect-level classification, our models achieve state-of-the-art performance.

4.3 Comparison with baseline methods

We compare our model with several baselines, including LSTM, TD-LSTM, and TC-LSTM.

LSTM: Standard LSTM cannot capture any aspect information in sentence, so it must get the same



(a) the aspect of this sentence: *service*

(b) the aspect of this sentence: *food*

Figure 4: Attention Visualizations. The aspects of (a) and (b) are *service* and *food* respectively. The color depth expresses the importance degree of the weight in attention vector α. From (a), attention can detect the important words from the whole sentence dynamically even though multi-semantic phrase such as "*fastest delivery times*" which can be used in other areas. From (b), attention can know multi-keypoints if more than one keypoint existing.

Models	Three-way	Pos./Neg.
LSTM	74.3	-
TD-LSTM	75.6	-
AE-LSTM	76.6	89.6
ATAE-LSTM	**77.2**	**90.9**

Table 3: Accuracy on aspect term polarity classification about restaurants. *Three-way* stands for 3-class prediction. *Pos./Neg.* indicates binary prediction where ignoring all neutral instances. Best scores are in bold.

Models	Three-way	Pos./Neg.
LSTM	66.5	-
TD-LSTM	68.1	-
AE-LSTM	**68.9**	87.4
ATAE-LSTM	68.7	**87.6**

Table 4: Accuracy on aspect term polarity classification about laptops. *Three-way* stands for 3-class prediction. *Pos./Neg.* indicates binary prediction where ignoring all neutral instances. Best scores are in bold.

sentiment polarity although given different aspects. Since it cannot take advantage of the aspect information, not surprisingly the model has worst performance.

TD-LSTM: TD-LSTM can improve the performance of sentiment classifier by treating an aspect as a target. Since there is no attention mechanism in TD-LSTM, it cannot "know" which words are important for a given aspect.

TC-LSTM: TC-LSTM extended TD-LSTM by incorporating a target into the representation of a sentence. It is worth noting that TC-LSTM performs worse than LSTM and TD-LSTM in Table 2. TC-LSTM added target representations, which was obtained from word vectors, into the input of the LSTM cell unit.

In our models, we embed aspects into another vector space. The embedding vector of aspects can be learned well in the process of training. ATAE-LSTM not only addresses the shortcoming of the unconformity between word vectors and aspect embeddings, but also can capture the most important information in response to a given aspect. In addition, ATAE-LSTM can capture the important and different parts of a sentence when given different aspects.

4.4 Qualitative Analysis

It is enlightening to analyze which words decide the sentiment polarity of the sentence given an aspect. We can obtain the attention weight α in Equation 8 and visualize the attention weights accordingly.

Figure 4 shows the representation of how attention focuses on words with the influence of a given aspect. We use a visualization tool Heml (Deng

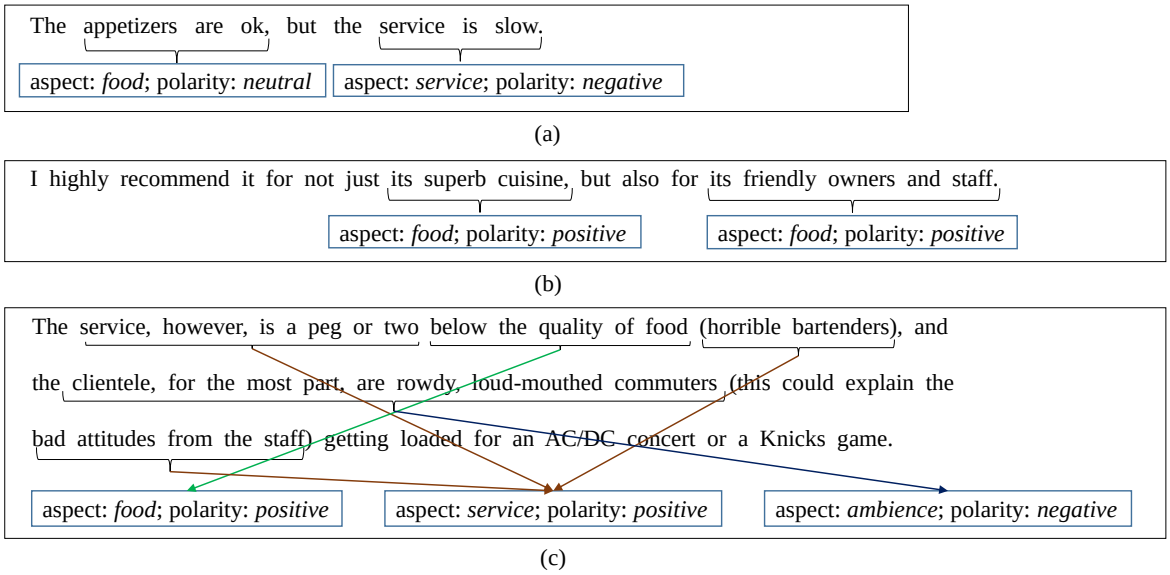


Figure 5: Examples of classification. (a) is an instance with different aspects. (b) represents that our model can focus on where the keypoints are and not disturbed by the privative word *not*. (c) stands for long and complicated sentences. Our model can obtain correct sentiment polarity.

et al., 2014) to visualize the sentences. The color depth indicates the importance degree of the weight in attention vector α, the darker the more important. The sentences in Figure 4 are "*I have to say they have one of the fastest delivery times in the city* ." and "*The fajita we tried was tasteless and burned and the mole sauce was way too sweet.*". The corresponding aspects are *service* and *food* respectively. Obviously attention can get the important parts from the whole sentence dynamically. In Figure 4 (a), "*fastest delivery times*" is a multi-word phrase, but our attention-based model can detect such phrases if *service* can is the input aspect. Besides, the attention can detect multiple keywords if more than one keyword is existing. In Figure 4 (b), *tastless* and *too sweet* are both detected.

4.5 Case Study

As we demonstrated, our models obtain the state-of-the-art performance. In this section, we will further show the advantages of our proposals through some typical examples.

In Figure 5, we list some examples from the test set which have typical characteristics and cannot be inferred by LSTM. In sentence (a), "*The appetizers are ok, but the service is slow.*", there are two aspects *food* and *service*. Our model can discriminate different sentiment polarities with different aspects. In sentence (b), "*I highly recommend it for not just its superb cuisine, but also for its friendly owners and staff.*", there is a negation word *not*. Our model can obtain correct polarity, not affected by the negation word who doesn't represent negation here. In the last instance (c), "*The service, however, is a peg or two below the quality of food (horrible bartenders), and the clientele, for the most part, are rowdy, loud-mouthed commuters (this could explain the bad attitudes from the staff) getting loaded for an AC/DC concert or a Knicks game.*", the sentence has a long and complicated structure so that existing parser may hardly obtain correct parsing trees. Hence, tree-based neural network models are difficult to predict polarity correctly. While our attention-based LSTM can work well in those sentences with the help of attention mechanism and aspect embedding.

5 Conclusion and Future Work

In this paper, we have proposed attention-based LSTMs for aspect-level sentiment classification. The key idea of these proposals are to learn aspect

embeddings and let aspects participate in computing attention weights. Our proposed models can concentrate on different parts of a sentence when different aspects are given so that they are more competitive for aspect-level classification. Experiments show that our proposed models, AE-LSTM and ATAE-LSTM, obtain superior performance over the baseline models.

Though the proposals have shown potentials for aspect-level sentiment analysis, different aspects are input separately. As future work, an interesting and possible direction would be to model more than one aspect simultaneously with the attention mechanism.

Acknowledgments

This work was partly supported by the National Basic Research Program (973 Program) under grant No.2012CB316301/2013CB329403, the National Science Foundation of China under grant No.61272227/61332007, and the Beijing Higher Education Young Elite Teacher Project. The work was also supported by Tsinghua University Beijing Samsung Telecom R&D Center Joint Laboratory for Intelligent Media Computing.

References

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

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In Advances in Neural Information Processing Systems, pages 1223–1231.

Wankun Deng, Yongbo Wang, Zexian Liu, Han Cheng, and Yu Xue. 2014. Hemi: a toolkit for illustrating heatmaps. PloS one, 9(11):e111988.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In ACL (2), pages 49–54.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. arXiv preprint arXiv:1604.00727.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems, pages 1684–1692.

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

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In EMNLP-CoNLL, pages 1075–1083.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

Bing Liu. 2012. Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1):1–167.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In INTERSPEECH, volume 2, page 3.

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

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In Advances in Neural Information Processing Systems, pages 2204–2212.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. arXiv preprint arXiv:1308.6242.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In EMNLP, volume 4, pages 412–418.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In Proceedings of the 2nd international conference on Knowledge capture, pages 70–77. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12:1532–1543.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In LREC, volume 12, page 73.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), pages 27–35.

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu, Xuan Zhu, and Xiaoyan Zhu. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, volume 1, pages 1365–1374.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pages 675–682. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. arXiv preprint arXiv:1509.06664.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In EMNLP, volume 1631, page 1642. Citeseer.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015a. Target-dependent sentiment classification with long short term memory. arXiv preprint arXiv:1512.01100.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422–1432.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:1512.05193.

Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis

Wenya Wang[†‡] **Sinno Jialin Pan**[†] **Daniel Dahlmeier**[‡] **Xiaokui Xiao**[†]
[†]Nanyang Technological University, Singapore
[‡]SAP Innovation Center Singapore
[†]{wa0001ya, sinnopan, xkxiao}@ntu.edu.sg, [‡]{d.dahlmeier}@sap.com

Abstract

In aspect-based sentiment analysis, extracting aspect terms along with the opinions being expressed from user-generated content is one of the most important subtasks. Previous studies have shown that exploiting connections between aspect and opinion terms is promising for this task. In this paper, we propose a novel joint model that integrates recursive neural networks and conditional random fields into a unified framework for explicit aspect and opinion terms co-extraction. The proposed model learns high-level discriminative features and double propagates information between aspect and opinion terms, simultaneously. Moreover, it is flexible to incorporate hand-crafted features into the proposed model to further boost its information extraction performance. Experimental results on the dataset from SemEval Challenge 2014 task 4 show the superiority of our proposed model over several baseline methods as well as the winning systems of the challenge.

1 Introduction

Aspect-based sentiment analysis (Pang and Lee, 2008; Liu, 2011) aims to extract important information, e.g., opinion targets, opinion expressions, target categories, and opinion polarities, from user-generated content, such as microblogs, reviews, etc. This task was first studied by Hu and Liu (2004a; 2004b), followed by Popescu and Etzioni (2005), Zhuang et al. (2006), Zhang et al. (2010), Qiu et al. (2011), Li et al. (2010). In aspect-based sentiment analysis, one of the goals is to extract explicit aspects of an entity from text, along with the opinions being expressed. For example, in a restaurant review *"I have to say they have one of the fastest delivery times in the city."*, the aspect term is *delivery times*, and the opinion term is *fastest*.

Among previous work, one of the approaches is to accumulate aspect and opinion terms from a seed collection without label information, by utilizing syntactic rules or modification relations between them (Qiu et al., 2011; Liu et al., 2013b). In the above example, if we know *fastest* is an opinion word, then *delivery times* is probably inferred to be an aspect because *fastest* is its modifier. However, this approach largely relies on hand-coded rules and is restricted to certain Part-of-Speech (POS) tags, e.g., opinion words are restricted to be adjectives. Another approach focuses on feature engineering based on predefined lexicons, syntactic analysis, etc. (Jin and Ho, 2009; Li et al., 2010). A sequence labeling classifier is then built to extract aspect and opinion terms. This approach requires extensive efforts for designing hand-crafted features and only combines features linearly for classification which ignores higher order interactions.

To overcome the limitations of existing methods, we propose a novel model, named Recursive Neural Conditional Random Fields (RNCRF). Specifically, RNCRF consists of two main components. The first component is a recursive neural network (RNN)[1] (Socher et al., 2010) based on a dependency tree of each sentence. The goal is to learn a high-level feature representation for each word in the context of each sentence and make the representation learning for aspect and opinion terms interactive through the underlying dependency structure among them. The output of the RNN is then fed into a Conditional Random Field (CRF) (Lafferty et al., 2001) to learn a discriminative mapping from high-

[1]Note that in this paper, RNN stands for recursive neural network instead of recurrent neural network.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 616–626,
Austin, Texas, November 1-5, 2016.

level features to labels, i.e., *aspects*, *opinions*, or *others*, so that context information can be well captured. Our main contributions are to use RNN for encoding aspect-opinion relations in high-level representation learning and to present a joint optimization approach based on maximum likelihood and backpropagation to learn the RNN and CRF components simultaneously. In this way, the label information of aspect and opinion terms can be dually propagated from parameter learning in CRF to representation learning in RNN. We conduct expensive experiments on the dataset from SemEval challenge 2014 task 4 (subtask 1) (Pontiki et al., 2014) to verify the superiority of RNCRF over several baseline methods as well as the winning systems of the challenge.

2 Related Work

2.1 Aspects and Opinions Co-Extraction

Hu et al. (2004a) proposed to extract product aspects through association mining, and opinion terms by augmenting a seed opinion set using synonyms and antonyms in WordNet. In follow-up work, syntactic relations are further exploited for aspect/opinion extraction (Popescu and Etzioni, 2005; Wu et al., 2009; Qiu et al., 2011). For example, Qiu et al. (2011) used syntactic relations to double propagate and augment the sets of aspects and opinions. Although the above models are unsupervised, they heavily depend on predefined rules for extraction and are restricted to specific types of POS tags for product aspects and opinions. Jin et al. (2009), Li et al. (2010), Jakob et al. (2010) and Ma et al. (2010) modeled the extraction problem as a sequence tagging problem and proposed to use HMMs or CRFs to solve it. These methods rely on rich hand-crafted features and do not consider interactions between aspect and opinion terms explicitly. Another direction is to use word alignment model to capture opinion relations among a sentence (Liu et al., 2012; Liu et al., 2013a). This method requires sufficient data for modeling desired relations.

Besides explicit aspects and opinions extraction, there are also other lines of research related to aspect-based sentiment analysis, including aspect classification (Lakkaraju et al., 2014; McAuley et al., 2012), aspect rating (Titov and McDonald, 2008; Wang et al., 2011; Wang and Ester, 2014), domain-specific and target-dependent sentiment classification (Lu et al., 2011; Ofek et al., 2016; Dong et al., 2014; Tang et al., 2015).

2.2 Deep Learning for Sentiment Analysis

Recent studies have shown that deep learning models can automatically learn the inherent semantic and syntactic information from data and thus achieve better performance for sentiment analysis (Socher et al., 2011b; Socher et al., 2012; Socher et al., 2013; Glorot et al., 2011; Kalchbrenner et al., 2014; Kim, 2014; Le and Mikolov, 2014). These methods generally belong to sentence-level or phrase/word-level sentiment polarity predictions. Regarding aspect-based sentiment analysis, Irsoy et al. (2014) applied deep recurrent neural networks for opinion expression extraction. Dong et al. (2014) proposed an adaptive recurrent neural network for target-dependent sentiment classification, where targets or aspects are given as input. Tang et al. (2015) used Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for the same task. Nevertheless, there is little work in aspects and opinions co-extraction using deep learning models.

To the best of our knowledge, the work of Liu et al. (2015) and Yin et al. (2016) are the most related to ours. Liu et al. (2015) proposed to combine recurrent neural network and word embeddings to extract explicit aspects. However, the proposed model simply uses recurrent neural network on top of word embeddings, and thus its performance heavily depends on the quality of word embeddings. In addition, it fails to explicitly model dependency relations or compositionalities within certain syntactic structure in a sentence. Recently, Yin et al. (2016) proposed an unsupervised learning method to improve word embeddings using dependency path embeddings. A CRF is then trained with the embeddings independently in the pipeline.

Different from (Yin et al., 2016), our model does not focus on developing a new unsupervised word embedding methods, but encodes the information of dependency paths into RNN for constructing syntactically meaningful and discriminative hidden representations with labels. Moreover, we integrate RNN and CRF into a unified framework and develop a joint optimization approach, instead of training word

embeddings and a CRF separately as in (Yin et al., 2016). Note that Weiss et al. (2015) proposed to combine deep learning and structured learning for language parsing which can be learned by structured perceptron. However, they also separate neural network training with structured prediction.

2.3 Recursive Neural Networks

Among deep learning methods, RNN has shown promising results on various NLP tasks, such as learning phrase representations (Socher et al., 2010), sentence-level sentiment analysis (Socher et al., 2013), language parsing (Socher et al., 2011a), and question answering (Iyyer et al., 2014). The tree structures used for RNNs include constituency tree and dependency tree. In a constituency tree, all the words lie at leaf nodes, each internal node represents a phrase or a constituent of a sentence, and the root node represents the entire sentence (Socher et al., 2010; Socher et al., 2012; Socher et al., 2013). In a dependency tree, each node including terminal and nonterminal nodes, represents a word, with dependency connections to other nodes (Socher et al., 2014; Iyyer et al., 2014). The resultant model is known as dependency-tree RNN (DT-RNN). An advantage of using dependency tree over the other is the ability to extract word-level representations considering syntactic relations and semantic robustness. Therefore, we adopt DT-RNN in this work.

3 Problem Statement

Suppose that we are given a training set of customer reviews in a specific domain, denoted by $S = \{s_1, ..., s_N\}$, where N is the number of review sentences. For any $s_i \in S$, there may exist a set of aspect terms $A_i = \{a_{i1}, ..., a_{il}\}$, where each $a_{ij} \in A_i$ can be a single word or a sequence of words expressing explicitly some aspect of an entity, and a set of opinion terms $O_i = \{o_{i1}, ..., o_{im}\}$, where each o_{ir} can be a single word or a sequence of words expressing the subjective sentiment of the comment holder. The task is to learn a classifier to extract the set of aspect terms A_i and the set of opinion terms O_i from each review sentence $s_i \in S$.

This task can be formulated as a sequence tagging problem by using the BIO encoding scheme. Specifically, each review sentence s_i is composed of a sequence of words $s_i = \{w_{i1}, ..., w_{in_i}\}$. Each word $w_{ip} \in s_i$ is labeled as one out of the following 5 classes: BA (beginning of aspect), IA (inside of aspect), BO (beginning of opinion), IO (inside of opinion), and O (others). Let $\mathbf{L} = \{\text{BA}, \text{IA}, \text{BO}, \text{IO}, \text{O}\}$. We are also given a test set of review sentences denoted by $S' = \{s'_1, ..., s'_{N'}\}$, where N' is the number of test reviews. For each test review $s'_i \in S'$, our objective is to predict the class label $y'_{iq} \in \mathbf{L}$ for each word w'_{iq}. Note that a sequence of predictions with BA at the beginning followed by IA are indication of one aspect, which is similar for opinion terms.[2]

4 Recursive Neural CRFs

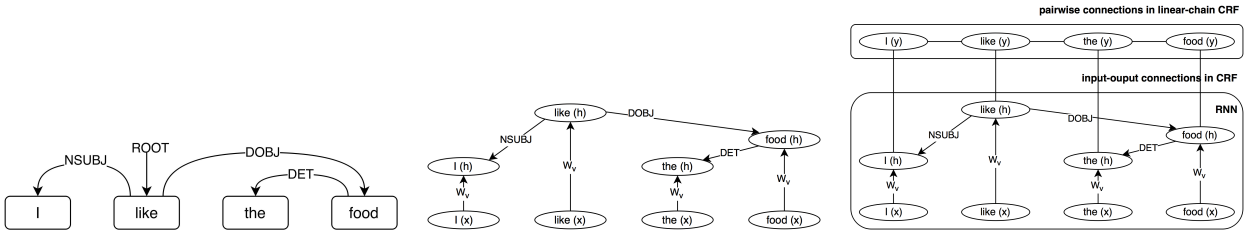
As described in Section 1, RNCRF consists of two main components: 1) a DT-RNN to learn a high-level representation for each word in a sentence, and 2) a CRF to take the learned representation as input to capture context around each word for explicit aspect and opinion terms extraction. Next, we present these two components in details.

4.1 Dependency-Tree RNNs

We begin by associating each word w in our vocabulary with a feature vector $x \in \mathbb{R}^d$, which corresponds to a column of a word embedding matrix $W_e \in \mathbb{R}^{d \times v}$, where v is the size of the vocabulary. For each sentence, we build a DT-RNN based on the corresponding dependency parse tree with word embeddings as initialization. An example of the dependency parse tree is shown in Figure 1(a), where each edge starts from the parent and points to its dependent with a syntactic relation.

In a DT-RNN, each node n, including leaf nodes, internal nodes and the root node, in a specific sentence is associated with a word w, an input feature vector x_w, and a hidden vector $h_n \in \mathbb{R}^d$ of the same dimension as x_w. Each dependency relation r is associated with a separate matrix $W_r \in \mathbb{R}^{d \times d}$. In addition, a common transformation matrix $W_v \in \mathbb{R}^{d \times d}$ is introduced to map the word embedding x_w at node n to its corresponding hidden vector h_n.

[2] In this work we focus on extraction of aspect and opinion terms, not polarity predictions on opinion terms. Polarity prediction can be done by either post-processing on the extracted opinion terms or redefining the BIO labels by encoding the polarity information.



(a) Example of a dependency tree. (b) Example of a DT-RNN tree structure. (c) Example of a RNCRF structure.

Figure 1: Examples of dependency tree, DT-RNN structure and RNCRF structure for a review sentence.

Along with a particular dependency tree, a hidden vector h_n is computed from its own word embedding x_w at node n with the transformation matrix W_v and its children's hidden vectors $h_{\text{child(n)}}$ with the corresponding relation matrices $\{W_r\}$'s. For instance, given the parse tree shown in Figure 1(a), we first compute the leaf nodes associated with the words *I* and *the* using W_v as follows,

$$\begin{aligned} h_{\text{I}} &= f(W_v \cdot x_{\text{I}} + b), \\ h_{\text{the}} &= f(W_v \cdot x_{\text{the}} + b), \end{aligned}$$

where f is a nonlinear activation function and b is a bias term. In this paper, we adopt $tanh(\cdot)$ as the activation function. Once the hidden vectors of all the leaf nodes are generated, we can recursively generate hidden vectors for interior nodes using the corresponding relation matrix W_r and the common transformation matrix W_v as follows,

$$\begin{aligned} h_{\text{food}} &= f(W_v \cdot x_{\text{food}} + W_{\text{DET}} \cdot h_{\text{the}} + b), \\ h_{\text{like}} &= f(W_v \cdot x_{\text{like}} + W_{\text{DOBJ}} \cdot h_{\text{food}} \\ &\quad + W_{\text{NSUBJ}} \cdot h_{\text{I}} + b). \end{aligned}$$

The resulting DT-RNN is shown in Figure 1(b). In general, a hidden vector for any node n associated with a word vector x_w can be computed as follows,

$$h_n = f\left(W_v \cdot x_w + b + \sum_{k \in \mathcal{K}_n} W_{r_{nk}} \cdot h_k\right), \quad (1)$$

where $\mathcal{K}_n$ denotes the set of children of node n, r_{nk} denotes the dependency relation between node n and its child node k, and h_k is the hidden vector of the child node k. The parameters of DT-RNN, $\Theta_{\text{RNN}} = \{W_v, W_r, W_e, b\}$, are learned during training.

4.2 Integration with CRFs

CRFs are a discriminative graphical model for structured prediction. In RNCRF, we feed the output of DT-RNN, i.e., the hidden representation of each word in a sentence, to a CRF. Updates of parameters for RNCRF are carried out successively from the top to bottom, by propagating errors through CRF to the hidden layers of RNN (including word embeddings) using backpropagation through structure (BPTS) (Goller and Küchler, 1996).

Formally, for each sentence s_i, we denote the input for CRF by $\mathbf{h}_i$, which is generated by DT-RNN. Here $\mathbf{h}_i$ is a matrix with columns of hidden vectors $\{h_{i1}, ..., h_{in_i}\}$ to represent a sequence of words $\{w_{i1}, ..., w_{in_i}\}$ in a sentence s_i. The model computes a structured output $\mathbf{y}_i = \{y_{i1}, ..., y_{in_i}\} \in \mathbf{Y}$, where $\mathbf{Y}$ is a set of possible combinations of labels in label set $\mathbf{L}$. The entire structure can be represented by an undirected graph $G = (V, E)$ with cliques $c \in C$. In this paper, we employed linear-chain CRF, which has two different cliques: unary cliques (U) representing input-output connection, and pairwise cliques (P) representing adjacent output connections, as shown in Figure 1(c). During inference, the model aims to output $\hat{\mathbf{y}}$ with the maximum conditional probability $p(\mathbf{y}|\mathbf{h})$. (We drop the subscript i here for simplicity.) The probability is computed from potential outputs of the cliques:

$$p(\mathbf{y}|\mathbf{h}) = \frac{1}{Z(\mathbf{h})} \prod_{c \in C} \psi_c(\mathbf{h}, \mathbf{y}_c), \quad (2)$$

where $Z(\mathbf{h})$ is the normalization term, and $\psi_c(\mathbf{h}, \mathbf{y}_c)$ is the potential of clique c, computed as $\psi_c(\mathbf{h}, \mathbf{y}_c) = \exp\langle W_c, F(\mathbf{h}, \mathbf{y}_c)\rangle$, where the RHS is the exponential of a linear combination of feature vector $F(\mathbf{h}, \mathbf{y}_c)$ for clique c, and the weight vector W_c is tied for unary and pairwise cliques. We

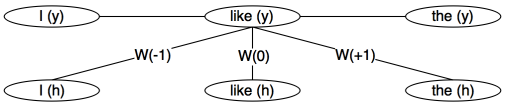


Figure 2: An example for computing input-ouput potential for the second position *like*.

also incorporate a context window of size $2T+1$ when computing unary potentials (T is a hyper-parameter). Thus, the potential of unary clique at node k can be written as

$$\psi_U(\mathbf{h}, y_k) = \exp\Bigg((W_0)_{y_k} \cdot h_k + \sum_{t=1}^{T} (W_{-t})_{y_k} \cdot h_{k-t} + \sum_{t=1}^{T} (W_{+t})_{y_k} \cdot h_{k+t} \Bigg), \quad (3)$$

where W_0, W_{+t} and W_{-t} are weight matrices of the CRF for the current position, the t-th position to the right, and the t-th position to the left within context window, respectively. The subscript y_k indicates the corresponding row in the weight matrix.

For instance, Figure 2 shows an example of window size 3. At the second position, the input features for *like* are composed of the hidden vectors at position 1 (h_{I}), position 2 (h_{like}) and position 3 (h_{the}). Therefore, the conditional distribution for the entire sequence $\mathbf{y}$ in Figure 1(c) can be calculated as

$$p(\mathbf{y}|\mathbf{h}) = \frac{1}{Z(\mathbf{h})} \exp\Bigg(\sum_{k=1}^{4} (W_0)_{y_k} \cdot h_k + \sum_{k=2}^{4} (W_{-1})_{y_k} \cdot h_{k-1} + \sum_{k=1}^{3} (W_{+1})_{y_k} \cdot h_{k+1} + \sum_{k=1}^{3} V_{y_k, y_{k+1}} \Bigg),$$

where the first three terms in the exponential of the RHS consider unary clique while the last term considers the pairwise clique with matrix V representing pairwise state transition score. For simplicity in description on parameter updates, we denote the log-potential for clique $c \in \{U, P\}$ by $g_c(\mathbf{h}, \mathbf{y}_c) = \langle W_c, F(\mathbf{h}, \mathbf{y}_c) \rangle$.

4.3 Joint Training for RNCRF

Through the objective of maximum likelihood, updates for parameters of RNCRF are first conducted on the parameters of the CRF (unary weight matrices $\Theta_U = \{W_0, W_{+t}, W_{-t}\}$ and pairwise weight matrix V) by applying chain rule to log-potential updates. Below is the gradient for Θ_U (updates for V are similar through the log-potential of pairwise clique $g_P(y'_k, y'_{k+1})$):

$$\triangle \Theta_U = \frac{\partial - \log p(\mathbf{y}|\mathbf{h})}{\partial g_U(\mathbf{h}, y'_k)} \cdot \frac{\partial g_U(\mathbf{h}, y'_k)}{\partial \Theta_U}, \quad (4)$$

where

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{h})}{\partial g_U(\mathbf{h}, y'_k)} = -(1_{y_k = y'_k} - p(y'_k|\mathbf{h})), \quad (5)$$

and y'_k represents possible label configuration of node k. The hidden representations of each word and the parameters of DT-RNN are updated subsequently by applying chain rule with (5) through BPTS as follows,

$$\triangle h_{\text{root}} = \frac{\partial - \log p(\mathbf{y}|\mathbf{h})}{\partial g_U(\mathbf{h}, y'_{\text{root}})} \cdot \frac{\partial g_U(\mathbf{h}, y'_{\text{root}})}{\partial h_{\text{root}}}, \quad (6)$$

$$\triangle h_{k \neq \text{root}} = \frac{\partial - \log p(\mathbf{y}|\mathbf{h})}{\partial g_U(\mathbf{h}, y'_k)} \cdot \frac{\partial g_U(\mathbf{h}, y'_k)}{\partial h_k} + \triangle h_{\text{par(k)}} \cdot \frac{\partial h_{\text{par}(k)}}{\partial h_k}, \quad (7)$$

$$\triangle \Theta_{\text{RNN}} = \sum_{k=1}^{K} \frac{\partial - \log p(\mathbf{y}|\mathbf{h})}{\partial h_k} \cdot \frac{\partial h_k}{\partial \Theta_{\text{RNN}}}, \quad (8)$$

where h_{root} represents the hidden vector of the word pointed by ROOT in the corresponding DT-RNN. Since this word is the topmost node in the tree, it only inherits error from the CRF output. In (7), $h_{\text{par}(k)}$ denotes the hidden vector of the parent node of node k in DT-RNN. Hence the lower nodes receive error from both the CRF output and error propagation from parent node. The parameters within DT-RNN, Θ_{RNN}, are updated by applying chain rule with respect to updates of hidden vectors, and aggregating among all associated nodes, as shown in (8). The overall procedure of RNCRF is summarized in Algorithm 1.

5 Discussion

The best performing system (Toh and Wang, 2014) for SemEval challenge 2014 task 4 (subtask 1) employed CRFs with extensive hand-crafted features including those induced from dependency trees. However, their experiments showed that the addition of the features induced from dependency relations does not improve the performance. This indicates

Algorithm 1 Recursive Neural CRFs

Input: A set of customer review sequences: $S = \{s_1, ..., s_N\}$, and feature vectors of d dimensions for each word $\{x_w\}$'s, window size T for CRFs
Output: Parameters: $\Theta = \{\Theta_{\text{RNN}}, \Theta_U, V\}$
Initialization: Initialize W_e using *word2vec*. Initialize W_v and $\{W_r\}$'s randomly with uniform distribution between $\left[-\frac{\sqrt{6}}{\sqrt{2d+1}}, \frac{\sqrt{6}}{\sqrt{2d+1}}\right]$. Initialize W_0, $\{W_{+t}\}$'s, $\{W_{-t}\}$'s, V, and b with all 0's
for each sentence s_i **do**
1: Use DT-RNN (1) to generate $\mathbf{h}_i$
2: Compute $p(\mathbf{y}_i|\mathbf{h}_i)$ using (2)
3: Use the backpropagation algorithm to update parameters Θ through (4)-(8)
end for

the difficulty of incorporating dependency structure explicitly as input features, which motivates the design of our model to use DT-RNN to encode dependency between words for feature learning. The most important advantage of RNCRF is the ability to learn the underlying dual propagation between aspect and opinion terms from the tree structure itself. Specifically as shown in Figure 1(c), where the aspect is *food* and the opinion expression is *like*. In the dependency tree, *food* depends on *like* with the relation *DOBJ*. During training, RNCRF computes the hidden vector h_{like} for *like*, which is also obtained from h_{food}. As a result, the prediction for *like* is affected by h_{food}. This is one-way propagation from *food* to *like*. During backpropagation, the error for *like* is propagated through a top-down manner to revise the representation h_{food}. This is the other-way propagation from *like* to *food*. Therefore, the dependency structure together with the learning approach help to enforce the dual propagation of aspect-opinion pairs as long as the dependency relation exists, either directly or indirectly.

5.1 Adding Linguistic/Lexicon Features

RNCRF is an end-to-end model, where feature engineering is not necessary. However, it is flexible to incorporate light hand-crafted features into RNCRF to further boost its performance, such as features from POS tags, name-list, or sentiment lexicon. These features could be appended to the hidden vector of each word, but keep fixed during training, unlike learnable neural inputs and the CRF weights as described in Section 4.3. As will be shown in experiments, RNCRF without any hand-crafted features slightly outperforms the best performing systems that involve heavy feature engineering efforts, and RNCRF with light feature engineering can achieve even better performance.

Domain	Training	Test	Total
Restaurant	3,041	800	3,841
Laptop	3,045	800	3,845
Total	6,086	1,600	7,686

Table 1: SemEval Challenge 2014 task 4 dataset

6 Experiment

6.1 Dataset and Experimental Setup

We evaluate our model on the dataset from SemEval Challenge 2014 task 4 (subtask 1), which includes reviews from two domains: restaurant and laptop[3]. The detailed description of the dataset is given in Table 1. As the original dataset only includes manually annotate labels for aspect terms but not for opinion terms, we manually annotated opinion terms for each sentence by ourselves to facilitate our experiments.

For word vector initialization, we train word embeddings with word2vec (Mikolov et al., 2013) on the Yelp Challenge dataset[4] for the restaurant domain and on the Amazon reviews dataset[5] (McAuley et al., 2015) for the laptop domain. The Yelp dataset contains 2.2M restaurant reviews with 54K vocabulary size. For the Amazon reviews, we only extracted the electronic domain that contains 1M reviews with 590K vocabulary size. We vary different dimensions for word embeddings and chose 300 for both domains. Empirical sensitivity studies on different dimensions of word embeddings are also conducted. Dependency trees are generated using Stanford Dependency Parser (Klein and Manning, 2003). Regarding CRFs, we implement a linear-chain CRF using CRFSuite (Okazaki, 2007). Because of the relatively small size of training data and a large number of parameters, we perform pre-training on the parameters of DT-RNN with cross-

[3]Experiments with more publicly available datasets, e.g. restaurant review dataset from SemEval Challenge 2015 task 12 will be conducted in our future work.

[4]http://www.yelp.com/dataset_challenge

[5]http://jmcauley.ucsd.edu/data/amazon/links.html

entropy error, which is a common strategy for deep learning (Erhan et al., 2009). We implement mini-batch stochastic gradient descent (SGD) with a batch size of 25, and an adaptive learning rate (AdaGrad) initialized at 0.02 for pretraining of DT-RNN, which runs 4 epochs for the restaurant domain and 5 epochs for the laptop domain. For parameter learning of the joint model RNCRF, we implement SGD with a decaying learning rate initialized at 0.02. We also try with varying context window size, and use 3 for the laptop domain and 5 for the restaurant domain, respectively. All parameters are chosen by cross validation.

As discussed in Section 5.1, hand-crafted features can be easily incorporated into RNCRF. We generate three types of simple features based on POS tags, name-list and sentiment lexicon to show further improvement by incorporating these features. Following (Toh and Wang, 2014), we extract two sets of name list from the training data for each domain, where one includes high-frequency aspect terms, and the other includes high-probability aspect words. These two sets are used to construct two lexicon features, i.e. we build a 2D binary vector: if a word is in a set, the corresponding value is 1, otherwise 0. For POS tags, we use Stanford POS tagger (Toutanova et al., 2003), and convert them to universal POS tags that have 15 different categories. We then generate 15 one-hot POS tag features. For sentiment lexicon, we use the collection of commonly used opinion words (around 6,800) (Hu and Liu, 2004a). Similar to name list, we create a binary feature to indicate whether the word belongs to opinion lexicon. We denote by RNCRF+F the proposed model with the three types of features.

Compared to the winning systems of SemEval Challenge 2014 task 4 (subtask 1), RNCRF or RNCRF+F uses additional labels of opinion terms for training. Therefore, to conduct fair comparison experiments with the winning systems, we implement RNCRF-O by omitting opinion labels to train our model (i.e., labels become "BA", "IA", "O"). Accordingly, we denote by RNCRF-O+F the RNCRF-O model with the three additional types of hand-crafted features.

6.2 Experimental Results

We compare our model with several baselines:

- **CRF-1**: a linear-chain CRF with standard linguistic features including word string, stylistics, POS tag, context string, and context POS tags.
- **CRF-2**: a linear-chain CRF with both standard linguistic features and dependency information including head word, dependency relations with parent token and child tokens.
- **LSTM**: an LSTM network built on top of word embeddings proposed by (Liu et al., 2015). We keep original settings in (Liu et al., 2015) but replace their word embeddings with ours (300 dimension). We try different hidden layer dimensions (50, 100, 150, 200) and reported the best result with size 50.
- **LSTM+F**: the above LSTM model with the three additional types of hand-crafted features as with RNCRF.
- **SemEval-1**, **SemEval-2**: the top two winning systems for SemEval challenge 2014 task 4 (subtask 1).
- **WDEmb+B+CRF**[6]: the model proposed by (Yin et al., 2016) using word and dependency path embeddings combined with linear context embedding features, dependency context embedding features and hand-crafted features (i.e., feature engineering) as CRF input.

The comparison results are shown in Table 2 for both the restaurant domain and the laptop domain. Note that we provide the same annotated dataset (both aspect labels and opinion labels are included for training) for CRF-1, CRF-2 and LSTM for fair comparison. It is clear that our proposed model RNCRF achieves superior performance compared with most of the baseline models. The performance is even better by adding simple hand-crafted features, i.e., RNCRF+F, with 0.92% and 3.87% absolute improvement over the best system in the challenge for aspect extraction for the restaurant domain and the laptop domain, respectively. This shows the advantage of

[6]We report the best results from the original paper (Yin et al., 2016).

	Restaurant		Laptop	
Models	Aspect	Opinion	Aspect	Opinion
SemEval-1	84.01	-	74.55	-
SemEval-2	83.98	-	73.78	-
WDEmb+B+CRF	**84.97**	-	75.16	-
CRF-1	77.00	78.95	66.21	71.78
CRF-2	78.37	78.65	68.35	70.05
LSTM	81.15	80.22	72.73	74.98
LSTM+F	82.99	82.90	73.23	77.67
RNCRF-O	82.73	-	74.52	-
RNCRF-O+F	84.25	-	77.26	-
RNCRF	84.05	80.93	76.83	76.76
RNCRF+F	84.93	**84.11**	**78.42**	**79.44**

Table 2: Comparison results in terms of F1 scores.

	Restaurant		Laptop	
Models	Aspect	Opinion	Aspect	Opinion
DT-RNN+SoftMax	72.45	69.76	66.11	64.66
CRF+word2vec	82.57	78.83	63.62	56.96
RNCRF	84.05	80.93	76.83	76.76
RNCRF+POS	84.08	81.48	77.04	77.45
RNCRF+NL	84.24	81.22	78.12	77.20
RNCRF+Lex	84.21	**84.14**	77.15	78.56
RNCRF+F	**84.93**	84.11	**78.42**	**79.44**

Table 3: Impact of different components.

combining high-level continuous features and discrete hand-crafted features. Though CRFs usually show promising results in sequence tagging problems, they fail to achieve comparable performance when lacking of extensive features (e.g., CRF-1). By adding dependency information explicitly in CRF-2, the result only improves slightly for aspect extraction. Alternatively, by incorporating dependency information into a deep learning model (e.g., RNCRF), the result shows more than 7% improvement for aspect extraction and 2% for opinion extraction.

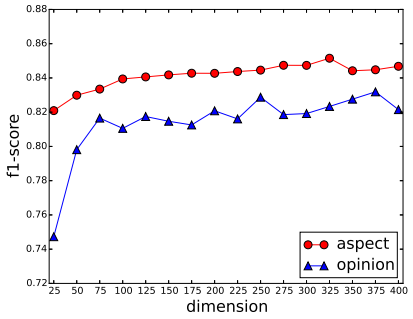
By removing the labels for opinion terms, RNCRF-O produces inferior results than RNCRF because the effect of dual propagation of aspect and opinion pairs disappears with the absence of opinion labels. This verifies our previous assumption that DT-RNN could learn the interactive effects within aspects and opinions. However, the performance of RNCRF-O is still comparable to the top systems and even better with the addition of simple linguistic features: 0.24% and 2.71% superior than the best system in the challenge for the restaurant domain and the laptop domain, respectively. This shows the robustness of our model even without additional opinion labels.

LSTM has shown comparable results for aspect extraction (Liu et al., 2015). However, in their work, they used well-pretrained word embeddings by training with large corpus or extensive external resources, e.g., chunking, and NER. To compare their model with RNCRF, we re-implement LSTM with the same word embedding strategy and labeling resources as ours. The results show that our model outperforms LSTM in aspect extraction by 2.90% and 4.10% for the restaurant domain and the laptop domain, respectively. We conclude that a standard LSTM model fails to extract the relations between aspect and opinion terms. Even with the addition of same linguistic features, LSTM is still inferior than RNCRF itself in terms of aspect extraction. Moreover, our result is comparable with WDEmb+B+CRF in the restaurant domain and better in the laptop domain (+3.26%). Note that WDEmb+B+CRF appended dependency context information into CRF while our model encode such information into high-level representation learning.

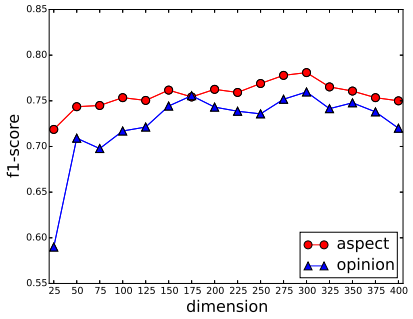
To test the impact of each component of RNCRF and the three types of hand-crafted features, we conduct experiments on different model settings:

- **DT-RNN+SoftMax**: rather than using a CRF, a softmax classifier is used on top of DT-RNN.
- **CRF+word2vec**: a linear-chain CRF with word embeddings only without using DT-RNN.
- **RNCRF+POS/NL/Lex**: the RNCRF model with POS tag or name list or sentiment lexicon feature(s).

The comparison results are shown in Table 3. Similarly, both aspect and opinion term labels are provided for training for each of the above models. Firstly, RNCRF achieves much better results compared to DT-RNN+SoftMax (+11.60% and +10.72% for the restaurant domain and the laptop domain in aspect extraction). This is because DT-RNN fails to fully exploit context information for sequence labeling, which, in contrast, can be achieved by CRF. Secondly, RNCRF outperforms CRF+word2vec, which proves the importance of



(a) On the restaurant domain.



(b) On the laptop domain.

Figure 3: Sensitivity studies on word embeddings.

DT-RNN for modeling interactions between aspects and opinions. Hence, the combination of DT-RNN and CRF inherits the advantages from both models. Moreover, by separately adding hand-crafted features, we can observe that name-list-based features and the sentiment lexicon feature are most effective for aspect extraction and opinion extraction, respectively. This may be explained by the fact that name-list-based features usually contain informative evident for aspect terms and sentiment lexicon provides explicit indication about opinions.

Besides the comparison experiments, we also conduct sensitivity test for our proposed model in terms of word vector dimensions. We tested a set of different dimensions ranging from 25 to 400, with 25 increment. The sensitivity plot is shown in Figure 3. The performance for aspect extraction is smooth with different vector lengths for both domains. For restaurant domain, the result is stable when dimension is larger than or equal to 100, with the highest at 325. For the laptop domain, the best result is at dimension 300, but with relatively small variations. For opinion extraction, the performance reaches a good level when the dimension is larger than or equal to 75 for the restaurant domain and 125 for the laptop domain. This proves the stability and robustness of our model.

7 Conclusion

We have presented a joint model, RNCRF, that achieves the state-of-the-art performance for explicit aspect and opinion term extraction on a benchmark dataset. With the help of DT-RNN, high-level features can be learned by encoding the underlying dual propagation of aspect-opinion pairs. RNCRF combines the advantages of DT-RNNs and CRFs, and thus outperforms the traditional rule-based methods in terms of flexibility, because aspect terms and opinion terms are not only restricted to certain observed relations and POS tags. Compared to feature engineering methods with CRFs, the proposed model saves much effort in composing features, and it is able to extract higher-level features obtained from non-linear transformations.

Acknowledgements

This research is partially funded by the Economic Development Board and the National Research Foundation of Singapore. Sinno J. Pan thanks the support from Fuji Xerox Corporation through joint research on *Multilingual Semantic Analysis* and the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020.

References

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*, pages 49–54.

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, pages 153–160.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 97–110.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, pages 347–352.

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

Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. In *AAAI*, pages 755–760.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *EMNLP*, pages 1035–1045.

Wei Jin and Hung Hay Ho. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *ICML*, pages 465–472.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Himabindu Lakkaraju, Richard Socher, and Christopher D. Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*.

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

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *COLING*, pages 653–661.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *EMNLP-CoNLL*, pages 1346–1356.

Kang Liu, Liheng Xu, Yang Liu, and Jun Zhao. 2013a. Opinion target extraction using partially-supervised word alignment model. In *IJCAI*, pages 2134–2140.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2013b. A logic programming approach to aspect extraction in opinion mining. In *WI*, pages 276–283.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, pages 1433–1443.

Bing Liu. 2011. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Second Edition*. Data-Centric Systems and Applications. Springer.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: An optimization approach. In *WWW*, pages 347–356.

Tengfei Ma and Xiaojun Wan. 2010. Opinion target extraction in Chinese news comments. In *COLING*, pages 782–790.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *ICDM*, pages 1020–1025.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52.

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

Nir Ofek, Soujanya Poria, Lior Rokach, Erik Cambria, Amir Hussain, and Asaf Shabtai. 2016. Unsupervised commonsense knowledge enrichment for domain-specific sentiment analysis. *Cognitive Computation*, 8(3):467–477.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2).

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*, pages 27–35.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *EMNLP*, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks. In *NIPS*, pages 1–9.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *CoRR*, abs/1512.01100.

Ivan Titov and Ryan T. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, pages 308–316.

Zhiqiang Toh and Wenting Wang. 2014. DLIREC: Aspect term extraction and term polarity classification system. In *SemEval*, pages 235–240.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, pages 173–180.

Hao Wang and Martin Ester. 2014. A sentiment-aligned topic model for product aspect rating prediction. In *EMNLP*, pages 1192–1202.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *KDD*, pages 618–626.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *ACL-IJCNLP*, pages 323–333.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *EMNLP*, pages 1533–1541.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *IJCAI*, pages 2979–2985.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *COLING*, pages 1462–1470.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *CIKM*, pages 43–50.

Extracting Aspect Specific Opinion Expressions

Abhishek Laddha
IIT Delhi *
New Delhi, India, 110016
laddhaabhishek11@gmail.com

Arjun Mukherjee
Department of Computer Science,
University of Houston, TX, USA
arjun@cs.uh.edu

Abstract

Opinionated expression extraction is a central problem in fine-grained sentiment analysis. Most existing works focus on either generic subjective expression or aspect expression extraction. However, in opinion mining, it is often desirable to mine the aspect specific opinion expressions (or aspect-sentiment phrases) containing both the aspect and the opinion. This paper proposes a hybrid generative-discriminative framework for extracting such expressions. The hybrid model consists of (i) an unsupervised generative component for modeling the semantic coherence of terms (words/phrases) based on their collocations across different documents, and (ii) a supervised discriminative sequence modeling component for opinion phrase extraction. Experimental results using Amazon.com reviews demonstrate the effectiveness of the approach that significantly outperforms several state-of-the-art baselines.

1 Introduction

Aspect based sentiment analysis is one of the main frameworks in opinion mining (Liu and Zhang, 2012). Most of the websites only display the aggregated ratings of products but people are more interested in fine-grained opinions that capture aspect specific properties in reviews. Therefore, it is desirable to have a holistic approach to mine aspect specific opinion expressions containing both aspect and opinion terms within the sentence context as a composite aspect-sentiment phrase (e.g., "had to flash firmware everyday", "clear directions in voice" etc.) and further group them under coherent aspect categories. Apart from knowing the key issues in products that are often expressed via aspect-sentiment phrases, they are also useful in applications such as comparing similar products and summarizing their important features where it is more convenient to have the aspect-sentiment phrases rather than generic aspect/sentiment words lacking the natural aspect opinion correspondence in the right context. They can also be applied to the various tasks such as sentiment classification, comparative aspect evaluations, aspect rating prediction, etc.

The thread of research in (Brody and Elhadad, 2010; Titov and McDonald, 2008; Zhao et al., 2010; Mei et al., 2007; Jo and Oh, 2011) focus on extracting and grouping aspect and opinion words via generative models but lack the natural aspect opinion correspondence (e.g., in the manner they appear in sentences). (Wang et al., 2016; Fei et al., 2016) can discover aspect specific opinion unigrams but does not focus on phrases. The thread on fine grained opinion expressions (Wiebe et al., 2005; Choi et al., 2006; Breck et al., 2007) focus on subjective expression extraction which are generic instead of aspect specific. Formally, the task can be stated as follows:

Given a set of reviews, for each sentence, $s = (w_1, \ldots w_n)$, with the head aspect $(HA), w_{HA=i}, i \in [1, n]$, discover a sub-sequence $(w_p, \ldots w_q)$ where $p \leq i \leq q$ that best describes the aspect-sentiment phrase containing the head aspect. We refer head aspect to the word describing

Research performed during author's internship at University of Houston

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 627–637,
Austin, Texas, November 1-5, 2016.

fine-grained property of product. Further, group these phrases under relevant aspect categories. The examples below show labeled aspect-sentiment phrases within [[ ]] with the head aspect (HA) italicized:

- I've been very happy with it so far done a [[*firmware* update without a hitch]].
- After less than two years, the [[*signal* became spotty]].

In this paper, we propose a novel hybrid model to solve the problem. We call this Phrase Sentiment Model (PSM). PSM is capable of extracting a myriad of expression types covering: verb phrases ("screen has poor viewability"), noun, adjective or adverbial phrase ("recurrent black screen of death", "quite stable and fast connection"), implied positive ("voice activated directions"), implied negative ("requires reboot every few hours") etc. The hybrid framework facilitates holistic modeling that caters for varied expression types (leveraging its discriminative sequence model) and also grouping them under relevant aspect categories with context (exploiting its generative framework). Our approach is also context and polarity independent facilitating generic aspect-sentiment phrase extraction in any domain.

Further, we propose a novel sampling scheme based on Generalized Pólya urn models that optimizes phrasal collocations to improve coherence. To the best of our knowledge, a hybrid framework has not been attempted before for opinion phrase extraction. Additionally, the paper produced a labeled dataset of aspect specific opinion phrases across 4 domains containing more than 5200 sentences coded with phrase boundaries across both positive and negative polarities which will be released to serve as a language resource. Experimental evaluation shows that our approach outperformed the baselines by a large margin.

2 Related Work

Subjective expression extraction (Choi et al., 2005) has traditionally used sequence models (e.g., CRFs). Various parsing, syntactic, lexical and dictionary based features (Kim and Hovy, 2006; Jakob and Gurevych, 2010; Kobayashi et al., 2007) have been used for subjective expression extraction. In (Yang and Cardie, 2013; Johansson and Moschitti, 2011) dependency relations were also used for opinion expression extraction. Sauper et al., (2011) employs an HMM over words and model the latent topics as states in an HMM to discover the product properties (often aspects) and its associated attributes (pos/neg) polarities separately. In Yang and Cardie, (2012) a semi-CRF based approach is used which allow sequence labeling at segment level and (Yang and Cardie, 2014) employed semi-CRF for opinion expression intensity and polarity classification. However, all the above works focus on generic subjective expressions as opposed to aspect specific opinion-sentiment phrases.

In (Choi et al., 2006; Yang and Cardie, 2013) joint models were proposed for identifying opinion holders and expression, relations among them in news articles. In (Johansson and Moschitti, 2011) a re-ranking approach was used on the output of a sequence model to improve opinion expression extraction. In (Li et al., 2015; Mukherjee, 2016) subjective expressions implying a negative opinion were discovered using sequence models and markov networks; while in (Berend, 2011) supervised keyphrase extraction was used for phrase extraction. These works mostly relied on word level features under the first-order Markov assumption. Above works are tailored for only expression extraction and do not group coherent phrases under relevant aspect categories.

Another thread of research involves topic phrase mining. Wang et al., (2007) proposes a Topical n-gram model (TNG) that mines phrases based on statistical collocation. Lindsey et al., (2012) employ hierarchical Pitman-Yor process to model phrases. In Fei et al., (2014), Generalized Pólya urn model (LDA-P-GPU) was used to group the candidate noun phrases. In (El-Kishky et al., 2014; Liu et al., 2015), frequency based information were used for mining phrases that are good for generic phrases but cannot model relevant yet longer phrases due to their infrequency. Thus, they are unable to capture long phrases containing both aspect and opinion. The models TNG and LDA-P-GPU are closest to our task as they can discover relevant aspect expressions that can contain opinions and are considered as baselines.

Next there are works that generate phrasal

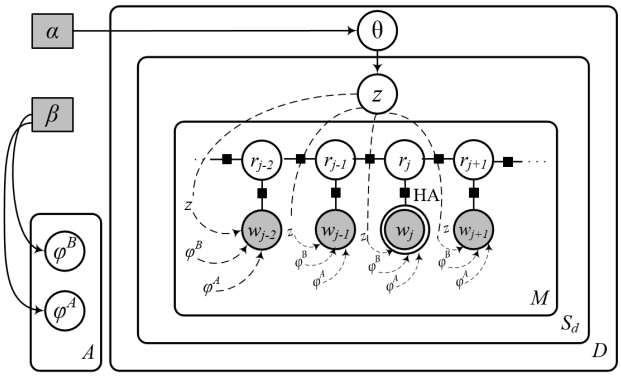


Figure 1: Plate Notation of PSM

datasets. In SemEval 2015, Aspect based Sentiment Analysis Task (Pontiki et al., 2015), a dataset was produced that had annotations for aspect phrases. The focus was on aspect phrases as opposed to aspect-sentiment phrases. The MPQA 2.0 corpus (Wiebe et al., 2005) has some labeled opinion expressions, but they are generic subjective expressions as opposed to aspect-sentiment phrases (see Section 1) we find in reviews.

Zhao et al., (2011) extracts topical phrase in tweets using relevance and interestingness. Wu et al., (2009) proposed a phrase dependency parsing approach to extract product feature (aspect expression) and opinion expression and the relation between them. They considered all noun and verb phrases (NPs, VPs) as product features and its surrounding dictionary opinion words as opinions. Features were constructed using phrase dependency tree to extract relation among all product features and opinions that were later used in aspect and opinion expression extraction. Although, Wu et al., (2009) doesn't discover aspect specific opinion phrases, its use of NPs in extracting candidate opinion phrases is similar to Fei et al., (2014) which is considered as a baseline.

3 Phrase Sentiment Model (PSM)

PSM is a hybrid between generative and discriminative modeling that combines the best of both worlds. Its generative modeling lays the foundation for emission of aspects and aspect specific opinion phrases in documents, while its discriminative sequence modeling component (via an embedded CRF) facilitates aspect specific opinion phrase extraction.

As noted in Titov and McDonald (2008), modeling entire reviews as documents tend to correspond to the global properties of a product (e.g., brand, name, etc.) resulting in rather overlapping aspects. To avoid this, we perform sentence level modeling that helps improve aspect sharpness. A review sentence s_d of N words, is denoted as $w_{d,s,j} = \{w_{d,s,1}, w_{d,s,2} \dots w_{d,s,N}\}$ where each $w_{d,s,j}$ is one of the V words in the vocabulary. There can be exponential number of phrase sequence possible for each sentence s_d i.e. $\langle w_{d,s,j} \rangle_{j=st}^{j=st+len-1}$ of arbitrary length $len (len = 1$ for words; $len > 1$ for phrases) starting at an index $st \in [0, |s_d|]$. We observed that most of opinion expression are centered around the head aspect thereby causing the space of potential opinion expressions to be quite sparse. We took advantage of following observation and trained a sequence model (e.g., CRF) for phrase sequence tagging as described in the next subsection. We generated $M = 5$ best sequence labelings of each sentence $s_d (s_d^{m \in 1 \dots M})$ via forward Viterbi and backward A^* search[1]. Hence, our vocabulary is the union of unigrams and n-grams discovered by CRF over M best sequence labeling, i.e., the model's vocabulary, $V = \{w_{d,s,j}\} \cup \{\langle w_{d,s_d^m,j} \rangle_{j=st}^{j=st+len-1}\} \; \forall d, s, j, len, st, m.$

In PSM, for each aspect a, we model its aspect specific terms (words/phrases) distributions and aspect background word distributions using multinomials φ_a^A and φ_a^B, drawn from $Dir(\beta)$ over the vocabulary $v_{1 \dots V}$. For each domain d, we first draw a domain specific aspect distribution $\theta_d \sim Dir(\alpha)$. Next, for each review sentence (document), s_d of a domain, d we draw an aspect, $z_{d,s} \sim Mult(\theta_d)$. We assume that each sentence evaluates one aspect which mostly holds true in the review domain. We associate a latent switch variable for each word $r_{d,s,j}$ and each phrase $\langle r_{d,s_d^m,j} \rangle_{j=st}^{j=st+len-1}$ of vocabulary where each switch variable $r \in \{0, 1\}$. To generate each term $\langle w_{d,s_d^m,j} \rangle_{j=st}^{j=st+len-1}$ of the labeled sequence $s_d^{m \in \{1 \dots M\}}$, we first set the switch variables for the sentence, s_d via the discriminative CRF model, i.e. $\langle r_{d,s_d^m,j} \rangle_{j=1}^{j=|s_d|} \leftarrow \frac{1}{Z} exp\left(\sum_j \sum_k \lambda_k f_k \left(r_{j-1}, r_j, w_j\right)\right)$ by fitting a previously trained CRF model. The switch variables, $r \in \{1, 0\}$ for a particular tagging s_d^m of

[1]Value of M was tuned via pilot experiments using the CRF++ toolkit (Kudo, 2009)

sentence s_d span over all its words and take values $r = 1$ for words being part of an aspect specific opinion phrase or $r = 0$ for aspect background words, upon observing all words in s_d. Finally, depending upon the aspect, $z_{d,s}$ and the switch variable $r_{d,s,j}$, we emit (unigram) terms in the sentence as follows:

$$w_{d,s,j} \sim \begin{cases} Mult(\varphi^A_{z_{d,s}}) & \text{if } r_{d,s,j} = 1 \\ Mult(\varphi^B_{z_{d,s}}) & \text{if } r_{d,s,j} = 0 \end{cases} \quad (1)$$

and for phrasal terms (i.e., when $\langle r_{d,s^m_d,j}\rangle_{j=1}^{j=st+len-1} = 1 \forall$ valid st and len), we emit $\langle w_{d,s^m_d,j}\rangle_{j=st}^{j=st+len-1} \sim Mult(\varphi^A_{z_{d,s}})$.

3.1 Inference

We employ MCMC Gibbs sampling for posterior inference. As latent variables z and r belong to different levels, we hierarchically sample z and then r for each sweep of a Gibbs iteration as follows:

$$p(z_{d,s} = a|Z_{\neg d,s}, R_{\neg d,s}, W_{\neg d,s}) \propto \frac{(n^s_{d,a})_{\neg d,s}+\alpha}{(n^s_{d,(\cdot)})_{\neg d,s}+A\alpha} \times \left[\left(\prod_{v=1}^{V} \frac{\Gamma(n^A_{a,v}+\beta)}{\Gamma((n^A_{a,v})_{\neg d,s}+\beta)}\right) / \left(\frac{\Gamma(n^A_{a,(\cdot)}+V\beta)}{\Gamma((n^A_{a,(\cdot)})_{\neg d,s}+V\beta)}\right)\right] \times \left[\left(\prod_{v=1}^{V} \frac{\Gamma(n^B_{a,v}+\beta)}{\Gamma((n^B_{a,v})_{\neg d,s}+\beta)}\right) / \left(\frac{\Gamma(n^B_{a,(\cdot)}+V\beta)}{\Gamma((n^B_{a,(\cdot)})_{\neg d,s}+V\beta)}\right)\right] \quad (2)$$

Samplers for r consist of three cases: (i) individual aspect-specific opinion words, (ii) individual background words, and (iii) phrasal opinion:

$$p(r_{d,s,j} = 1|z_{d,s} = a, w_{d,s,j} = v, \ldots) \propto \frac{(n^A_{a,v})_{\neg d,s,j}+\beta)}{(n^A_{a,(\cdot)})_{\neg d,s,j}+V\beta)} \times p_{CRF}(r_{d,s,j-1}, r_{d,s,j} = 1|v) \quad (3)$$

$$p(r_{d,s,j} = 0|z_{d,s} = a, w_{d,s,j} = v, \ldots) \propto \frac{(n^B_{a,v})_{\neg d,s,j}+\beta)}{(n^B_{a,(\cdot)})_{\neg d,s,j}+V\beta)} \times p_{CRF}(r_{d,s,j-1}, r_{d,s,j} = 0|v) \quad (4)$$

$$p(\langle r_{d,s,j} = 1\rangle_{j=st}^{j=st+len-1}|z_{d,s} = a, \langle w_{d,s,j} = v'\rangle_{j=st}^{j=st+len-1}, \ldots) \propto \frac{(n^A_{a,v})_{\neg d,s,j}+\beta)}{(n^A_{a,(\cdot)})_{\neg d,s,j}+V\beta)} \times p_{CRF}(r_{d,s,j=st-1}, r_{d,s,j=st} = 1, r_{d,s,j=st+1} = 1, \ldots r_{d,s,j=st+len-1} = 1|v') \quad (5)$$

where $n^s_{d,a}$ denotes the # of sentences in domain d assigned to aspect a, $n^A_{a,v}, n^B_{a,v}$ denote the # of times term v was assigned to aspect a in the aspect specific opinion, and aspect specific background language models respectively. A count variable with subscript $(\cdot)$ signifies the marginalized sum over the latter index and $\neg$ denotes the discounted counts. The sampler in (5) computes the likelihood of a sequence of contiguous terms, $\langle r_{d,s,j} = 1\rangle_{j=st}^{j=st+len-1}$ forming an aspect ($z_{d,s} = a$) specific opinion phrase, v' starting at index $j = st$ and of length len, where $v' = \langle w_{d,s,j}\rangle_{j=st}^{j=st+len-1}$. The sequence probabilities, p_{CRF} in equations (3, 4, 5) can be obtained as follows.

Let $\mathbf{w} = (w_{t=1} \ldots w_{t=T})$ denote the sequence of observed words in a sentence, and let each observation w_t have a label $y_t \in Y$ indicating whether w_t is part of a aspect specific opinion phrase, where $Y = \{1, 0\}$. We consider a first order Markov linear-chain CRF in our hybrid model. We define the Markovian transition and forward-backward variables of our embedded CRF as follows:

$$\psi_t(j, i, w) = p(y_t = j|y_{t-1} = i)p(w_t = w|y_t = j) \quad (6)$$

$$\alpha_t(j) = \textstyle\sum_{i\in Y} \psi_t(j, i, w_t)\alpha_{t-1}(i) \quad (7)$$

$$\beta_t(j) = \textstyle\sum_{j\in Y} \psi_{t+1}(j, i, w_{t+1})\beta_{t+1}(j) \quad (8)$$

where $\alpha_1(j) = \psi_1(j, y_0, w_1)$ and $\beta_T(i) = 1$. This lays the foundation for expressing the sequence probabilities, p_{CRF} in closed form as follows:

$$p_{CRF}(y_{t-1}, y_t|w) \propto \alpha_{t-1}(y_{t-1})\psi_t(y_t, y_{t-1}, w_t)\beta_t(y_t) \quad (9)$$

$$p_{CRF}(y_{t-2}, y_{t-1}, y_t|w) \propto \alpha_{t-2}(y_{t-2})\psi_{t-1}(y_{t-1}, y_{t-2}, w_{t-1})\psi_t(y_t, y_{t-1}, w_t)\beta_t(y_t) \quad (10)$$

Eq. (9) is used for computing the sequence probabilities for individual opinion/background words for samplers in eq. (3, 4) while eq. (10) and its extensions are used for computing the sequence probabilities in the phrase samplers in eq. (5). The values for $\psi_t, \alpha_t, \beta_t$ are obtained from a previously trained CRF model upon fitting to the current sentence s_d for which sampling is being performed.

3.2 Embedded CRF Training

We employ linear-chain CRFs (Lafferty et al., 2001) for modeling phrases. While word (W) and Part-Of-Speech (POS) tag features are effective in various sequence modeling tasks (Yang and Cardie, 2014; Yang and Cardie, 2012), in our problem context, (W+POS) features are insufficient as they do not consider the head aspect (HA) and its relevant positional/contextual features, i.e., how do different POS tags, syntactic units (chunks), polar sentiments appear in proximity to the head aspect? Hence, centering on the HA, we propose a set of pivot features to model context.

Pivot Features: We consider five feature families:
POS Tags (T): DT, IN, JJ, MD, NN, RB, VB, etc.
Phrase Chunks (C): ADJP, ADVP, NP, PP, VP, etc.
Prefixes (P): *anti, in, mis, non, pre, sub, un,* etc.

Category	Feature Template	Example of feature apperaing in a sentence
1^{st} order features $W_{i+j}; -4 \leq j \leq 4$ $W \in \{T, C, P, S, SP\}$	SP_{i+j}	$SP_{i-1} = NEG$; previous term of HA is of NEG polarity,. . . have this terrible *voice* on the . . .
	S_{i+j}	$S_{i-2} = ing$; suffix of 2^{nd} previous term of head aspect is "ing", . . . kept dropping the *signal* . . .
	. . .	. . .
2^{nd} order features $W_{i+j}, Y_{i+j}; -4 \leq j \leq 4$ $W, Y \in \{T, C, P, S, SP\}$	$T_{i+j}, T_{i+j'}$	$T_{i-2} = JJ, T_{i-1} = VBZ, \cdots$ frequently drops *connection*. . .
	$T_{i+j}, C_{i+j'}$	$T_{i+2} = RB, C_{i+3} = ADJP$; . . . *screen* clarity is good. . .
	. . .	. . .
3^{rd} order features $W_{i+j}, Y_{i+j}, Z_{i+j}; -4 \leq j \leq 4$ $W, Y, Z \in \{T, C, P, S, SP\}$	$T_{i+j}, S_{i+j'}, T_{i+j''}$	$T_{i+2} = JJ, P_{i+4} = un, T_{i+4} = JJ$; . . . *screen* is blank and unresponsive. . .
	. . .	. . .

Table 1: Pivot Templates: Subscript i denotes the index of the head aspect, HA (italicized). Subscript j denotes the index relative to i

Suffixes (S): *able, est, ful, ic, ing, ive, ness* etc.
Word Sentiment Polarity (SP): POS, NEG, NEU

Pivoting on the head aspect, we look forward and backward to generate a family of binary features defined by a specific template (see Table 1). Each template generates several features that capture various positional context around the HA. Additionally, we consider up to 3^{rd} order pivot features allowing us to model various dependencies as features. For polarity, we used the opinion lexicon[2] derived from (Hu and Liu, 2004).

Feature Templates: Table 1 details the templates for features pivoting on the head aspect. Various features from these templates coupled with the value of the current sequence tag at y_t or a combination of current and previous labels y_t, y_{t-1} serve as our linear chain features (LCF), $f(y_{i-1}, y_i, w)$. Further, the index i for LCF can refer to any word in the sentence and not necessarily the head aspect, yielding us a very rich feature space.

Learning the CRF λs: Given a set of training examples $\{\mathbf{w_i}, \overline{y_t}\}$ where $\overline{y_t}$ are the correct sequence tags, we estimate the CRF $\Lambda = \{\lambda_k\}$parameters by minimizing the negative log-likelihood (NLL),

$$\Lambda = argmin_\Lambda(C \sum_i \log(p(\overline{y_t}|\mathbf{w_i}, \Lambda)) + \sum_k \lambda_k^2) \quad (11)$$

Where $p(\overline{y_t}|\mathbf{w}, \Lambda) \propto exp(\sum_k \sum_t \lambda_k f_k(y_{t-1}, y_t, w))$, C is the soft-margin parameter, and the term $\sum_k \lambda_k^2$ indicates L_2 regularization on the feature weights, λ_k. The training set for learning the embedded CRF models λs is detailed in Table 3 (col 1, 2).

[2]http://www.cs.uic.edu/∼liub/FBS/opinion-lexicon-English.rar

4 Optimizing Phrasal Collocations

Topic models can be described in terms of a simple Pólya urn (SPU) sampling schemes in the sense that when a particular term (word or phrase) is drawn from a topic, count of that term is incremented in that topic. This enforces the topic distribution to tend towards these terms over time as frequency of them increases. Therefore, the posterior of generative topic models often favors terms with high frequency e.g., unigrams, while phrasal terms are ranked lower due to their lower frequencies. This is undesirable for phrase extraction.

In contrast, Generalized Pőlya urn (GPU) model differs from SPU in its sampling process. When a certain term is drawn, the count of that term increases as well as it also increases the count of terms which are similar to drawn word/phrases via pseudo-counts for promotion. Thus, GPU caters for promotion of others terms in a principled manner. It has been previously used for unigram topic modeling in (Mimno et al., 2011). In this work we leverage it for phrases.

4.1 Proposed PSM-GPU model

We optimize the collocations of relevant aspect words and phrases in the GPU framework in two ways:

Word to phrase: Intuitively, if an aspect word is assigned to a topic then that topic should represent that aspect and to all other phrases in that aspect's phrase set (i.e., phrases containing that aspect) should belong to the same topic. Thus, when an aspect word is assigned to a topic then each phrase in its aspect set is promoted with a small count in that topic.
Phrase to word: When a phrase $\langle w_{d,s,j} \rangle_{j=st}^{j=st+len-1}$ is assigned to a topic, each component word $w_{d,s,j}$ where $j \in [st, st+len-1]$ in it is also promoted with a certain small count, i.e., each word of that phrase is also assigned to that topic by a certain amount.

We now define the term promotion matrix, A for the GPU framework. Every element of $A, A_{w,w'}$ refers to the promotion pseudocount, i.e., whenever a w was seen in an urn, we increment the count by $A_{w,w'}$ of $w^{'}$. $w, w^{'}$ can be word or phrases.

$$A_{w,w'} = \begin{cases} 1 & \text{if } w = w^{'} \\ \sigma & \text{if } w \text{ is an aspect word,} \\ & w^{'} \text{is a phrase} \in \text{Phrase set of } w \\ \delta & w^{'} \text{is a word} \in \text{Phrase } w \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

To improve the ranking of phrases, the value of σ is kept greater than δ. Empirically values are given section 5.1.

4.2 PSM-GPU Inference

Accounting the GPU process above, the approximate Gibbs samplers for z and r take the following form:

$$p(z_{d,s} = a | Z_{\neg d,s}, R_{\neg d,s}, W_{\neg d,s}) \propto \left[\frac{(n^s_{d,a})_{\neg d,s} + \alpha)}{(n^s_{d,(\cdot)})_{\neg d,s} + A\alpha)} \right] \times \frac{\left(\prod_{v=1}^{V} \frac{\Gamma(\sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'}) + \beta)}{\Gamma(\sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'})_{\neg d,s} + \beta)} \right)}{\left(\frac{\Gamma(\sum_{v=1}^{V} \sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'}) + V\beta)}{\Gamma(\sum_{v=1}^{V} \sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'})_{\neg d,s} + V\beta)} \right)} \times \left[\left(\prod_{v=1}^{V} \frac{\Gamma(n^B_{a,v} + \beta)}{\Gamma((n^B_{a,v})_{\neg d,s} + \beta)} \right) / \left(\frac{\Gamma(n^B_{a,(\cdot)} + V\beta)}{\Gamma((n^B_{a,(\cdot)})_{\neg d,s} + V\beta)} \right) \right] \quad (13)$$

$$p(r_{d,s,j} = 1 | z_{d,s} = a, w_{d,s,j} = v, \ldots) \propto \frac{\sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'})_{\neg d,s,j} + \beta)}{\sum_{v=1}^{V} \sum_{w'=1}^{V} (A_{v,w'} * n^A_{a,w'})_{\neg d,s,j} + V\beta} \times p_{CRF}(r_{d,s,j-1}, r_{d,s,j} = 1 | v) \quad (14)$$

Similarly, phrasal opinion switch variable can be derived. The sampler for individual background words remains unchanged.

5 Experimental Evaluation

In this section, we evaluate our proposed models. We first detail our dataset, followed by baselines and results.

5.1 Dataset and Parameter Settings

Domain	Pos. Labeled Phrase	Neg. Labeled Phrase	Positive	Negative	Total
Router	414	1256	1937	5291	7228
GPS	948	672	2473	2231	4704
Mouse	376	477	1421	2591	4012
Keyboard	398	660	912	1539	2451

Table 3: Statistics of dataset of four domain

Dataset Statistics: For CRF training, we created a phrase labeled dataset of aspect opinion phrases using product reviews from Amazon across 4 domains each spanning 4 head aspects. In this work, head aspects for a domain are known a priori either directly using unsupervised topic induction or guided by domain knowledge (e.g. using aspect models such as (Zhao et al., 2010; Chen et al., 2013; Mukherjee and Liu, 2012)). Our focus is on phrase extraction and grouping. We labeled the positive and negative opinion phrases spans (Table 3; col 2, 3) in the reviews following the annotation schemes in (Wilson et al., 2005) for embedded CRF training in PSM. Table 3 details our labeled data for CRF training. This phrase boundary labeled dataset (Table 3; col 2, 3) is "orthogonal" or disjoint from the data where the PSM model was fit and evaluated (Table 3, col 4, 5). This avoids overfitting and makes a fair case for all the experiments of PSM.
Preprocessing and Parameter Setting: We removed the stopwords, punctuation, special characters and words appearing less than 5 times in each domain. For all models, posterior estimates of latent variables were taken with a sampling lag of 50 iterations post burn-in phase (of 200 iterations) with 2,000 iterations in total. Dirichlet priors were set to $\alpha = 50/K$, where K is the number of topics (empirically set to 10 via pilot) and $\beta = 0.1$. The CRF parameters $C = 1$ and GPU parameters $\sigma = 0.05$ and $\delta = 0.01$ were estimated using cross validation.

5.2 Baselines

We consider the following relevant phrase extraction models as our baselines:

PSM-GPU	sMC-GPU	LDA-P-GPU
Router→Connection:"updating firmware secure connection", "dropping connection", "connection excellent", "instability wireless connection", "crashes entire connection", *"updating"*, "kills current connection including downloads", "affected plugged connection",*"cable radio frequency connection", "halfway"*	**Router→Connection:** *"connection big time", "signal weak time connection", "connection time stable"*, "lose connection", "internet connection speed dropped", *"stop working lot connection"*, "started dropping internet connection", "started dropping connection", "drop wireless connection drops wired connection", "connection dropping problems"	**Router→Connection:** *"drops", "times day internet connection"*, "internet connection multiple times", *"internet connection times", "internet connection minutes"*, "dropping internet connection", "broadband internet connection", "extremely slow internet connection", "lost internet connection",*"internet connection couple"*
GPS→Screen:"poor screen contrast daylight", *"direction screen missing poor screen contrast"*, "slow screen size makes useless", "screen turned", *"turn", "nothing screen"*, "night screen bit bright", "screen unpredictable directions", "lacks faster screen refresh rate", "smoother screen refresh"	**GPS→Screen:** *"bright"*, "excellent nice touch screen", *"touch screen n't"*, "nice big touch screen", "touch screen big size", *"smaller", "touch screen but nice", "screen accurate spoken direction"*, "touch screen nice", "nice slim touch screen"	**GPS→Screen:** *"smaller screen but dont"*, "wide screen", *"nothing but screen", "ok but screen", "screen but normal", "screen size"*, "large screen", "better traffic features cons touch screen", *"screen real estate than", "than years screen"*

Table 2: Example aspect specific opinion phrases (comma delimited in order) discovered by PSM-GPU, sMC-GPU, LDA-P-GPU. Errors are italicized and marked in red.

LDA with phrases (LDA-P): As aspect-sentiment phrases are often noun phrases, a basic approach is to include the noun phrases (extracted using a parser) as separate terms in the corpus.

Topical N-gram (TNG): The TNG model in (Wang et al., 2007) extends LDA to model n-grams of arbitrary length. As aspects often appear close to their opinion in the sentence, topical n-grams for each aspect form a natural baseline. We used the authors original implementation in the MALLET toolbox.

LDA-P with GPU (LDA-P-GPU): This model is due to (Fei et al., 2014) and is tailored for phrase extraction in opinion mining. It employs LDA with noun phrases in the GPU framework to rank the aspect phrases higher in their topics. Our implementations of LDA-P and LDA-P-GPU use the noun phrases discovered by the Stanford Parser.

semi-Markov CRF with GPU(sMC-GPU): This model builds over the model of (Yang and Cardie, 2012) that used dependency tree features and semi-CRF to model the arbitrarily long expressions. We used these expression spans as multiword in vocabulary. Then we employ GPU based sampling with LDA proposed by (Fei et al., 2014) to collocate opinion expressions.

5.3 Qualitative Analysis

To assess the quality of extracted expressions, we labeled the topics following instructions in (Mimno et al., 2011). First, each topic was labeled as coherent or incoherent and an aspect name was given if the topic was coherent. Each topic was presented as a list of top 45 terms in descending order of their probabilities under that topic. A topic was considered coherent if the terms in the topic were semantically related to each other.

Next, for coherent topics, their terms were labeled as correct (if the terms semantics was relevant to the topic) or incorrect (otherwise). Two human judges were used in the annotation. Agreements being high ($\kappa > 0.78$), disagreements were resolved upon consensus among judges.

Table 2 reports the top 10 terms(words/phrases) for aspect 'connection' (Router domain) and aspect 'screen'(GPS domain) across PSM-GPU, sMC-GPU and LDA-P-GPU (the two closest competitor). We note that PSM-GPUs phrases are more expressive compared to sMC-GPU because sMC-GPU is prone to have longer phrases due to segment features but PSM's switch variable captured more relevant aspect specific opinion expressions. sMC-GPU has better

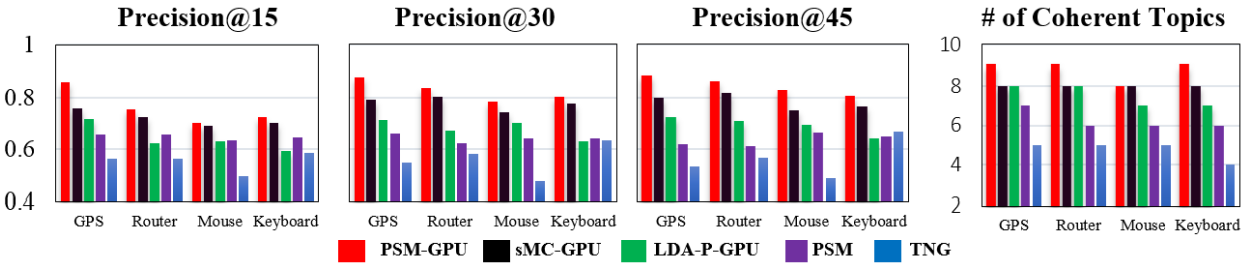


Figure 2: Charts from left to right are Topical words Precision@15, Precision@30, Precision@45 of coherent topics of each model and last one is number of coherent topics of each model.

Domain	PSM-GPU				PSM				sMC-GPU				LDA-P-GPU				TNG				LDA-P			
	P	R	F	Ac.	P	R	F	Ac.	P	R	F	Ac.	P	R	F	Ac.	P	R	F	Ac.	P	R	F	Ac.
Router	71.8	67.8	69.7	68.8	69.0	67.3	68.1	67.7	69.6	67.7	68.6	68.2	65.2	67.9	66.5	67.1	65.2	66.8	66.0	65.9	65.5	68.0	66.7	67.4
GPS	69.3	69.2	69.2	69.3	66.7	67.7	67.2	67.4	65.3	68.0	65.7	65.5	63.6	68.4	65.9	66.8	64.1	68.3	66.1	58.8	66.0	69.7	67.8	68.6
Mouse	87.4	81.4	84.0	87.3	82.9	80.4	81.3	82.4	83.4	81.5	82.6	82.3	81.3	78.3	79.4	80.9	82.9	77.9	80.0	82.5	77.9	77.3	77.1	77.7
Keyboard	92.5	72.9	81.2	84.1	90.2	70.1	78.4	81.6	88.4	70.3	77.6	81.0	86.7	69.3	76.4	79.9	88.8	66.6	75.8	79.1	83.6	63.4	71.4	75.0
Avg.	80.3	72.8	**76.0**	77.4	77.2	71.4	**73.8**	74.8	76.7	71.9	**73.6**	74.3	74.2	71.0	**72.1**	73.7	75.3	69.9	**72.0**	71.6	73.3	69.6	**70.8**	72.2

Table 4: Sentiment classification: Precision, Recall, F1 and accuracy from top to down for each domain and each model

phrases compared to LDA-P-GPU because the latter only considers noun phrases which may not always be semantically coherent under an aspect. The qualitative results of other baselines TNG and LDA-P were worse than that of LDA-P-GPU and hence omitted due to space constraints. However, the subsequent experiments compare all models quantitatively.

5.4 Quantitative Analysis

We consider the following metrics and tasks:

Average Precision: Figure 2 shows the average Precision@n (p@n) for $n = 15, 30, 45$ of all coherent topics for each model in each domain. We note that PSM-GPU achieves the highest precision for all domains significantly ($p < 0.01$) outperforming its closest competitor sMC-GPU. sMC-GPU tends to discover longer phrases due to segment features in semi-CRF and combined with GPU gains the maximum strength among other baselines. Next in order are LDA-P-GPU, PSM, and TNG. We have not shown the result of LDA-P as its top terms didn't contain enough phrases and its precision scores were quite lower compared to other models. But it is worthwhile to note that PSM outperforms LDA-P-GPU (2nd best competitor) at lower ranks which is more important (e.g., in majority domains for $p@15$) and shows its effectiveness. It is a bit unfair to compare PSM with sMC-GPU because PSM is lacking phrase rank optimization whereas sMC-GPU enforces it, and the $p@n$ metric uses rank position as its goodness criterion. However, we will see that in an actual application task, both PSM and PSM-GPU does better than sMC-GPU. Also, we observed that p@45 is higher compare to p@15 or p@30. The reason is even though we are promoting the phrases using GPU it is not able to remove some aspect opinion words from top 15 terms due to their high occurrence in phrases. For e.g. Table 2 has opinion words like "updating", "turn" which are considered incorrect because of non-phrasal terms.

# of coherent topics: Figure 2 (rightmost chart) shows the number of coherent topics produced by each model. A model that can discover more coherent topics is better. We find that PSM-GPU can discover more coherent topics with phrases than its baselines across all domains. The trends of other models are similar to $p@n$ and can be analogously explained.

We note that the Topic Coherence (TC) metric in (Mimno et al., 2011) which is often used to approximate coherence in unigram topic models as it correlates with human notions of coherence, uses co-document frequency of individual words in topics.

However, in our problem as phrases are sparse, their co-document frequency is far lower than words. Hence, the TC metric is not directly applicable. Our measure of coherence is based on human judgment (achieves high agreements, $\kappa > 0.78$ see Section 5.3) and from Table 2 we can see the discovered phrases do reflect coherence. Hence, to evaluate the phrases quantitatively, we employ an actual sentiment classification task that uses the posterior of our models (top phrases) as features. This is reasonable because the estimated topics (when used as features) improve sentiment classification, it shows that they are meaningful and capable of capturing latent sentiment that govern polarities.

Sentiment Classification: For this task, instead of using all the words as features, we used the posterior on φ^A (top 50 terms of φ^A) as features. For all models, all possible n-grams of top 50 terms are also considered as features. We trained SVMs[3] (using the SVMLight toolkit) with the features described above. Evaluation for this task employed 5-fold cross validation on the data in Table 3 (col 4, 5). For each test fold, the features were induced upon fitting the aspect extraction models on the training data of that fold. From Table 4 we note that both PSM-GPU and PSM outperform all competitors on average F1 across all domains. More specifically, we note that PSM alone that uses no rank optimization performs better than sMC-GPU employing phrasal rank optimization under GPU scheme. We believe this is due to PSM's switching component that can discover correct aspect/sentiment terms (sufficient for polarity classification) and rank it higher based on frequency even though the expressive aspect specific phrases remain ranked lower. sMC-GPU tends to have longer phrases so it does well, however, under GPU, longer phrases may not be promoted well as they lack anchor aspect terms under a relevant topic. LDA-P-GPU uses standard (Noun Phrases) NPs for phrases with rank optimization and hence is the next in performance order as NPs may not capture opinion well. TNG does not perform as well as it relies on multiword collocation as opposed to NP/VP for phrase extraction. LDA-P's performance is lowest as it cannot rank the relevant NPs high. PSM-GPU has the right balance of phrase boundary span and phrasal rank optimization via GPU that makes it significantly outperform ($p < 0.01$) all competitors.

[3]Using an RBF kernel ($C = 10, g = 0.01$) which performed best upon tuning various SVM parameters via cross validation.

Domain	Precision	Recall	F1	Acc.
Router	69.68	67.3	68.4	67.9
GPS	66.47	68.6	67.5	68.00
Mouse	87.24	80.5	83.4	86.9
Keyboard	85.45	70.9	77.1	81.1

Table 5: Domain ablation result on polarity classification

Sequence Model Sensitivity: To assess the robustness of the hybrid framework, we evaluate the sensitivity of the embedded CRF model via domain ablation. We choose the best performer PSM-GPU and ablate each domain during its CRF training. We repeat the previous experiment on sentiment classification using the ablated model. From the results in Table 5, we note that the reduction in precision is relatively more than that of recall. However, the F1 score does not drop significantly (compared to Table 4) for any domain showing the robustness of the hybrid framework. We note that even with some skewness in the labeled data (Table 3), CRF is not overfitting here and the proposed pivot features (Table 1) are powerful enough to learn the phrasal structure across domain.

6 Conclusion

This paper presented a novel hybrid framework for aspect specific opinion expression extraction. Two models PSM and PSM-GPU were proposed that employ CRF discriminative sequence modeling for phrase boundary extraction and generative modeling for grouping relevant terms under a topic. PSM-GPU further optimized the aspect coherence using the generalized Pólya urn sampling scheme. Experimental results showed that the proposed hybrid models can extract more coherent aspect specific opinion expressions significantly outperforming all competitors across all domains and are robust in cross-domain knowledge transfer.

Acknowledgment

This work was supported in part by NSF 1527364.

References

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *IJCNLP*, pages 1162–1170. Citeseer.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *IJCAI*, volume 7, pages 2683–2688.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *EMNLP*, pages 1655–1667.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 355–362. Association for Computational Linguistics.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439. Association for Computational Linguistics.

Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. 2014. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment*, 8(3):305–316.

Geli Fei, Zhiyuan Chen, and Bing Liu. 2014. Review topic discovery with phrases using the pólya urn model. In *COLING*, pages 667–676.

Geli Fei, Zhiyuan Brett Chen, Arjun Mukherjee, and Bing Liu. 2016. Discovering correspondence of sentiment words and aspects. In *In proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1035–1045. Association for Computational Linguistics.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Richard Johansson and Alessandro Moschitti. 2011. Extracting opinion expressions and their polarities: exploration of pipelines and joint models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 101–106. Association for Computational Linguistics.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8. Association for Computational Linguistics.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL*, volume 7, pages 1065–1074. Citeseer.

T Kudo. 2009. Crf++: Yet another crf toolkit [ol].

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Huayi Li, Arjun Mukherjee, Jianfeng Si, and Bing Liu. 2015. Extracting verb expressions implying negative opinions. In *AAAI*, pages 2411–2417.

Robert V Lindsey, William P Headden III, and Michael J Stipicevic. 2012. A phrase-discovering topic model using hierarchical pitman-yor processes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 214–222. Association for Computational Linguistics.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1729–1744. ACM.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 339–348. Association for Computational Linguistics.

Arjun Mukherjee. 2016. Extracting aspect specific sentiment expres-sions implying negative opinions. In *In proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics*.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 486–495.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2011. Content models with attitude. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 350–358. Association for Computational Linguistics.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 697–702. IEEE.

Shuai Wang, Zhiyuan Chen, and Bing Liu. 2016. Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the 25th International Conference on World Wide Web*, pages 167–176. International World Wide Web Conferences Steering Committee.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1533–1541. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.

Bishan Yang and Claire Cardie. 2014. Joint modeling of opinion expression extraction and attribute classification. *Transactions of the Association for Computational Linguistics*, 2:505–516.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65. Association for Computational Linguistics.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 379–388. Association for Computational Linguistics.

Emotion Distribution Learning from Texts

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, Xin Geng*
MOE Key Laboratory of Computer Network and Information Integration
School of Computer Science and Engineering
Southeast University, Nanjing, China
{d.zhou, zhouying1, xuanzhang, zhaoquan, xgeng}@seu.edu.cn

Abstract

The advent of social media and its prosperity enable users to share their opinions and views. Understanding users' emotional states might provide the potential to create new business opportunities. Automatically identifying users' emotional states from their texts and classifying emotions into finite categories such as *joy*, *anger*, *disgust*, etc., can be considered as a text classification problem. However, it introduces a challenging learning scenario where multiple emotions with different intensities are often found in a single sentence. Moreover, some emotions co-occur more often while other emotions rarely coexist. In this paper, we propose a novel approach based on emotion distribution learning in order to address the aforementioned issues. The key idea is to learn a mapping function from sentences to their emotion distributions describing multiple emotions and their respective intensities. Moreover, the relations of emotions are captured based on the Plutchik's wheel of emotions and are subsequently incorporated into the learning algorithm in order to improve the accuracy of emotion detection. Experimental results show that the proposed approach can effectively deal with the emotion distribution detection problem and perform remarkably better than both the state-of-the-art emotion detection method and multi-label learning methods.

1 Introduction

The advent of social media and its prosperity enable the creation of massive online user-generated content including opinions and product reviews. Analyzing such user-generated content allows the detection of users' emotional states, which might be potentially useful for downstream applications such as brand watching, product recommendation, and detection of health-related issues, etc. Based on the way emotions are represented, computational models for emotion analysis can be categorized into dimensional models and categorical models (Calvo and D'Mello, 2010). Dimensional approaches (Russell, 2003) emphasize the fundamental dimensions of valence and arousal in understanding emotional experience, which have long been studied by emotion theorists. Categorical models (Gupta et al., 2013) involve the use of a categorical representation, in which emotions are represented by a number of labels. For example, Ekman's basic emotion set (Ekman, 1992) consists of anger, disgust, fear, happiness, sadness and surprise. An example of a sentence and the annotated emotions can be found in Table 1.

Sentence	Trains crash near Thai resort town					
Emotions	anger	disgust	fear	joy	sadness	surprise
	2	0	62	0	90	10

Table 1: An example of a sentence containing emotions selected from SemEval 2007 Task#14, Affective Text, where each of the six emotions are indicated using a score of [0, 100].

Considering each basic emotion as class label for the sentence, emotion detection can be treated as a classification problem. There is a large body of prior work on emotion classification (Mishne and de Rijke, 2006; Lin and He, 2009; Quan et al., 2015; Wang and Pal, 2015). By choosing the strongest emotion as the emotion label for the sentence, most of

*Corresponding author

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 638–647,
Austin, Texas, November 1-5, 2016.

classification approaches are based on single-label learning. However, as shown in Table 1, a sentence might contain multiple emotions with varying intensities. Although, some lexicon-based approach such as (Wang and Pal, 2015) can output multiple emotions with intensities using non-negative matrix factorization. It can only guarantee convergence to a local minimum, which is prohibitive on the large, realistically-sized emotion detection problem.

Machine learning methods such as multi-label learning (MLL) can be employed to identify multiple emotions for each sentence (Zhang and Zhou, 2014). MLL usually selects a threshold, then labels emotions with scores higher than the threshold as relevant and the others as irrelevant. However, these methods are not able to learn the intensity of each emotion. To address this problem, a new machine learning paradigm called Label Distribution Learning (LDL) (Geng, 2016) was proposed in recently years. Similarly, in this paper, we propose an emotion distribution learning (EDL) algorithm. Different from the previous approaches, EDL assumes that each sentence contains a mixture of basic emotions with different intensities. Using categorical model, we can label each sentence with an emotion vector where each element corresponds to one basic emotion and the value of each element indicates the intensity of the emotion. We require that each vector element has a value between 0 and 1 and they sum up to 1. By doing so, the emotion vectors can be considered as emotion distributions and the proposed EDL algorithm aims to learn the mapping from sentences to their corresponding emotion distributions by minimizing the differences between the true distributions and the predicted distributions. Both the single-label learning and MLL can be considered as special cases of EDL in emotion detection. Moreover, as some emotions co-occur more often while others rarely co-exist, the relations between basic emotions are captured according to the Plutchik's wheel of emotions theory (Plutchik, 1980) and are incorporated in the learning framework as constraints in order to improve the accuracy of emotion detection.

Our work makes the following contributions:

- We propose a novel approach based on emotion distribution learning to identify multiple emotions with their intensities from texts. To the best of our knowledge, it is the first attempt to identify both emotions and intensities in the distribution learning framework.
- The relations between basic emotions are incorporated into the learning framework as constraints to improve the emotion detection accuracy. To avoid the incorporation of noisy information from the training data, the relation constraint is set based on the Plutchik's wheel of emotions theory.
- Experimental results show that the proposed approach can effectively deal with the emotion distribution detection problem and perform remarkably better than the state-of-the-art multi-label learning methods and emotion detection method.

2 Related Work

In general, emotion classification can be approached by two types of methods, lexicon-based or corpus-based. Lexicon-based approaches rely on emotion lexicons consisting of words and their corresponding emotion labels for detecting emotions from text. For example, WordNetAffect (Strapparava and Valitutti, 2004) was constructed by extending Wordnet, a lexical database of English terms, with information on affective terms. EmoSenticNet assigns six WordNetAffect emotion labels to SenticNet concepts (Poria et al., 2013), which can be thought of as an expansion of WordNetAffect emotion labels to a larger vocabulary. Many approaches were proposed based on emotion lexicons. For example, (Aman and Szpakowicz, 2007) classified emotional and non-emotional sentences using the constructed emotion lexicon. (Choudhury et al., 2012) employed a classifier to detect human affective states in social media. (Wang and Pal, 2015) proposed a model with several constraints based on an emotion lexicon for emotion classification.

Corpus-based methods aim to train supervised classifiers from annotated training data where each sentence or document is labelled with an emotion class. (Mishne and de Rijke, 2006) constructed models to predict the levels of various moods according to the language used by bloggers at a giv-

en time. (Aman and Szpakowicz, 2007) described an emotion annotation task of identifying emotion category, emotion intensity and the words/phrases that indicate emotions in text. Emotion classification was conducted using trained support vector machines. (Agrawal and An, 2012) proposed an unsupervised context-based approach to detect emotions from text at the sentence level. They computed an emotion vector for each potential affect bearing word based on the semantic relatedness between words and various emotion concepts. The scores are then tuned using the syntactic dependencies within the sentence structure. (Bao et al., 2009) proposed an emotion topic model by augmenting latent Dirichlet allocation with an intermediate emotion layer. (Quan et al., 2015) proposed a logistic regression model for social emotion detection. Intermediate hidden variables were also introduced to model the latent structure of input text corpora.

Our work is partly inspired by (Quan et al., 2015). However, our proposed approach differs from (Quan et al., 2015) in two aspects: 1) by introducing the emotion distribution learning framework, many different criteria can be used to measure the distance between the true distribution and the predicted distribution, such as squared $\mathcal{X}^2$, Euclidean, Jeffery's divergence apart from Kullback-Leibler divergence employed in logistic regression model. 2) the relations between basic emotions are captured based on the Plutchik's wheel of emotions theory to avoid the incorporation of any noisy information from the training data.

3 Emotion Distribution Learning

3.1 Problem Setting

As have discussed in section 1, one sentence might contain one or more emotions, and each emotion has its own intensity. We use d_x^y to indicate the intensity of emotion y for sentence x, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The emotion intensity is normalized to make $d_x^y \in [0,1]$ and $\sum_y d_x^y = 1$ to constitute the emotion distribution.

Note that d_x^y denotes the proportion that y accounts for in a full emotion distribution of x. It is different from the probability of y being a correct emotion label for x. Probability distribution implies that only one emotion label is correct for each sentence, while emotion distribution allows multiple emotions in one sentence. The goal of EDL is to learn a mapping from sentences $\mathcal{X} = \mathbb{R}^m$ to the distributions over a finite set of labels $\mathcal{Y} = \{y_1, y_2, ...y_c\}$. Each label represents one of the basic emotions.

3.2 Learning

Given a training set $P = \{(x_1, E_1), (x_2, E_2), ..., (x_n, E_n)\}$, where $x_i \in \mathcal{X}$ is a sentence and $E_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, ..., d_{x_i}^{y_c}\}$ is the emotion distribution associated with x_i. The goal of EDL is to learn a conditional probability mass function $p(y|x)$ from P, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assuming that $p(y|x)$ is a parametric model $p(y|x;\theta)$, where θ are model parameters, many different criteria can be used to measure the distance between two distributions, such as Squared $\mathcal{X}^2$, Euclidean, Jeffery's divergence, Kullback-Leibler (K-L) divergence and so on. Here we use Divergence defined by

$$D_J(Q_a||Q_b) = 2\sum_j \frac{(Q_a^j - Q_b^j)^2}{(Q_a^j + Q_b^j)^2}$$

, where Q_a^j and Q_b^j are the j-th element of the two distributions Q_a and Q_b, respectively. Divergence is balanced, which makes $D_J(Q_a||Q_b)$ equal to $D_J(Q_b||Q_a)$. The formula above calculates the sum of all the distances between emotion intensities in the same position.

Then the optimal model parameters θ^* is determined by

$$\begin{aligned}
\theta^* &= \arg\min_\theta \left\{ \sum_i D_J(E_i||\hat{E}_i) + \frac{\xi_1}{n}\sum_{k,r}|\theta_{k,r}|_1 \right. \\
&\left. + \frac{\xi_2}{n}\sum_u\sum_{j,k}\omega_{jk}\|\theta_{u,j} - \theta_{u,k}\|_2^2 \right\} \\
&= \arg\min_\theta \left\{ 2\sum_{i,j}\frac{(d_{x_i}^{y_j} - p(y_j|x_i,\theta))^2}{(d_{x_i}^{y_j} + p(y_j|x_i,\theta))^2} \right. \\
&+ \frac{\xi_1}{n}\sum_{k,r}|\theta_{k,r}|_1 \\
&\left. + \frac{\xi_2}{n}\sum_u\sum_{j,k}\omega_{jk}\|\theta_{u,j} - \theta_{u,k}\|_2^2 \right\}
\end{aligned} \tag{1}$$

, where E_i is the ground truth emotion distribution of the i-th sentence and the $\hat{E}_i$ is the predicted one by $p(y|x_i;\theta)$. The second term is a regularizer to make the predicted emotion distribution sparse, and the third term considers the relationship between different emotions. As mentioned in section 1, some emotions often co-occur such as joy and love, and some rarely co-exist such as joy and anger. Therefore, the third term is employed to incorporate such prior knowledge. The weight ω_{jk} models the relationship between the *j*-th emotion and the *k*-th emotion in the distribution. In this paper, we capture the relationships between different emotions based on Plutchik's wheel of emotions (Plutchik, 1980) which is produced in psychology view. Plutchik's wheel of emotions includes several typical emotions and its eight sectors indicate eight primary emotion dimensions arranged as four pairs of opposites. We re-produce a wheel of eight emotions' relationships according to Plutchik's theory, which is shown in Figure 1.

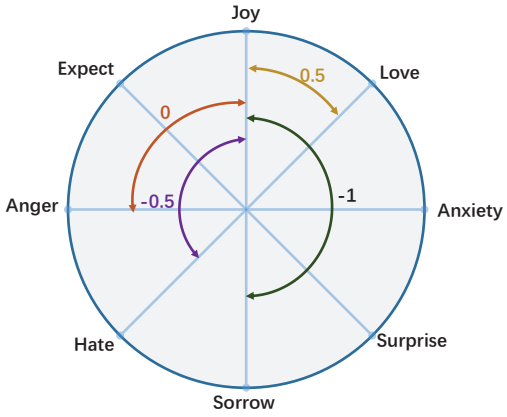


Figure 1: Plutchik's wheel of emotions.

In the emotion wheel, emotions sat at opposite end have an opposite relationship, while emotions next to each other are more closely related. We quantify the relations between each pair of emotions based on the angle between them in wheel of emotions (Plutchik, 2001). For example, emotion pairs with 180 degrees are opposite to each other, which are described by -1, while emotion pairs with 90 degrees are described by 0, meaning no relationship between them. Emotion pairs with 45 degrees have the relationship value of 0.5, while emotion pairs with 135 degrees have the relationship value of -0.5. Figure 2 shows the gray-scale image of the pair-wise relationships of emotions presented in Figure 1. In each cell, the darker the color is, the more similar the two emotions are.

As for $p(y|x;\theta)$, similar to (Geng, 2016), we assume it takes a maximum entropy model, i.e.,

$$p(y_k|x_i;\theta) = \frac{1}{\mathcal{Z}_i}\exp(\sum_r \theta_{kr}x_i^r) \tag{2}$$

, where $\mathcal{Z}_i = \sum_k \exp(\sum_r \theta_{kr}x_i^r)$ is the normalization factor, x_i^r is the *r*-th feature of x_i, and θ_{kr} is an element in θ. Substituting Equation 2 into Equation 1 yields the target function,

$$\begin{aligned} T(\theta) = 2\sum_{i,j}\left(1 - \frac{4\mathcal{Z}_i d_{x_i}^{y_j}\exp\left(\sum_r \theta_{jr}x_i^r\right)}{\left(\mathcal{Z}_i d_{x_i}^{y_j} + \exp\left(\sum_r \theta_{jr}x_i^r\right)\right)^2}\right) \\ + \frac{\xi_1}{n}\sum_{k,r}|\theta_{k,r}|_1 \\ + \frac{\xi_2}{n}\sum_u\sum_{j,k}\omega_{jk}\|\theta_{u,j} - \theta_{u,k}\|_2^2. \end{aligned} \tag{3}$$

The minimization of the function $T(\theta)$ can be effectively solved by the limited-memory quasi-Newton method (L-BFGS). The basic idea of L-BFGS is to avoid explicit calculation of the inverse Hessian matrix used in the Newton method. L-BFGS approximates the inverse Hessian matrix with an iteratively updated matrix instead of actually storing the full matrix. Here we follow the idea of an effective quasi-Newton method BFGS. Consider the second-order Taylor series of $T'(\theta) = -T(\theta)$ at the current estimate of the parameter vector $\theta^{(l)}$:

$$\begin{aligned} T'(\theta^{(l+1)}) \approx T'(\theta^{(l)}) + \nabla T'(\theta^{(l+1)})^T\Delta \\ + \frac{1}{2}\Delta^T H(\theta^{(l)})\Delta, \end{aligned} \tag{4}$$

where $\Delta = \theta^{(l+1)} - \theta^{(l)}$ is the update step, $\nabla T'(\theta^{(l)})$ and $h(\theta^{(l)})$ are the gradient and Hessian matrix of $T'(\theta^{(l)})$ at $\theta^{(l)}$, respectively. The minimizer of Equation 4 is

$$\Delta^l = -H^{-1}(\theta^{(l)})\nabla T'(\theta^{(l)}). \tag{5}$$

The line search Newton method uses $\Delta^{(l)}$ as the search direction $p^{(l)} = \Delta^{(l)}$ and updates model parameters by

$$\theta^{(l+1)} = \theta^{(l)} + \alpha^{(l)}p^{(l)}, \tag{6}$$

where the step length $\alpha^{(l)}$ is obtained from a line search procedure to satisfy the strong Wolfe conditions (Nocedal and Wright, 2006):

$$T'(\theta^{(l)}+\alpha^{(l)}p^{(l)}) \leq T'(\theta^{(l)})+c_1\alpha^{(l)}\nabla T'(\theta^{(l)})^T p^{(l)}$$
$$|\nabla T'(\theta^{(l)}+\alpha^{(l)}p^{(l)}| \leq c_2|\nabla T'(\theta^{(l)})^T p^{(l)}|,$$

where $0 < c_1 < c_2 < 1$. The idea of BFGS is to avoid explicit calculation of $H^{-1}(\theta^{(l)})$ by approximating it with an iteratively updated matrix B, i.e.

$$\begin{aligned} B^{(L+1)} &= (I-\rho^{(l)}s^{(l)}(u^{(l)})^T)\times B^{(l)} \\ &\quad \times(I-\rho^{(l)}u^{(l)}(s^{(l)})^T) \\ &\quad +\rho^{(l)}s^{(l)}(s^{(l)})^T \end{aligned}$$

where

$$\begin{aligned} s^{(l)} &= \theta^{(l+1)}-\theta^{(l)}, \\ u^{(l)} &= \nabla T'(\theta^{(l+1)})-\nabla T'(\theta^{(l)}), \\ \rho^{(l)} &= \frac{1}{s^{(l)}u^{(l)}}. \end{aligned}$$

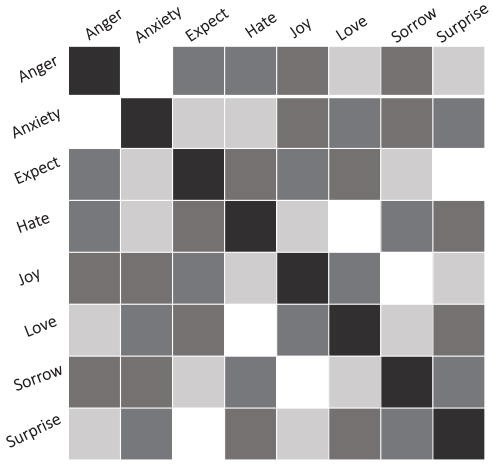


Figure 2: Gray-scale image of the pair-wise relationships of emotions shown in Figure 1.

As for the optimization of the target function $T(\theta)$, the computation of BFGS is mainly related to the first-order gradient of $T'(\theta)$, which can be achieved by

$$\begin{aligned} \frac{\partial T(\theta)}{\partial \theta_{jr}} &= \frac{4d_{x_i}^{y_j}p_{ij}(1-p_{ij})(d_{x_i}^{y_j}-p_{ij})}{(d_{x_i}^{y_j}+p_{ij})^3} \\ &\quad +\xi_1\sum_{k,r} sgn(\theta_{k,r}) \\ &\quad +\frac{1}{n}\xi_2\sum_k \omega_{jk}(\theta_j-2\theta_k), \end{aligned} \quad (7)$$

where $p_{ij} = \frac{1}{Z_i}\exp(\sum_r \theta_{jr}x_i^r)$. Thus it performs more efficiently than the standard line search Newton method.

In order to compare with the MLL methods, labels in the predicted distribution need to be divided into two sets, i.e, the relevant and irrelevant sets. For this purpose, an extra virtual label y_0 is added into the label set, i.e., the extended label set $\mathcal{Y}' = \mathcal{Y} \cup \{y_0\}$=$\{y_0, y_1, y_2 ... y_c\}$. Using the new extended label set in the training process, the optimal parameter vector θ^* is learned. As y_0 is the label that distinguishes the relevant and irrelevant emotions directly, it is initialized as the threshold used in MLL. Given a sentence x', its emotion distribution is predicted by $p(y|x';\theta^*)$. The intensity value of y_0 splits the predicted distribution into two sets. The emotions with the intensity value higher than y_0's are regarded as the relevant emotions, and the rest emotions are regarded as irrelevant ones. Therefore, EDL in fact implements the function of MLL without the need of setting the threshold manually.

4 Experiments

4.1 Setup

We evaluate the proposed approach on the RenCECps corpus (Quan and Ren, 2010). It contains $35,096$ sentences selected from blogs in Chinese. Each sentence is annotated with 8 basic emotions, such as *anger*, *anxiety*, *expect*, *hate*, *joy*, *love*, *sorrow* and *surprise*, together with their emotion scores. Higher score represents higher emotion intensity. We use $AS_i(j)$ to represent the score of emotion j in sentence i. Given a sentence x_i, the intensity of emotion j is calculated by $d_{x_i}^{y_j} = \frac{AS_i(j)}{\sum_k AS_i(k)}$. By doing so, each intensity value fulfills $d_{x_i}^{y_j} \in [0,1]$ and $\sum_y d_{x_i}^{y_j} = 1$.

For each sentence, features are extracted using recursive auto-encoders (RAEs) (Socher et al., 2011). RAEs are neural networks that represent meanings of fixed-size inputs in the reduced dimensional space. For example, each word in a sentence is represented using a vector $w \in \mathbb{R}^d$, and the RAE method reduces the entire sentence to a single vector of size $\mathbb{R}^d$. Sentences are sequences of words that can be represented by a binary tree structure. The words are the leaves of the tree and their combined grouping is used to get a notion of the meaning of the sentence.

The internal nodes of the tree correspond to the combined meaning of the nodes underneath them. Each internal node is also represented in the same manner as individual words in the form of a vector $\hat{w} \in \mathbb{R}^d$. These internal nodes are the hidden representations of the neural network. In the RAE model, the vocabulary is stored in an embedding matrix $V \in \mathbb{R}^d \times D$ where D is the cardinality of the vocabulary. Typically, each word $w \in V$ is initialized independently following a Gaussian distribution $w_i \sim N(0, \gamma^2)$. In our experiment, we set the dimension of each sentence representation to 100.

We build a gray-scale image shown in Figure 3 by computing the correlation coefficient of the emotions from the Ren-CECps corpus. It can be observed that Figure 3 is quite similar to Figure 2, which shows that our proposed way in capturing the relations between emotions is inline with what have been revealed by the emotion annotations in the Ren-CECps corpus.

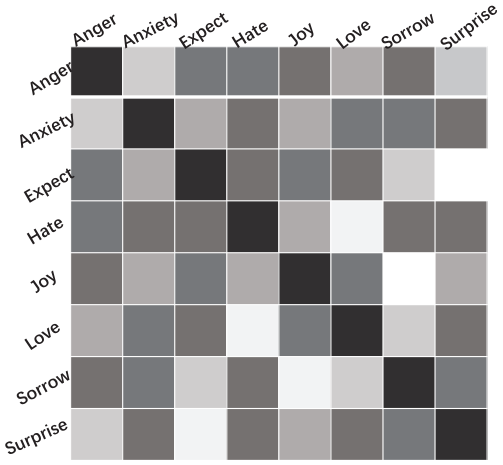


Figure 3: Gray-scale image of the pair-wise relations of the emotions in the Ren-CECps corpus.

4.2 Experimental Results

As the output of EDL is a distribution, a natural choice of criteria is the averaged similarity or distance between the actual emotion distribution and the predicted distribution. There are many metrics that can be applied to measure the distance between two distributions. In this paper six of them are used to evaluate the results of EDL, i.e, Euclidean, Sϕrensen, Squared $\mathcal{X}^2$, KL divergence, Intersection and Fidelity, as suggested in (Geng and Ji, 2013).

	Name	Formula
Distance	Euclidean	$Euclidean(P,Q) = \sqrt{\sum_{j=1}^{c}(P_j - Q_j)^2}$
	Sϕrensen	$S\phi rensen(P,Q) = \frac{\sum_{j=1}^{c}\lvert P_j - Q_j\rvert}{\sum_{j=1}^{c}(P_j + Q_j)}$
	Squared $\mathcal{X}^2$	$Squared\mathcal{X}^2(P,Q) = \sum_{j=1}^{c}\frac{(P_j - Q_j)^2}{P_j + Q_j}$
	Kullback-Leibler (KL)	$K\text{-}L(P,Q) = \sum_{j=1}^{c} P_j \ln \frac{P_j}{Q_j}$
Similarity	Intersection	$Intersection(P,Q) = \sum_{j=1}^{c}\min(P_j, Q_j)$
	Fidelity	$Fidelity(P,Q) = \sum_{j=1}^{c}\sqrt{P_j Q_j}$

Table 2: Evaluation criteria for the Label Distribution Learning (LDL) methods.

Name	Formula
Hamming Loss	$hloss(h) = \frac{1}{P}\sum_{i=1}^{P}\lvert h(x_i) \triangle Y_i\rvert$
One error	$one\text{-}error(f) = \frac{1}{P}\sum_{i=1}^{P}[\arg\max_{y \in Y} f(x_i, y)] \notin Y_i$
Coverage	$Coverage(f) = \frac{1}{P}\sum_{i=1}^{P}\max_{y \in Y_i} rank_f(x_i, y) - 1$
Ranking Loss	$rloss(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{\lvert Y_i\rvert\lvert\bar{Y}_i\rvert} \cdot \lvert R\rvert, Where$ $R = (y', y'')\lvert f(x_i, y') \leq f(x_i, y''), (y', y'') \in Y_i \times \bar{Y}_i$
Average Precision	$Average(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{\lvert Y_i\rvert}\sum_{y \in Y_i}\frac{\lvert P_i\rvert}{rank_f(x_i, y)}, where$ $P_i = y'\lvert rank_f(x_i, y') \leq rank_f(x_i, y), (y)' \in Y_i$

Table 3: Evaluation criteria for the MLL methods.

The formulae of the six criteria are summarized in Table 4.2. Note that the virtual label y_0 is removed before evaluation.

As EDL can output both the relevant emotions and their respective emotion intensities, MLL can be seen as a special case of EDL that it only outputs emotion labels but not their intensities. Several evaluation criteria typically used in MLL can also be used to measure EDL's ability of distinguishing relevant emotions from irrelevant ones, including hamming loss, one error, coverage, ranking loss, and average precision as suggested by (Zhang and Zhou, 2014), which are summarized in Table 4.2. Hamming loss evaluates how many times an emotion label is misclassified. One-error evaluates the fraction of sentences whose top-ranked emotion is not in the relevant emotion set. Coverage evaluates how many steps are needed to move down the ranked emotion list so as to cover all the relevant emotions of the example. Ranking loss evaluates the fraction of reversely ordered emotion pairs. Average precision evaluates the average fraction of the relevant emotions ranked higher than a particular emotion $y \in \mathcal{Y}$.

For each algorithm, ten-fold cross validation is conducted. EDL is first compared with four existing Label Distribution Learning (LDL) methods (Geng,

Algorithm	Evaluation Criterion					
	Euclidean(↓)	Sørensen(↓)	Squared $\mathcal{X}^2$(↓)	K-L(↓)	Intersection(↑)	Fidelity(↑)
EDL	**0.2361±0.0057**	**0.2346±0.0061**	**0.1780±0.0037**	**0.2067±0.0046**	**0.7654±0.0046**	**0.9523±0.0019**
AA-KNN (Geng, 2016)	0.2948±0.0101•	0.2941±0.0123•	0.2688±0.0102•	0.3163±0.0087•	0.7059±0.0078•	0.9258±0.0090•
PT-Bayes (Geng, 2016)	0.3295±0.0125•	0.3288±0.0158•	0.2826±0.0115•	0.3263±0.0238•	0.6711±0.0241•	0.9238±0.0060•
PT-SVM (Geng, 2016)	0.3614±0.0869•	0.3625±0.0145•	0.3415±0.0089•	0.4073±0.0209•	0.6375±0.0099•	0.9069±0.0073•
AA-BP (Geng, 2016)	0.3299±0.0159•	0.3430±0.0264•	0.2885±0.0251•	0.3406±0.0092•	0.6569±0.0166•	0.9229±0.0056•
emoDetect (Wang and Pal, 2015)	0.3333±0.0678•	0.3468±0.0719•	0.2928±0.0674•	0.3463±0.0790•	0.6532±0.0719•	0.9212±0.0180•

Table 4: Experimental results in comparison with the LDL methods and the emotion detection approach.

Algorithm	Evaluation Criterion				
	Average Precision(↑)	Coverage(↓)	Hamming Loss(↓)	One Error(↓)	Ranking Loss(↓)
EDL	**0.6419±0.0235**	**2.1412±0.0235**	**0.1772±0.0568**	**0.5239±0.0945**	**0.2513±0.0560**
ML-KNN (Zhang and Zhou, 2014)	0.5917±0.0742•	2.448±0.0981•	0.2459±0.0781•	0.5339±0.0954•	0.2908±0.0431•
LIFT (Zhang, 2011)	0.5979±0.0891•	2.4267±0.0492•	0.1779±0.0597•	0.5131±0.0666•	0.2854±0.0427•
Rank-SVM (Zhang and Zhou, 2014)	0.5738±0.0892•	2.5861±0.0777•	0.2485±0.0458•	0.5603±0.0921•	0.3055±0.0579•
MLLOC (Huang and Zhou, 2012)	0.4135±0.0568•	3.6994±0.0764•	0.1850±0.0659•	0.6971±0.0924•	0.4742±0.0734•
BP-MLL (Zhang and Zhou, 2006)	0.4791±0.0999•	3.3773±0.0681•	0.2108±0.0986•	0.6316±0.0988•	0.4293±0.0956•
ECC (Read et al., 2011)	0.5121±0.0892•	2.7767±0.0876•	0.1812±0.0945•	0.6969±0.0598•	0.3281±0.0659•

Table 5: Experimental results in comparison with the MLL methods.

2016), i.e., PT-Bayes, PT-SVM, AA-KNN, AA-BP. k in AA-KNN is set to 8. Linear kernel is used in PT-SVM. The number of hidden-layer neurons for AA-BP is set to 60. The evaluation results of our proposed approach in comparison to the LDL baselines are presented in Table 4.2. For all the measures, "↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each measure is highlighted by boldface. The two-tailed t-tests with 5% significance level are performed to see whether the differences between EDL and the baselines are statistically significant. We use • to indicate significance difference. As the state-of-the-art emotion detection method proposed in (Wang and Pal, 2015) can output the emotion distributions based on a dimensional reduction method, we present its experimental results on the Ren-CECps corpus in the last row of Table 4.2. It can be observed that EDL performs significantly better than all the baseline LDL methods and the state-of-the-art emotion detection approach on all criteria considered here.

Since EDL can be seen as an extension of MLL, EDL is compared with 7 widely used MLL methods using the virtual label y_0, namely ML-KNN (Zhang and Zhou, 2014), ECC (Read et al., 2011), MLLOC (Huang and Zhou, 2012), LIFT (Zhang, 2011), ML-RBF (Zhang, 2009), Rank-SVM (Zhang and Zhou, 2014), BP-MLL (Zhang and Zhou, 2006). Among the compared algorithms, ML-kNN is derived from the traditional k-nearest neighbor (kNN) algorithm. Maximum a posteriori (MAP) principle is used to determine which emotion set is related to the given sentence. CC (classifier chains method) overcomes the limitations of BR and performs better but requires more computations. ECC (ensemble classifier chains) applies classifier chains in an ensemble framework and obtains high predictive performances. MLLOC (Multi-label LOcal Correlation) tries to exploit emotion correlations in the expression data locally. The global discrimination fitting and local correlation sensitivity are incorporated into a unified framework, and solution for the optimization are developed. Rank-SVM provides a way of controlling the complexity of the overall learning system while having a small empirical error. The architectures of Rank-SVM is based on linear models of Support Vector Machines (SVM) (Boser et al., 1992). LIFT constructs features specific to each emotion by conducting clustering analysis on its positive or negative instances, and then performs training and testing by querying the clustering results (Zhang, 2011). BP-MLL is derived from the famous backpropagation algorithm through employing a novel error function capturing the characteristics of multi-label learning, i.e., the emotions belonging to a sentence should be ranked higher than those not belonging to that sentence (Zhang and

Zhou, 2006).

The virtual label y_0 used in EDL and the threshold value used in MLL are all set to 2.5. Besides, the ε, ξ_1 and ξ_2 are set as 0.25, 0.0001, 0.1 respectively. For the MLL methods, the value of k is set to 8 in ML-KNN, ratio is 0.02 and μ is 2 in ML-RBF. Linear kernel is used in LIFT. Rank-SVM uses the RBF kernel with the width σ equals to 1. The evaluation results of the proposed approach in comparison to all MLL baselines are presented in Table 4.2. EDL performs best on all evaluation measures. It verifies the advantage of EDL owing to the consideration of varying intensity of the basic emotions.

4.3 Further Analysis

To fully understand the emotion detection results, we use word cloud (Harris, 2011) to output the top 30 frequent words in the testing data for the emotion *love* and *anxiety* based on the annotation as shown in the left part of Figure 4. We also output the top 30 frequent words for the two emotions based on the prediction generated by EDL as shown in Figure 4 's right part. It can be observed that most words based on prediction indeed express their associated emotions. For example, word "like" delivers the emotion of *love* (right part of Figure 4(a)) and word "problem" tells *anxiety* (right part of Figure 4(b)). Moreover, the annotation and the prediction share 20 out of the top 30 most frequent words for the emotion *love* such as "friend", "joy", "happiness", etc as shown in the middle of Figure 4(a) and 19 out of 30 for the emotion of *anxiety* (the middle of Figure 4(b). It demonstrates that EDL can learn emotions from text precisely.

To investigate the emotion distributions generated by EDL, a sentence from the Ren-CECps corpus together with the emotion distribution output by EDL is illustrated in Figure 5. The ground truth emotion distribution is obtained by normalizing the scores and the virtual label y_0. As can be seen, the curve of the predicted emotion distribution is very similar as the ground truth distribution, which demonstrates that EDL can learn the varying intensities of all the basic emotions well.

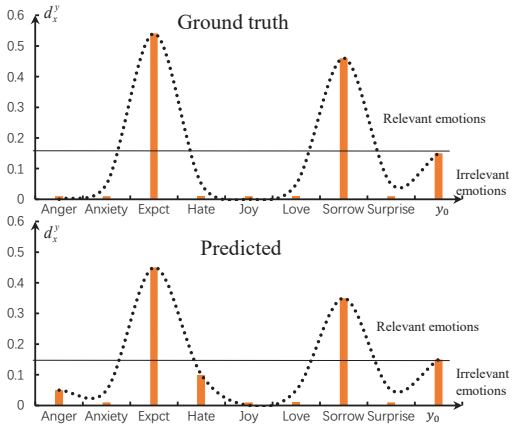


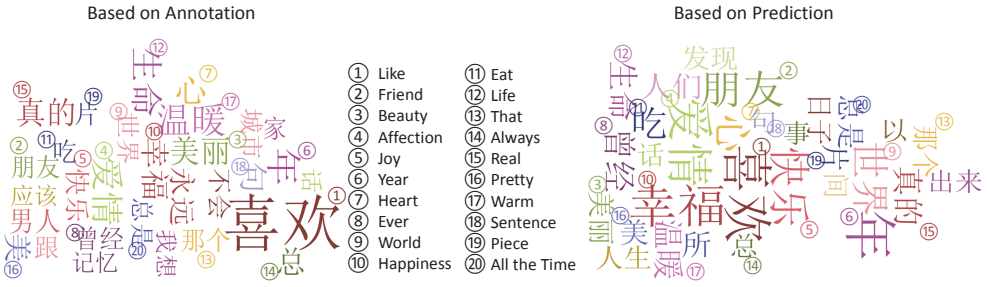
Figure 5: A sentence with the emotion distribution predicted by EDL.

5 Conclusions and Future Work

In this paper, we have proposed a novel approach based on EDL to identify multiple emotions with their intensities from texts. Moreover, the relations between basic emotions is incorporated in the learning framework as constraints to improve the learning accuracy. Experimental results show that the proposed approach can effectively deal with the emotion distribution detection problem and perform remarkably better than the state-of-the-art multi-label learning methods and the emotion detection method. In future work, we will investigate the efficiency of the proposed approach in other datasets and explore other methods in capturing the inter-relations of emotions.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (61273300, 61232007, 61528302, 61622203), the Jiangsu Natural Science Funds for Distinguished Young Scholar (BK20140022), the Natural Science Foundation of Jiangsu Province of China (BK20161430), and the Collaborative Innovation Center of Wireless Communications Technology.



(a) Love

(b) Anxiety

Figure 4: Top 30 frequent words for the emotion *love* and *anxiety* based on the annotation or the prediction.

References

Ameeta Agrawal and Aijun An. 2012. Unsupervised emotion detection from text using semantic and syntactic relations. In *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, pages 346–353.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. *Lecture Notes in Computer Science*, 4629:196–205.

S. Bao, Shengliang Xu, Li Zhang, Rong Yan, Zhong Su, Dingyi Han, and Yong Yu. 2009. Joint emotion-topic modeling for social affective text mining. In *Proceedings of the Ninth IEEE International Conference on Data Mining*, pages 699–704.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

R.A. Calvo and S. D'Mello. 2010. Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on*, 1(1):18–37.

Munmun De Choudhury, Michael Gamon, and Scott Counts. 2012. Happy, nervous or surprised? classification of human affective states in social media. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*, pages 435–438.

Paul Ekman. 1992. An argument for basic emotions. *Cogition and emotion*, 6(3-4):169–200.

Xin Geng and Rongzi Ji. 2013. Label distribution learning. In *Proceedings of the 13th IEEE International Conference on Data Mining Workshops*, pages 377–383.

Xin Geng. 2016. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748.

Narendra Gupta, Mazin Gilbert, and Giuseppe Di Fabbrizio. 2013. Emotion detection in email customer care. *Computational Intelligence*, 29(3):489–505.

Jacob Harris. 2011. Word clouds considered harmful. *Nieman Journalism Lab*.

Sheng-Jun Huang and Zhi-Hua Zhou. 2012. Multi-label learning by exploiting label correlations locally. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 949–955, Toronto, Canada.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 375–384, New York, NY, USA. ACM.

Gilad Mishne and Maarten de Rijke. 2006. Capturing global mood levels using blog posts. In *AAAI Sympo-*

sium on Computational Approaches to Analysing Weblogs (AAAI-CAAW), pages 145–152, June.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Theories of emotion*, 1.

Robert Plutchik. 2001. An argument for basic emotions. *American Scientist*, 89(4):344–350.

S. Poria, A. Gelbukh, A. Hussain, N. Howard, D. Das, and S. Bandyopadhyay. 2013. Enhanced senticnet with affective labels for concept-based opinion mining. *Intelligent Systems, IEEE*, 28(2):31–38.

Changqin Quan and Fuji Ren. 2010. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence*, 2(1):105–117.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Liu Wenyin. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Trans. Inf. Syst.*, 34(1):2:1–2:19.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359.

James A. Russell. 2003. Core affect and the psychological construction of emotion. *Psychological Review*, 110(1):145–172.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet-affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086.

Yichen Wang and Aditya Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 996–1002.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Min-Ling Zhang. 2009. Ml-rbf: Rbf neural networks for multi-label learning. *Neural Processing Letters*, 29(2):61–74.

Min-Ling Zhang. 2011. Lift: Multi-label learning with label-specific features. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1609–1614, Barcelona, Spain.

Building an Evaluation Scale using Item Response Theory

John P. Lalor[1], **Hao Wu**[2], **Hong Yu**[1,3]
[1] University of Massachusetts, MA, USA
[2] Boston College, MA, USA
[3] Bedford VAMC and CHOIR, MA, USA
lalor@cs.umass.edu, hao.wu.5@bc.edu, hong.yu@umassmed.edu

Abstract

Evaluation of NLP methods requires testing against a previously vetted gold-standard test set and reporting standard metrics (accuracy/precision/recall/F1). The current assumption is that all items in a given test set are equal with regards to difficulty and discriminating power. We propose Item Response Theory (IRT) from psychometrics as an alternative means for gold-standard test-set generation and NLP system evaluation. IRT is able to describe characteristics of individual items - their difficulty and discriminating power - and can account for these characteristics in its estimation of human intelligence or ability for an NLP task. In this paper, we demonstrate IRT by generating a gold-standard test set for Recognizing Textual Entailment. By collecting a large number of human responses and fitting our IRT model, we show that our IRT model compares NLP systems with the performance in a human population and is able to provide more insight into system performance than standard evaluation metrics. We show that a high accuracy score does not always imply a high IRT score, which depends on the item characteristics and the response pattern.[1]

1 Introduction

Advances in artificial intelligence have made it possible to compare computer performance directly with human intelligence (Campbell et al., 2002; Ferrucci et al., 2010; Silver et al., 2016). In most cases, a common approach to evaluating the performance of a new system is to compare it against an unseen gold-standard test dataset (GS items). Accuracy, recall, precision and F1 scores are commonly used to evaluate NLP applications. These metrics assume that GS items have equal weight for evaluating performance. However, individual items are different: some may be so hard that most/all NLP systems answer incorrectly; others may be so easy that every NLP system answers correctly. Neither item type provides meaningful information about the performance of an NLP system. Items that are answered incorrectly by some systems and correctly by others are useful for differentiating systems according to their individual characteristics.

In this paper we introduce Item Response Theory (IRT) from psychometrics and demonstrate its application to evaluating NLP systems. IRT is a theory of evaluation for characterizing test items and estimating human ability from their performance on such tests. IRT assumes that individual test questions (referred to as "items" in IRT) have unique characteristics such as difficulty and discriminating power. These characteristics can be identified by fitting a joint model of human ability and item characteristics to human response patterns to the test items. Items that do not fit the model are removed and the remaining items can be considered a scale to evaluate performance. IRT assumes that the probability of a correct answer is associated with both item characteristics and individual ability, and therefore a collection of items of varying characteristics can determine an individual's overall ability.

Our aim is to build an intelligent evaluation metric to measure performance for NLP tasks. With IRT we

[1]Data and code will be made available for download at https://people.cs.umass.edu/lalor/irt.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 648–657,
Austin, Texas, November 1-5, 2016.

can identify an appropriate set of items to measure ability in relation to the overall human population as scored by an IRT model. This process serves two purposes: (i) to identify individual items appropriate for a test set that measures ability on a particular task, and (ii) to use the resulting set of items as an evaluation set in its own right, to measure the ability of future subjects (or NLP models) for the same task. These evaluation sets can measure the ability of an NLP system with a small number of items, leaving a larger percentage of a dataset for training.

Our contributions are as follows: First, we introduce IRT and describe its benefits and methodology. Second, we apply IRT to Recognizing Textual Entailment (RTE) and show that evaluation sets consisting of a small number of sampled items can provide meaningful information about the RTE task. Our IRT analyses show that different items exhibit varying degrees of difficulty and discrimination power and that high accuracy does not always translate to high scores in relation to human performance. By incorporating IRT, we can learn more about dataset items and move past treating each test case as equal. Using IRT as an evaluation metric allows us to compare NLP systems directly to the performance of humans.

2 Background and Related Work

2.1 Item Response Theory

IRT is one of the most widely used methodologies in psychometrics for scale construction and evaluation. It is typically used to analyze human responses (graded as right or wrong) to a set of questions (called "items"). With IRT individual ability and item characteristics are jointly modeled to predict performance (Baker and Kim, 2004). This statistical model makes the following assumptions: (a) Individuals differ from each other on an unobserved latent trait dimension (called "ability" or "factor"); (b) The probability of correctly answering an item is a function of the person's ability. This function is called the item characteristic curve (ICC) and involves item characteristics as parameters; (c) Responses to different items are independent of each other for a given ability level of the person ("local independence assumption"); (d) Responses from different individuals are independent of each other.

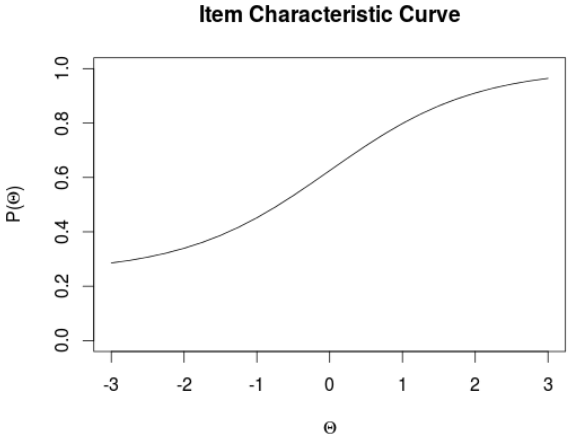


Figure 1: Example ICC for a 3PL model with the following parameters: $a = 1.0$, $b = 0.0$, $c = 0.25$.

More formally, if we let j be an individual, i be an item, and θ_j be the latent ability trait of individual j, then the probability that individual j answers item i correctly can be modeled as:

$$p_{ij}(\theta_j) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta_j - b_i)}} \quad (1)$$

where a_i, b_i, and c_i are item parameters: a_i (the slope or discrimination parameter) is related to the steepness of the curve, b_i (the difficulty parameter) is the level of ability that produces a chance of correct response equal to the average of the upper and lower asymptotes, and c_i (the guessing parameter) is the lower asymptote of the ICC and the probability of guessing correctly. Equation 1 is referred to as the three-parameter logistic (3PL) IRT model. A two-parameter logistic (2PL) IRT model assumes that the guessing parameter c_i is 0.

Figure 1 shows an ICC of a 3PL model. The ICC for a good item will look like a sigmoid plot, and should exhibit a relatively steep increasing ICC between ability levels -3 and 3, where most people are located, in order to have appropriate power to differentiate different levels of ability. We have described a one factor IRT model where ability is uni-dimensional. Multi-factor IRT models would involve two or more latent trait dimensions and will not be elaborated here.

To identify the number of factors in an IRT model, the polychoric correlation matrix of the items is calculated and its ordered eigenvalues are plotted. The

number of factors is suggested by the number of large eigenvalues. It can be further established by fitting (see below) and comparing IRT models with different numbers of factors. Such comparison may use model selection indices such as Akaike Information Criterion (AIC) and Conditional Bayesian Information Criterion (CBIC) and should also take into account the interpretablility of the loading pattern that links items to factors.

An IRT model can be fit to data with the marginal maximum likelihood method through an EM algorithm (Bock and Aitkin, 1981). The marginal likelihood function is the probability to observe the current response patterns as a function of the item parameters with the persons' ability parameters integrated out as random effects. This function is maximized to produce estimates of the item parameters. For IRT models with more than one factor, the slope parameters (i.e. loadings) that relate items and factors must be properly rotated (Browne, 2001) before they can be interpreted. Given the estimated item parameters, Bayesian estimates of the individual person's ability parameters are obtained with the standard normal prior distribution.

After determining the number of factors and fitting the model, the local independence assumption can be checked using the residuals of marginal responses of item pairs (Chen and Thissen, 1997) and the fit of the ICC for each item can be checked with item fit statistics (Orlando and Thissen, 2000) to determine whether an item should be retained or removed. If both tests are passed and all items have proper discrimination power, then the set of items is considered a calibrated measurement scale and the estimated item parameters can be further used to estimate an individual person's ability level.

IRT accounts for differences among items when estimating a person's ability. In addition, ability estimates from IRT are on the ability scale of the population used to estimate item parameters. For example, an estimated ability of 1.2 can be interpreted as 1.2 standard deviations above the average ability in this population. The traditional total number of correct responses generally does not have such quantitative meaning.

IRT has been widely used in educational testing. For example, it plays an instrumental role in the construction, evaluation, or scoring of standardized tests such as the Test of English as a Foreign Language (TOEFL), Graduate Record Examinations (GRE) and the SAT college admissions standardized test.

2.1.1 IRT Terminology

Here we outline common IRT terminology in terms of RTE. An *item* refers to a pair of sentences to which humans or NLP systems assign a label (entailment, contradiction, or neutral). A set of responses to all items (each graded as correct or incorrect) is a *response pattern*. An *evaluation scale* is a test set of items to be administered to an NLP system and assigns an *ability score* (or *theta score*) to the system as its performance.

2.2 Recognizing Textual Entailment

RTE was introduced to standardize the challenge of accounting for semantic variation when building models for a number of NLP applications (Dagan et al., 2006). RTE defines a directional relationship between a pair of sentences, the text (T) and the hypothesis (H). T entails H if a human that has read T would infer that H is true. If a human would infer that H is false, then H contradicts T. If the two sentences are unrelated, then the pair are said to be neutral. Table 1 shows examples of T-H pairs and their respective classifications. Recent state-of-the-art systems for RTE require a large amount of feature engineering and specialization to achieve high performance (Beltagy et al., 2015; Lai and Hockenmaier, 2014; Jimenez et al., 2014).

A number of gold-standard datasets are available for RTE (Marelli et al., 2014; Young et al., 2014; Levy et al., 2014). We consider the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). SNLI examples were obtained using only human-generated sentences with Amazon Mechanical Turk (AMT) to mitigate the problem of poor data that was being used to build models for RTE. In addition, SNLI included a quality control assessment of a sampled portion of the dataset (about 10%, 56,951 sentence pairs). This data was provided to 4 additional AMT users to provide labels (entailment, contradiction, neutral) for the sentence pairs. If at least 3 of the 5 annotators (the original annotator and 4 additional annotators) agreed on a label the item was retained. Most of the items (98%) received

Text	Hypothesis	Label
Retained - 4GS		
1. A toddler playing with a toy car next to a dog	A toddler plays with toy cars while his dog sleeps	Neutral
2. People were watching the tournament in the stadium	The people are sitting outside on the grass	Contradiction
Retained - 5GS		
3. A person is shoveling snow	It rained today	Contradiction
4 Two girls on a bridge dancing with the city skyline in the background	The girls are sisters.	Neutral
5. A woman is kneeling on the ground taking a photograph	A picture is being snapped	Entailment
Removed - 4GS		
6. Two men and one woman are dressed in costume hats	The people are swingers	Neutral
7. Man sweeping trash outside a large statue	A man is on vacation	Contradiction
8. A couple is back to back in formal attire	Two people are facing away from each other	Entailment
9. A man on stilts in a purple, yellow and white costume	A man is performing on stilts	Entailment
Removed - 5GS		
10. A group of soccer players are grabbing onto each other as they go for the ball	A group of football players are playing a game	Contradiction
11. Football players stand at the line of scrimmage	The players are in uniform	Neutral
12. Man in uniform waiting on a wall	Near a wall, a man in uniform is waiting	Entailment

Table 1: Examples of retained & removed sentence pairs. The selection is not based on right/wrong labels but based on IRT model fitting and item elimination process. Note that no 4GS entailment items were retained (Section 4.2)

a gold-standard label. Specifics of SNLI generation are at Bowman et al. (2015).

2.3 Related Work

To identify low-quality annotators (*spammers*), Hovy et al. (2013) modeled annotator responses, either answering correctly or guessing, as a random variable with a guessing parameter varying only across annotators. Passonneau and Carpenter (2014) used the model of Dawid and Skene (1979) in which an annotator's response depends on both the true label and the annotator. In both models an annotator's response depends on an item only through its correct label. In contrast, IRT assumes a more sophisticated response mechanism involving both annotator qualities and item characteristics. To our knowledge we are the first to introduce IRT to NLP and to create a gold standard with the intention of comparing NLP applications to human intelligence.

Bruce and Wiebe (1999) analyze patterns of agreement between annotators in a case-study sentence categorization task, and use a latent-trait model to identify true labels. That work uses 4 annotators at varying levels of expertise and does not consider the discriminating power of dataset items.

Current gold-standard dataset generation methods include web crawling (Guo et al., 2013), automatic and semi-automatic generation (An et al., 2003), and expert (Roller and Stevenson, 2015) and non-expert human annotation (Bowman et al., 2015; Wiebe et al., 1999). In each case validation is required to ensure that the data collected is appropriate and usable for the required task. Automatically generated data can be refined with visual inspection or post-collection processing. Human annotated data usually involves more than one annotator, so that comparison metrics such as Cohen's or Fleiss' κ can be used to determine how much they agree. Disagreements between annotators are resolved by researcher intervention or by majority vote.

3 Methods

We collected and evaluated a random selection from the SNLI RTE dataset (GS_{RTE}) to build our IRT models. We first randomly selected a subset of GS_{RTE}, and then used the sample in an AMT Human Intelligence Task (HIT) to collect more labels

for each text-hypothesis pair. We then applied IRT to evaluate the quality of the examples and used the final IRT models to create evaluation sets (GS_{IRT}) to measure ability for RTE.

3.1 Item Selection

For our evaluation we looked at two sets of data: sentence-pairs selected from SNLI where 4 out of 5 annotators agreed on the gold-standard label (referred to as 4GS), and sentence-pairs where 5 out of 5 annotators agreed on the gold-standard label (referred to as 5GS). We make the assumption for our analysis that the 4GS items are harder than the 5GS items due to the fact that there was not a unanimous decision regarding the gold-standard label.

We selected the subset of GS_{RTE} to use as an examination set in 4GS and 5GS according to the following steps: (1) Identify all "quality-control" items from GS_{RTE} (i.e. items where 5 annotators provided labels, see §2.2), (2) Identify items in this section of the data where 4 of the 5 annotators agreed on the eventual gold label (to be selected from for 4GS) and 5 of the 5 annotators agreed on the gold standard label (to be selected from for 5GS), (3) Randomly select 30 entailment sentence pairs, 30 neutral pairs, and 30 contradiction pairs from those items where 4 of 5 annotators agreed on the gold label (4GS) and those items where 5 of 5 annotators agreed on the gold label (5GS) to obtain two sets of 90 sentence pairs.

90 sentence pairs for 4GS and 5GS were sampled so that the annotation task (supplying 90 labels) could be completed in a reasonably short amount of time during which users remained engaged. We selected items from 4GS and 5GS because both groups are considered high quality for RTE. We evaluated the selected 180 sentence pairs using the model provided with the original dataset (Bowman et al., 2015) and found that accuracy scores were similar compared to performance on the SNLI test set.

3.2 AMT Annotation

For consistency we designed our AMT HIT to match the process used to validate the SNLI quality control items (Bowman et al., 2015) and to generate labels for the SICK RTE dataset (Marelli et al., 2014). Each AMT user was shown 90 premise-hypothesis pairs (either the full 5GS or 4GS set) one pair at a time, and was asked to choose the appropriate label for each. Each user was presented with the full set, as opposed to one-label subsets (e.g. just the entailment pairs) in order to avoid a user simply answering with the same label for each item.

For each 90 sentence-pair set (5GS and 4GS), we collected annotations from 1000 AMT users, resulting in 1000 label annotations for each of the 180 sentence pairs. While there is no set standard for sample sizes in IRT models, this sample size satisfies the standards based on the non-central χ^2 distribution (MacCallum et al., 1996) used when comparing two multidimensional IRT models. This sample size is also appropriate for tests of item fit and local dependence that are based on small contingency tables.

Only AMT users with approval ratings above 97% were used to ensure that users were of a high quality. The task was only available to users located in the United States, as a proxy for identifying English speakers. Attention check questions were included in the HIT, to ensure that users were paying attention and answering to the best of their ability. Responses where the attention-check questions were answered incorrectly were removed. After removing individuals that failed the attention-check, we retained 976 labels for each example in the 4GS set and 983 labels for each example in the 5GS set. Average time spent for each task was roughly 30 minutes, a reasonable amount for AMT users.

3.3 Statistical Analysis

Data collected for 4GS and 5GS were analyzed separately in order to evaluate the differences between "easier" items (5GS) and "harder" items (4GS), and to demonstrate the ability to show that theta score is consistent even if dataset difficulty varies. For both sets of items, the number of factors was identified by a plot of eigenvalues of the 90 x 90 tetrachoric correlation matrix and by a further comparison between IRT models with different number of factors. A target rotation (Browne, 2001) was used to identify a meaningful loading pattern that associates factors and items. Each factor could then be interpreted as the ability of a user to recognize the correct relationship between the sentence pairs associated with that factor (e.g. contradiction).

Once the different factors were associated with different sets of items, we built a unidimensional

	4GS	5GS	Overall
Pairs with majority agreement	95.6%	96.7%	96.1%
Pairs with supermajority agreement	61.1%	82.2%	71.7%

Table 2: Summary statistics from the AMT HITs.

Fleiss' κ	4GS	5GS	Bowman et al. 2015
Contradiction	0.37	0.59	0.77
Entailment	0.48	0.63	0.72
Neutral	0.41	0.54	0.6
Overall	0.43	0.6	0.7

Table 3: Comparison of Fleiss' κ scores with scores from SNLI quality control sentence pairs.

IRT model for each set of items associated with a single factor. We fit and compared one- and two-factor 3PL models to confirm our assumption and the unidimensional structure underlying these items, assuming the possible presence of guessing in people's responses. We further tested the guessing parameter of each item in the one factor 3PL model. If the guessing parameter was not significantly different from 0, a 2PL ICC was used for that particular item.

Once an appropriate model structure was determined, individual items were evaluated for goodness of fit within the model (§2.1). If an item was deemed to fit the ICC poorly or to give rise to local dependence, it was removed for violating model assumptions. Furthermore, if the ICC of an item was too flat, it was removed for low discriminating power between ability levels. The model was then refit with the remaining items. This iterative process continued until no item could be removed (2 to 6 iterations depending on how many items were removed from each set).

The remaining items make up our final test set (GS_{IRT}), which is a calibrated scale of ability to correctly identify the relationship between the two sentence pairs. Parameters of these items were estimated as part of the IRT model and the set of items can be used as an evaluation scale to estimate ability of test-takers or RTE systems. We used the *mirt* R package (Chalmers et al., 2015) for our analyses.

4 Results

4.1 Response Statistics

Table 2 lists key statistics from the AMT HITs. Most of the sampled sentence pairs resulted in a gold standard label being identified via a majority vote. Due to the large number of individuals providing labels during the HIT, we also wanted to see if a gold standard label could be determined via a two-thirds supermajority vote. We found that 28.3% of the sentence pairs did not have a supermajority gold label. This highlights the ambiguity associated with identifying entailment.

We believe that the items selected for analysis are appropriate for our task in that we chose high-quality items, where at least 4 annotators selected the same label, indicating a strong level of agreement (Section 3.1). We argue that our sample is a high-quality portion of the dataset, and further analysis of items where the gold-standard label was only selected by 3 annotators originally would result in lower levels of agreement.

Table 3 shows that the level of agreement as measured by the Fleiss' κ score is much lower when the number of annotators is increased, particularly for the 4GS set of sentence pairs, as compared to scores noted in Bowman et al. (2015). The decrease in agreement is particularly large with regard to contradiction. This could occur for a number of reasons. Recognizing entailment is an inherently difficult task, and classifying a correct label, particularly for contradiction and neutral, can be difficult due to an individual's interpretation of the sentences and assumptions that an individual makes about the key facts of each sentence (e.g. coreference). It may also be the case that the individuals tasked with creating the sentence pairs on AMT created sentences that appeared to contradict a premise text, but can be interpreted differently given a different context.

Before fitting the IRT models we performed a visual inspection of the 180 sentence pairs and removed items clearly not suitable for an evaluation scale due to syntactic or semantic discrepancies. For example item 10 in Table 1 was removed from the 5GS contradiction set for semantic reasons. While many people would agree that the statement is a contradiction due to the difference between football and soccer, individuals from outside the U.S. would possibly consider the two to be synonyms and classify this as entailment. Six such pairs were identified

and removed from the set of 180 items, leaving 174 items for IRT model-fitting.

4.2 IRT Evaluation

4.2.1 IRT Models

We used the methods described in Section 3.3 to build IRT models to scale performance according to the RTE task. For both 4GS and 5GS items three factors were identified, each related to items for the three GS_{RTE} labels (entailment, contradiction, neutral). This suggests that items with the same GS_{RTE} label within each set defines a separate ability. In the subsequent steps, items with different labels were analyzed separately. After analysis, we were left with a subset of the 180 originally selected items. Refer to Table 1 for examples of the retained and removed items based on the IRT analysis. We retained 124 of the 180 items (68.9%). We were able to retain more items from the 5GS datasets (76 out of 90 - 84%) than from the 4GS datasets (48 out of 90 - 53.5%). Items that measure contradiction were retained at the lowest rate for both 4GS and 5GS datasets (66% in both cases). For the 4GS entailment items, our analysis found that a one-factor model did not fit the data, and a two-factor model failed to yield an interpretable loading pattern after rotation. We were unable to build an IRT model that accurately modeled ability to recognize entailment with the obtained response patterns. As a result, no items from the 4GS entailment set were retained.

Figure 2 plots the empirical spline-smoothed ICC of one item (Table 1, item 9) with its estimated response curve. The ICC is not continuously increasing, and thus a logistic function is not appropriate. This item was spotted for poor item fit and removed. Figure 3 shows a comparison between the ICC plot of a retained item (Table 1, item 4) and the ICC of a removed item (Table 1, item 8). Note that the removed item has an ICC that is very flat between -3 and 3. This item cannot discriminate individuals at any common level of ability and thus is not useful.

The items retained for each factor can be considered as an evaluation scale that measures a single ability of an individual test-taker. As each factor is associated with a separate gold-standard label, each factor (θ) is a person's ability to correctly classify the relationship between the text and hypothesis for one such label (e.g. entailment).

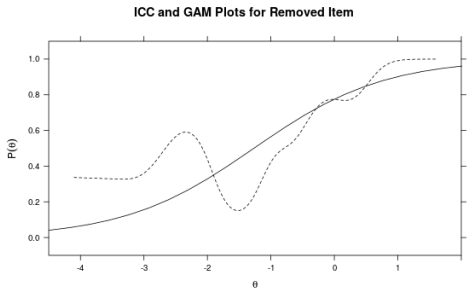


Figure 2: Estimated (solid) and actual (dotted) response curves for a removed item.

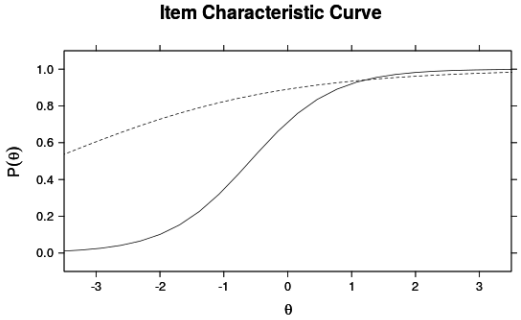


Figure 3: ICCs for retained (solid) and removed (dotted) items.

4.2.2 Item Parameter Estimation

Parameter estimates of retained items for each label are summarized in Table 4, and show that all parameters fall within reasonable ranges. All retained items have 2PL ICCs, suggesting no significant guessing. Difficulty parameters of most items are negative, suggesting that an average AMT user has at least 50% chance to answer these items correctly. Although some minimum difficulties are quite low for standard ranges for a human population, the low range of item difficulty is appropriate for the evaluation of NLP systems. Items in each scale have a wide range of difficulty and discrimination power.

With IRT we can use the heterogeneity of items to properly account for such differences in the estimation of a test-taker's ability. Figure 4 plots the estimated ability of each AMT user from IRT against their total number of correct responses to the retained items in the 4GS contradiction item set. The two estimates of ability differ in many aspects. First, test-takers with the same total score may differ in their IRT score because they have different response

Item Set	Min. Difficulty	Max. Difficulty	Min. Slope	Max. Slope
5GS				
Contradiction	-2.765	0.704	0.846	2.731
Entailment	-3.253	-1.898	0.78	2.61
Neutral	-2.082	-0.555	1.271	3.598
4GS				
Contradiction	-1.829	1.283	0.888	2.753
Neutral	-2.148	0.386	1.133	3.313

Table 4: Parameter estimates of the retained items

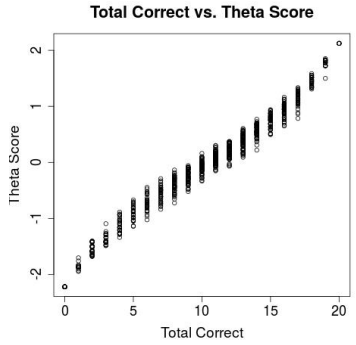


Figure 4: Plot of total correct answers vs. IRT scores.

patterns (i.e. they made mistakes on different items), showing that IRT is able to account for differences among items. Second, despite a rough monotonic trend between the two scores, people with a higher number of correct responses may have a lower ability estimate from IRT.

We can extend this analysis to the case of RTE systems, and use the newly constructed scales to evaluate RTE systems. A system could be trained on an existing dataset and then evaluated using the retained items from the IRT models to estimate a new ability score. This score would be a measurement of how well the system performed with respect to the human population used to fit the model. With this approach, larger sections of datasets can be devoted to training, with a small portion held out to build an IRT model that can be used for evaluation.

4.2.3 Application to an RTE System

As a demonstration, we evaluate the LSTM model presented in Bowman et al. (2015) with the items in our IRT evaluation scales. In addition to the theta scores, we calculate accuracy for the binary classification task of identifying the correct label for all

Item Set	Theta Score	Percentile	Test Acc.
5GS			
Entailment	-0.133	44.83%	96.5%
Contradiction	1.539	93.82%	87.9%
Neutral	0.423	66.28%	88%
4GS			
Contradiction	1.777	96.25%	78.9%
Neutral	0.441	67%	83%

Table 5: Theta scores and area under curve percentiles for LSTM trained on SNLI and tested on GS_{IRT}. We also report the accuracy for the same LSTM tested on all SNLI quality control items (see Section 3.1). All performance is based on binary classification for each label.

items eligible for each subset in Table 5 (e.g. all test items where 5 of 5 annotators labeled the item as *entailment* for 5GS). Note that these accuracy metrics are for subsets of the SNLI test set used for binary classifications and therefore do not compare with the standard SNLI test set accuracy measures.

The theta scores from IRT in Table 5 show that, compared to AMT users, the system performed well above average for contradiction items compared to human performance, and performed around the average for entailment and neutral items. For both the neutral and contradiction items, the theta scores are similar across the 4GS and 5GS sets, whereas the accuracy of the more difficult 4GS items is consistently lower. This shows the advantage of IRT to account for item characteristics in its ability estimates. A similar theta score across sets indicates that we can measure the "ability level" regardless of whether the test set is easy or hard. Theta score is a consistent measurement, compared to accuracy which varies with the difficulty of the dataset.

The theta score and accuracy for 5GS entailment show that high accuracy does not necessarily mean that performance is above average when compared to human performance. However, theta score is not meant to contradict accuracy score, but to provide a better idea of system performance compared against a human population. The theta scores are a result of the IRT model fit using human annotator responses and provide more context about the system performance than an accuracy score can alone. If accuracy is high and theta is close to 0 (as is the case with 5GS entailment), we know that the performance of RTE

is close to the average level of the AMT user population and that 5GS entailment test set was "easy" to both. Theta score and percentile are intrinsically in reference to human performance and independent of item difficulty, while accuracy is intrinsically in reference to a specific set of items.

5 Discussion and Future Work

As NLP systems have become more sophisticated, sophisticated methodologies are required to compare their performance. One approach to create an intelligent gold standard is to use IRT to build models to scale performance on a small section of items with respect to the tested population. IRT models can identify dataset items with different difficulty levels and discrimination powers based on human responses, and identify items that are not appropriate as scale items for evaluation. The resulting small set of items can be used as a scale to score an individual or NLP system. This leaves a higher percentage of a dataset to be used in the training of the system, while still having a valuable metric for testing.

IRT is not without its challenges. A large population is required to provide the initial responses in order to have enough data to fit the models; however, crowdsourcing allows for the inexpensive collection of large amounts of data. An alternative methodology is Classical Test Theory, which has its own limitations, in particular that it is test-centric, and cannot provide information for individual items.

We have introduced Item Response Theory from psychometrics as an alternative method for generating gold-standard evaluation datasets. Fitting IRT models allows us to identify a set of items that when taken together as a test set, can provide a meaningful evaluation of NLP systems with the different difficulty and discriminating characteristics of the items taken into account. We demonstrate the usefulness of the IRT-generated test set by showing that high accuracy does not necessarily indicate high performance when compared to a population of humans.

Future work can adapt this analysis to create evaluation mechanisms for other NLP tasks. The expectation is that systems that perform well using a standard accuracy measure can be stratified based on which types of items they perform well on. High quailty systems should also perform well when the models are used together as an overall test of ability. This new evaluation for NLP systems can lead to new and innovative methods that can be tested against a novel benchmark for performance, instead of gradually incrementing on a classification accuracy metric.

Acknowledgments

We thank the AMT Turkers who completed our annotation task. We would like to also thank the anonymous reviewers for their insightful comments.

This work was supported in part by the HSR&D award IIR 1I01HX001457 from the United States Department of Veterans Affairs (VA). We also acknowledge the support of HL125089 from the National Institutes of Health. This work was also supported in part by the Center for Intelligent Information Retrieval. The contents of this paper do not represent the views of CIIR, NIH, VA, or the United States Government

References

Joohui An, Seungwoo Lee, and Gary Geunbae Lee. 2003. Automatic Acquisition of Named Entity Tagged Corpus from World Wide Web. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 165–168, Stroudsburg, PA, USA. Association for Computational Linguistics.

Frank B. Baker and Seock-Ho Kim. 2004. *Item Response Theory: Parameter Estimation Techniques, Second Edition*. CRC Press, July.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2015. Representing Meaning with a Combination of Logical Form and Vectors. *arXiv:1505.06816 [cs]*. arXiv: 1505.06816.

R Darrell Bock and Murray Aitkin. 1981. Marginal maximum likelihood estimation of item parameters: Application of an em algorithm. *Psychometrika*, 46(4):443–459.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Michael W Browne. 2001. An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, 36(1):111–150.

Rebecca F Bruce and Janyce M Wiebe. 1999. Recognizing subjectivity: a case study in manual tagging. *Natural Language Engineering*, 5(02):187–205.

Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. 2002. Deep blue. *Artificial intelligence*, 134(1):57–83.

Phil Chalmers, Joshua Pritikin, Alexander Robitzsch, and Mateusz Zoltak. 2015. mirt: Multidimensional Item Response Theory, November.

Wen-Hung Chen and David Thissen. 1997. Local Dependence Indexes for Item Pairs Using Item Response Theory. *Journal of Educational and Behavioral Statistics*, 22(3):265–289, September.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer. DOI: 10.1007/11736790_9.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Weiwei Guo, Hao Li, Heng Ji, and Mona T. Diab. 2013. Linking Tweets to News: A Framework to Enrich Short Text Data in Social Media. In *ACL (1)*, pages 239–249. Citeseer.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H Hovy. 2013. Learning whom to trust with mace. In *HLT-NAACL*, pages 1120–1130.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Btiz, and Av Mendizbal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. *SemEval 2014*, page 732.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A Denotational and Distributional Approach to Semantics. *SemEval 2014*, page 329.

Omar Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. *Proc. CoNLL*.

Robert C MacCallum, Michael W Browne, and Hazuki M Sugawara. 1996. Power analysis and determination of sample size for covariance structure modeling. *Psychological methods*, 1(2):130.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, and Fondazione Bruno Kessler. 2014. *A SICK cure for the evaluation of compositional distributional semantic models*.

Maria Orlando and David Thissen. 2000. Likelihood-Based Item-Fit Indices for Dichotomous Item Response Theory Models. *Applied Psychological Measurement*, 24(1):50–64, March.

Rebecca J Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Roland Roller and Mark Stevenson. 2015. Held-out versus Gold Standard: Comparison of Evaluation Strategies for Distantly Supervised Relation Extraction from Medline abstracts. In *Sixth International Workshop on Health Text Mining and Information Analysis (LOUHI)*, page 97.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Janyce M. Wiebe, Rebecca F. Bruce, and Thomas P. O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2(0):67–78, February.

WordRank: Learning Word Embeddings via Robust Ranking

Shihao Ji
Parallel Computing Lab, Intel
shihao.ji@intel.com

Hyokun Yun
Amazon
yunhyoku@amazon.com

Pinar Yanardag
Purdue University
ypinar@purdue.edu

Shin Matsushima
University of Tokyo
shin_matsushima@mist.i.u-tokyo.ac.jp

S. V. N. Vishwanathan
Univ. of California, Santa Cruz
vishy@ucsc.edu

Abstract

Embedding words in a vector space has gained a lot of attention in recent years. While state-of-the-art methods provide efficient computation of word similarities via a low-dimensional matrix embedding, their motivation is often left unclear. In this paper, we argue that word embedding can be naturally viewed as a *ranking* problem due to the ranking nature of the evaluation metrics. Then, based on this insight, we propose a novel framework *WordRank* that efficiently estimates word representations via robust ranking, in which the attention mechanism and robustness to noise are readily achieved via the DCG-like ranking losses. The performance of *WordRank* is measured in word similarity and word analogy benchmarks, and the results are compared to the state-of-the-art word embedding techniques. Our algorithm is very competitive to the state-of-the- arts on large corpora, while outperforms them by a significant margin when the training set is limited (*i.e.*, sparse and noisy). With 17 million tokens, *WordRank* performs almost as well as existing methods using 7.2 billion tokens on a popular word similarity benchmark. Our multi-node distributed implementation of *WordRank* is publicly available for general usage.

1 Introduction

Embedding words into a vector space, such that *semantic* and *syntactic* regularities between words are preserved, is an important sub-task for many applications of natural language processing. Mikolov et al. (2013a) generated considerable excitement in the machine learning and natural language processing communities by introducing a neural network based model, which they call *word2vec*. It was shown that *word2vec* produces state-of-the-art performance on both word similarity as well as word analogy tasks. The word similarity task is to retrieve words that are similar to a given word. On the other hand, word analogy requires answering queries of the form *a:b;c:?*, where *a, b*, and *c* are words from the vocabulary, and the answer to the query must be semantically related to *c* in the same way as *b* is related to *a*. This is best illustrated with a concrete example: Given the query *king:queen;man:?* we expect the model to output *woman*.

The impressive performance of *word2vec* led to a flurry of papers, which tried to explain and improve the performance of *word2vec* both theoretically (Arora et al., 2015) and empirically (Levy and Goldberg, 2014). One interpretation of *word2vec* is that it is approximately maximizing the positive pointwise mutual information (PMI), and Levy and Goldberg (2014) showed that directly optimizing this gives good results. On the other hand, Pennington et al. (2014) showed performance comparable to *word2vec* by using a modified matrix factorization model, which optimizes a log loss.

Somewhat surprisingly, Levy et al. (2015) showed that much of the performance gains of these new word embedding methods are due to certain hyper-parameter optimizations and system-design choices. In other words, if one sets up careful experiments, then existing word embedding models more or less perform comparably to each other. We conjecture that this is because, at a high level, all these methods

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 658–668,
Austin, Texas, November 1-5, 2016.

are based on the following template: From a large text corpus eliminate infrequent words, and compute a $|\mathcal{W}| \times |\mathcal{C}|$ word-context co-occurrence count matrix; a context is a word which appears less than d distance away from a given word in the text, where d is a tunable parameter. Let $w \in \mathcal{W}$ be a word and $c \in \mathcal{C}$ be a context, and let $X_{w,c}$ be the (potentially normalized) co-occurrence count. One learns a function $f(w, c)$ which approximates a transformed version of $X_{w,c}$. Different methods differ essentially in the transformation function they use and the parametric form of f (Levy et al., 2015). For example, *GloVe* (Pennington et al., 2014) uses $f(w, c) = \langle \mathbf{u}_w, \mathbf{v}_c \rangle$ where $\mathbf{u}_w$ and $\mathbf{v}_c$ are k dimensional vectors, $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product, and one approximates $f(w, c) \approx \log X_{w,c}$. On the other hand, as Levy and Goldberg (2014) show, *word2vec* can be seen as using the same $f(w, c)$ as *GloVe* but trying to approximate $f(w, c) \approx \text{PMI}(X_{w,c}) - \log n$, where $\text{PMI}(\cdot)$ is the pairwise mutual information (Cover and Thomas, 1991) and n is the number of negative samples.

In this paper, we approach the word embedding task from a different perspective by formulating it as a *ranking* problem. That is, given a word w, we aim to output an ordered list $(c_1, c_2, \cdots)$ of context words from $\mathcal{C}$ such that words that co-occur with w appear at the top of the list. If $\text{rank}(w, c)$ denotes the rank of c in the list, then typical ranking losses optimize the following objective: $\sum_{(w,c)\in\Omega} \rho\left(\text{rank}(w, c)\right)$, where $\Omega \subset \mathcal{W} \times \mathcal{C}$ is the set of word-context pairs that co-occur in the corpus, and $\rho(\cdot)$ is a ranking loss function that is monotonically increasing and concave (see Sec. 2 for a justification).

Casting word embedding as ranking has two distinctive advantages. First, our method is *discriminative* rather than generative; in other words, instead of modeling (a transformation of) $X_{w,c}$ directly, we only aim to model the relative order of $X_{w,\cdot}$ values in each row. This formulation fits naturally to popular word embedding tasks such as word similarity/analogy since instead of the likelihood of each word, we are interested in finding the most relevant words in a given context[1]. Second, casting word embedding as a ranking problem enables us to design models robust to noise (Yun et al., 2014) and focusing more on differentiating top relevant words, a kind of attention mechanism that has been proved very useful in deep learning (Larochelle and Hinton, 2010; Mnih et al., 2014; Bahdanau et al., 2015). Both issues are very critical in the domain of word embedding since (1) the co-occurrence matrix might be noisy due to grammatical errors or unconventional use of language, *i.e.*, certain words might co-occur purely by chance, a phenomenon more acute in smaller document corpora collected from diverse sources; and (2) it's very challenging to sort out a few most relevant words from a very large vocabulary, thus some kind of attention mechanism that can trade off the resolution on most relevant words with the resolution on less relevant words is needed. We will show in the experiments that our method can mitigate some of these issues; with 17 million tokens our method performs almost as well as existing methods using 7.2 billion tokens on a popular word similarity benchmark.

2 Word Embedding via Ranking

2.1 Notation

We use w to denote a word and c to denote a context. The set of all words, that is, the vocabulary is denoted as $\mathcal{W}$ and the set of all context words is denoted $\mathcal{C}$. We will use $\Omega \subset \mathcal{W} \times \mathcal{C}$ to denote the set of all word-context pairs that were observed in the data, Ω_w to denote the set of contexts that co-occured with a given word w, and similarly Ω_c to denote the words that co-occurred with a given context c. The size of a set is denoted as $|\cdot|$. The inner product between vectors is denoted as $\langle \cdot, \cdot \rangle$.

2.2 Ranking Model

Let $\mathbf{u}_w$ denote the k-dimensional embedding of a word w, and $\mathbf{v}_c$ denote that of a context c. For convenience, we collect embedding parameters for words and contexts as $\mathbf{U} := \{\mathbf{u}_w\}_{w\in\mathcal{W}}$, and $\mathbf{V} := \{\mathbf{v}_c\}_{c\in\mathcal{C}}$.

We aim to capture the relevance of context c for word w by the inner product between their embedding vectors, $\langle \mathbf{u}_w, \mathbf{v}_c \rangle$; the more relevant a context is, the larger we want their inner product to be.

[1]Roughly speaking, this difference in viewpoint is analogous to the difference between pointwise loss function vs listwise loss function used in ranking (Lee and Lin, 2013).

We achieve this by learning a ranking model that is parametrized by $\mathbf{U}$ and $\mathbf{V}$. If we sort the set of contexts $\mathcal{C}$ for a given word w in terms of each context's inner product score with the word, the rank of a specific context c in this list can be written as (Usunier et al., 2009):

$$\begin{aligned}\text{rank}\,(w,c) &= \sum_{c'\in\mathcal{C}\setminus\{c\}} I\left(\langle \mathbf{u}_w, \mathbf{v}_c\rangle - \langle \mathbf{u}_w, \mathbf{v}_{c'}\rangle \leq 0\right) \\ &= \sum_{c'\in\mathcal{C}\setminus\{c\}} I\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle \leq 0\right), \quad (1)\end{aligned}$$

where $I(x \leq 0)$ is a 0-1 loss function which is 1 if $x \leq 0$ and 0 otherwise. Since $I(x \leq 0)$ is a discontinuous function, we follow the popular strategy in machine learning which replaces the 0-1 loss by its convex upper bound $\ell(\cdot)$, where $\ell(\cdot)$ can be any popular loss function for binary classification such as the hinge loss $\ell(x) = \max(0, 1-x)$ or the logistic loss $\ell(x) = \log_2(1+2^{-x})$ (Bartlett et al., 2006). This enables us to construct the following convex upper bound on the rank:

$$\text{rank}\,(w,c) \leq \overline{\text{rank}}\,(w,c) = \sum_{c'\in\mathcal{C}\setminus\{c\}} \ell(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle) \quad (2)$$

It is certainly desirable that the ranking model positions relevant contexts at the top of the list; this motivates us to write the objective function to minimize as:

$$J(\mathbf{U},\mathbf{V}) := \sum_{w\in\mathcal{W}} \sum_{c\in\Omega_w} r_{w,c} \cdot \rho\left(\frac{\overline{\text{rank}}\,(w,c)+\beta}{\alpha}\right) \quad (3)$$

where $r_{w,c}$ is the weight between word w and context c quantifying the association between them, $\rho(\cdot)$ is a monotonically increasing and concave ranking loss function that measures goodness of a rank, and $\alpha > 0$, $\beta > 0$ are the hyperparameters of the model whose role will be discussed later. Following Pennington et al. (2014), we use

$$r_{w,c} = \begin{cases} (X_{w,c}/x_{max})^{\epsilon} & \text{if } X_{w,c} < x_{max} \\ 1 & \text{otherwise,} \end{cases} \quad (4)$$

where we set $x_{max} = 100$ and $\epsilon = 0.75$ in our experiments. That is, we assign larger weights (with a saturation) to contexts that appear more often with the word of interest, and vice-versa. For the ranking loss function $\rho(\cdot)$, on the other hand, we consider the class of monotonically increasing and concave functions. While monotonicity is a natural requirement, we argue that concavity is also important so that the derivative of ρ is always non-increasing; this implies that the ranking loss to be the most sensitive at the top of the list (where the rank is small) and becomes less sensitive at the lower end of the list (where the rank is high). Intuitively this is desirable, because we are interested in a small number of relevant contexts which frequently co-occur with a given word, and thus are willing to tolerate errors on infrequent contexts[2]. Meanwhile, this insensitivity at the bottom of the list makes the model robust to noise in the data either due to grammatical errors or unconventional use of language. Therefore, a single ranking loss function $\rho(\cdot)$ serves two different purposes at two ends of the curve (see the example plots of ρ in Figure 1); while the left hand side of the curve encourages "high resolution" on most relevant words, the right hand side becomes less sensitive (with "low resolution") to infrequent and possibly noisy words[3]. As we will demonstrate in our experiments, this is a fundamental attribute (in addition to the ranking nature) of our method that contributes its superior performance as compared to the state-of-the-arts when the training set is limited (*i.e.*, sparse and noisy).

What are interesting loss functions that can be used for $\rho(\cdot)$? Here are four possible alternatives, all of which have a natural interpretation (see the plots of all four ρ functions in Figure 1(a) and the related work in Sec. 3 for a discussion).

$$\rho_0(x) := x \quad \text{(identity)} \quad (5)$$

$$\rho_1(x) := \log_2(1+x) \quad \text{(logarithm)} \quad (6)$$

$$\rho_2(x) := 1 - \frac{1}{\log_2(2+x)} \quad \text{(negative DCG)} \quad (7)$$

$$\rho_3(x) := \frac{x^{1-t}-1}{1-t} \quad (\log_t \text{ with } t \neq 1) \quad (8)$$

[2]This is similar to the attention mechanism found in human visual system that is able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution" (Larochelle and Hinton, 2010; Mnih et al., 2014).

[3]Due to the linearity of $\rho_0(x) = x$, this ranking loss doesn't have the benefit of attention mechanism and robustness to noise since it treats all ranking errors uniformly.

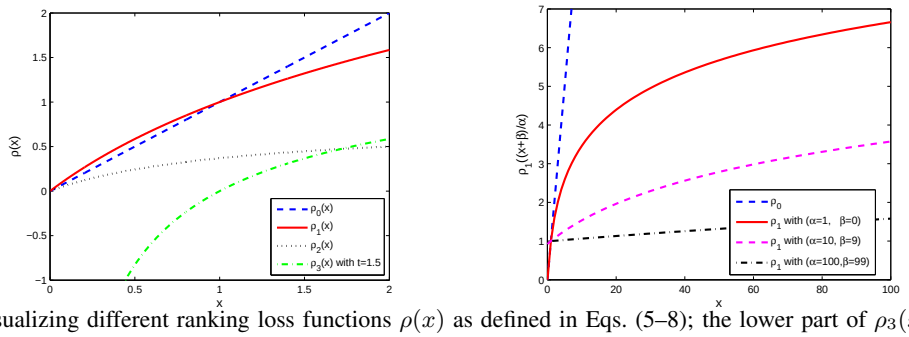


Figure 1: (a) Visualizing different ranking loss functions $\rho(x)$ as defined in Eqs. (5–8); the lower part of $\rho_3(x)$ is truncated in order to visualize the other functions better. (b) Visualizing $\rho_1((x+\beta)/\alpha)$ with different α and β; ρ_0 is included to illustrate the dramatic scale differences between ρ_0 and ρ_1.

We will explore the performance of each of these variants in our experiments. For now, we turn our attention to efficient stochastic optimization of the objective function (3).

2.3 Stochastic Optimization

Plugging (2) into (3), and replacing $\sum_{w\in\mathcal{W}}\sum_{c\in\Omega_w}$ by $\sum_{(w,c)\in\Omega}$, the objective function becomes:

$$J(\mathbf{U},\mathbf{V}) = \sum_{(w,c)\in\Omega} r_{w,c}\cdot \rho\left(\frac{\sum_{c'\in\mathcal{C}\setminus\{c\}}\ell\left(\langle\mathbf{u}_w,\mathbf{v}_c-\mathbf{v}_{c'}\rangle\right)+\beta}{\alpha}\right). \quad (9)$$

This function contains summations over Ω and $\mathcal{C}$, both of which are expensive to compute for a large corpus. Although stochastic gradient descent (SGD) (Bottou and Bousquet, 2011) can be used to replace the summation over Ω by random sampling, the summation over $\mathcal{C}$ cannot be avoided unless $\rho(\cdot)$ is a linear function. To work around this problem, we propose to optimize a linearized upper bound of the objective function obtained through a first-order Taylor approximation. Observe that due to the concavity of $\rho(\cdot)$, we have

$$\rho(x) \le \rho\left(\xi^{-1}\right) + \rho'\left(\xi^{-1}\right)\cdot\left(x-\xi^{-1}\right) \quad (10)$$

for any x and $\xi \neq 0$. Moreover, the bound is tight when $\xi = x^{-1}$. This motivates us to introduce a set of auxiliary parameters $\Xi := \{\xi_{w,c}\}_{(w,c)\in\Omega}$ and define the following upper bound of $J(\mathbf{U},\mathbf{V})$:

$$\overline{J}(\mathbf{U},\mathbf{V},\Xi) := \sum_{(w,c)\in\Omega} r_{w,c}\cdot\left\{\rho(\xi_{wc}^{-1}) + \rho'(\xi_{wc}^{-1})\cdot\left(\alpha^{-1}\beta+\alpha^{-1}\sum_{c'\in\mathcal{C}\setminus\{c\}}\ell\left(\langle\mathbf{u}_w,\mathbf{v}_c-\mathbf{v}_{c'}\rangle\right)-\xi_{w,c}^{-1}\right)\right\}. \quad (11)$$

Note that $J(\mathbf{U},\mathbf{V}) \le \overline{J}(\mathbf{U},\mathbf{V},\Xi)$ for any Ξ, due to (10)[4]. Also, minimizing (11) yields the same $\mathbf{U}$ and $\mathbf{V}$ as minimizing (9). To see this, suppose $\hat{\mathbf{U}} := \{\hat{\mathbf{u}}_w\}_{w\in\mathcal{W}}$ and $\hat{\mathbf{V}} := \{\hat{\mathbf{v}}_c\}_{c\in\mathcal{C}}$ minimizes (9). Then, by letting $\hat{\Xi} := \left\{\hat{\xi}_{w,c}\right\}_{(w,c)\in\Omega}$ where

$$\hat{\xi}_{w,c} = \frac{\alpha}{\sum_{c'\in\mathcal{C}\setminus\{c\}}\ell\left(\langle\hat{\mathbf{u}}_w,\hat{\mathbf{v}}_c-\hat{\mathbf{v}}_{c'}\rangle\right)+\beta}, \quad (12)$$

we have $\overline{J}\left(\hat{\mathbf{U}},\hat{\mathbf{V}},\hat{\Xi}\right) = J\left(\hat{\mathbf{U}},\hat{\mathbf{V}}\right)$. Therefore, it suffices to optimize (11). However, unlike (9), (11) admits an efficient SGD algorithm. To see this, rewrite (11) as

$$\overline{J}(\mathbf{U},\mathbf{V},\Xi) = \sum_{(w,c,c')} r_{w,c}\cdot\left(\frac{\rho(\xi_{w,c}^{-1})+\rho'(\xi_{w,c}^{-1})\cdot(\alpha^{-1}\beta-\xi_{w,c}^{-1})}{|\mathcal{C}|-1} + \frac{1}{\alpha}\rho'(\xi_{w,c}^{-1})\cdot\ell\left(\langle\mathbf{u}_w,\mathbf{v}_c-\mathbf{v}_{c'}\rangle\right)\right), \quad (13)$$

where $(w,c,c') \in \Omega\times(\mathcal{C}\setminus\{c\})$. Then, it can be seen that if we sample uniformly from $(w,c)\in\Omega$ and $c'\in\mathcal{C}\setminus\{c\}$, then $j(w,c,c') :=$

$$|\Omega|\cdot(|\mathcal{C}|-1)\cdot r_{w,c}\cdot\left(\frac{\rho(\xi_{w,c}^{-1})+\rho'(\xi_{w,c}^{-1})\cdot(\alpha^{-1}\beta-\xi_{w,c}^{-1})}{|\mathcal{C}|-1} + \frac{1}{\alpha}\rho'(\xi_{w,c}^{-1})\cdot\ell\left(\langle\mathbf{u}_w,\mathbf{v}_c-\mathbf{v}_{c'}\rangle\right)\right), \quad (14)$$

which does not contain any expensive summations and is an unbiased estimator of (13), i.e., $\mathbb{E}[j(w,c,c')] = \overline{J}(\mathbf{U},\mathbf{V},\Xi)$. On the other hand, one can optimize $\xi_{w,c}$ exactly by using (12). Putting

[4] When $\rho=\rho_0$, one can simply set the auxiliary variables $\xi_{w,c}=1$ because ρ_0 is already a linear function.

everything together yields a stochastic optimization algorithm *WordRank*, which can be specialized to a variety of ranking loss functions $\rho(\cdot)$ with weights $r_{w,c}$ (e.g., DCG (Discounted Cumulative Gain) (Manning et al., 2008) is one of many possible instantiations). Algorithm 1 contains detailed pseudo-code. It can be seen that the algorithm is divided into two stages: a stage that updates $(\mathbf{U}, \mathbf{V})$ and another that updates Ξ. Note that the time complexity of the first stage is $\mathcal{O}(|\Omega|)$ since the cost of each update in Lines 8–10 is independent of the size of the corpus. On the other hand, the time complexity of updating Ξ in Line 15 is $\mathcal{O}(|\Omega|\,|\mathcal{C}|)$, which can be expensive. To amortize this cost, we employ two tricks: we only update Ξ after a few iterations of $\mathbf{U}$ and $\mathbf{V}$ update, and we exploit the fact that the most computationally expensive operation in (12) involves a matrix and matrix multiplication which can be calculated efficiently via the SGEMM routine in BLAS (Dongarra et al., 1990).

Algorithm 1 *WordRank* algorithm.

1: η: step size
2: **repeat**
3: `// Stage 1: Update` $\mathbf{U}$ `and` $\mathbf{V}$
4: **repeat**
5: Sample (w, c) uniformly from Ω
6: Sample c' uniformly from $\mathcal{C} \setminus \{c\}$
7: `// following three updates are executed simultaneously`
8: $\mathbf{u}_w \leftarrow \mathbf{u}_w - \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'} \rangle\right) \cdot (\mathbf{v}_c - \mathbf{v}_{c'})$
9: $\mathbf{v}_c \leftarrow \mathbf{v}_c - \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'} \rangle\right) \cdot \mathbf{u}_w$
10: $\mathbf{v}_{c'} \leftarrow \mathbf{v}_{c'} + \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'} \rangle\right) \cdot \mathbf{u}_w$
11: **until** $\mathbf{U}$ and $\mathbf{V}$ are converged
12: `// Stage 2: Update` Ξ
13: **for** $w \in \mathcal{W}$ **do**
14: **for** $c \in \mathcal{C}$ **do**
15: $\xi_{w,c} = \alpha / \left(\sum_{c' \in \mathcal{C} \setminus \{c\}} \ell\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'} \rangle\right) + \beta \right)$
16: **end for**
17: **end for**
18: **until** $\mathbf{U}$, $\mathbf{V}$ and Ξ are converged

2.4 Parallelization

The updates in Lines 8–10 have one remarkable property: To update $\mathbf{u}_w$, $\mathbf{v}_c$ and $\mathbf{v}_{c'}$, we only need to *read* the variables $\mathbf{u}_w$, $\mathbf{v}_c$, $\mathbf{v}_{c'}$ and $\xi_{w,c}$. What this means is that updates to another triplet of variables $\mathbf{u}_{\hat{w}}$, $\mathbf{v}_{\hat{c}}$ and $\mathbf{v}_{\hat{c}'}$ can be performed independently. This observation is the key to developing a parallel optimization strategy, by distributing the computation of the updates among multiple processors. Due to lack of space, details including pseudo-code are relegated to the supplementary material.

2.5 Interpreting of α and β

The update (12) indicates that $\xi_{w,c}^{-1}$ is proportional to $\overline{\mathrm{rank}}\,(w, c)$. On the other hand, one can observe that the loss function $\ell\,(\cdot)$ in (14) is weighted by a $\rho'\left(\xi_{w,c}^{-1}\right)$ term. Since $\rho\,(\cdot)$ is concave, its gradient $\rho'\,(\cdot)$ is monotonically non-increasing (Rockafellar, 1970). Consequently, when $\overline{\mathrm{rank}}\,(w, c)$ and hence $\xi_{w,c}^{-1}$ is large, $\rho'\left(\xi_{w,c}^{-1}\right)$ is small. In other words, the loss function "gives up" on contexts with high ranks in order to focus its attention on top of the list. The rate at which the algorithm gives up is determined by the hyperparameters α and β. For the illustration of this effect, see the example plots of ρ_1 with different α and β in Figure 1(b). Intuitively, α can be viewed as a *scale* parameter while β can be viewed as an *offset* parameter. An equivalent interpretation is that by choosing different values of α and β one can modify the behavior of the ranking loss $\rho\,(\cdot)$ in a problem dependent fashion. In our experiments, we found that a common setting of $\alpha = 1$ and $\beta = 0$ often yields uncompetitive performance, while setting $\alpha = 100$ and $\beta = 99$ generally gives good results.

3 Related Work

Our work sits at the intersection of word embedding and ranking optimization. As we discussed in Sec. 2.2 and Sec. 2.5, it's also related to the attention mechanism widely used in deep learning. We therefore review the related work along these three axes.

Word Embedding. We already discussed some related work (*word2vec* and *GloVe*) on word embedding in the introduction. Essentially, *word2vec* and *GloVe* derive word representations by modeling a transformation (PMI or log) of $X_{w,c}$ directly, while

WordRank learns word representations via robust ranking. Besides these state-of-the-art techniques, a few ranking-based approaches have been proposed for word embedding recently, e.g., (Collobert and Weston, 2008; Vilnis and McCallum, 2015; Liu et al., 2015). However, all of them adopt a pair-wise binary classification approach with a linear ranking loss ρ_0. For example, (Collobert and Weston, 2008; Vilnis and McCallum, 2015) employ a hinge loss on positive/negative word pairs to learn word representations and ρ_0 is used *implicitly* to evaluate ranking losses. As we discussed in Sec. 2.2, ρ_0 has no benefit of the attention mechanism and robustness to noise since its linearity treats all the ranking errors uniformly; empirically, sub-optimal performances are often observed with ρ_0 in our experiments. More recently, by extending the Skip-Gram model of *word2vec*, Liu et al. (2015) incorporates additional pair-wise constraints induced from 3rd-party knowledge bases, such as WordNet, and learns word representations jointly. In contrast, *WordRank* is a fully ranking-based approach without using any additional data source for training.

Robust Ranking. The second line of work that is very relevant to *WordRank* is that of ranking objective (3). The use of score functions $\langle \mathbf{u}_w, \mathbf{v}_c \rangle$ for ranking is inspired by the latent collaborative retrieval framework of Weston et al. (2012). Writing the rank as a sum of indicator functions (1), and upper bounding it via a convex loss (2) is due to Usunier et al. (2009). Using $\rho_0(\cdot)$ (5) corresponds to the well-known pairwise ranking loss (see *e.g.*, (Lee and Lin, 2013)). On the other hand, Yun et al. (2014) observed that if they set $\rho = \rho_2$ as in (7), then $-J(\mathbf{U}, \mathbf{V})$ corresponds to the DCG (Discounted Cumulative Gain), one of the most popular ranking metrics used in web search ranking (Manning et al., 2008). In their RobiRank algorithm they proposed the use of $\rho = \rho_1$ (6), which they considered to be a special function for which one can derive an efficient stochastic optimization procedure. However, as we showed in this paper, the general class of monotonically increasing concave functions can be handled efficiently. Another important difference of our approach is the hyperparameters α and β, which we use to modify the behavior of ρ, and which we find are critical to achieve good empirical results. Ding and Vishwanathan (2010) proposed the use of $\rho = \log_t$ in the context of robust *binary* classification, while here we are concerned with ranking, and our formulation is very general and applies to a variety of ranking losses $\rho(\cdot)$ with weights $r_{w,c}$. Optimizing over $\mathbf{U}$ and $\mathbf{V}$ by distributing the computation across processors is inspired by work on distributed stochastic gradient for matrix factorization (Gemulla et al., 2011).

Attention. Attention is one of the most important advancements in deep learning in recent years (Larochelle and Hinton, 2010), and is now widely used in state-of-the-art image recognition and machine translation systems (Mnih et al., 2014; Bahdanau et al., 2015). Recently, attention has also been applied to the domain of word embedding. For example, under the intuition that not all contexts are created equal, Wang et al. (2015) assign an importance weight to each word type at each context position and learn an attention-based Continuous Bag-Of-Words (CBOW) model. Similarly, within a ranking framework, *WordRank* expresses the context importance by introducing the auxiliary variable $\xi_{w,c}$, which "gives up" on contexts with high ranks in order to focus its attention on top of the list.

4 Experiments

In our experiments, we first evaluate the impact of the weight $r_{w,c}$ and the ranking loss function $\rho(\cdot)$ on the test performance using a small dataset. We then pick the best performing model and compare it against *word2vec* (Mikolov et al., 2013b) and *GloVe* (Pennington et al., 2014). We closely follow the framework of Levy et al. (2015) to set up a careful and fair comparison of the three methods. Our code is publicly available at `https://bitbucket.org/shihaoji/wordrank`.

Training Corpus Models are trained on a combined corpus of 7.2 billion tokens, which consists of the 2015 Wikipedia dump with 1.6 billion tokens, the WMT14 News Crawl[5] with 1.7 billion tokens, the "One Billion Word Language Modeling Benchmark"[6] with almost 1 billion tokens, and UMBC

[5] `http://www.statmt.org/wmt14/translation-task.html`

[6] `http://www.statmt.org/lm-benchmark`

Corpus Size	17M*	32M	64M	128M	256M	512M	1.0B	1.6B	7.2B
Vocabulary Size $\|\mathcal{W}\|$	71K	100K	100K	200K	200K	300K	300K	400K	620K
Window Size win	15	15	15	10	10	10	10	10	10
Dimension k	100	100	100	200	200	300	300	300	300

* This is the Text8 dataset from `http://mattmahoney.net/dc/text8.zip`, which is widely used for word embedding demo.

Table 1: Parameter settings used in the experiments.

Task	Robi	ρ_0		ρ_1		ρ_2		ρ_3	
		off	on	off	on	off	on	off	on
Similarity	41.2	69.0	<u>71.0</u>	66.7	70.4	66.8	70.8	68.1	68.0
Analogy	22.7	24.9	31.9	34.3	<u>44.5</u>	32.3	40.4	33.6	42.9

Table 2: Performance of different ρ functions on Text8 dataset with 17M tokens.

webbase corpus[7] with around 3 billion tokens. The pre-processing pipeline breaks the paragraphs into sentences, tokenizes and lowercases each corpus with the Stanford tokenizer. We further clean up the dataset by removing non-ASCII characters and punctuation, and discard sentences that are shorter than 3 tokens or longer than 500 tokens. In the end, we obtain a dataset of 7.2 billion tokens, with the first 1.6 billion tokens from Wikipedia. When we want to experiment with a smaller corpus, we extract a subset which contains the specified number of tokens.

Co-occurrence matrix construction We use the *GloVe* code to construct the co-occurrence matrix X, and the same matrix is used to train *GloVe* and *WordRank* models. When constructing X, we must choose the size of the vocabulary, the context window and whether to distinguish left context from right context. We follow the findings and design choices of *GloVe* and use a symmetric window of size win with a decreasing weighting function, so that word pairs that are d words apart contribute $1/d$ to the total count. Specifically, when the corpus is small (*e.g.*, 17M, 32M, 64M) we let $win = 15$ and for larger corpora we let $win = 10$. The larger window size alleviates the data sparsity issue for small corpus at the expense of adding more noise to X. The parameter settings used in our experiments are summarized in Table 1.

Using the trained model It has been shown by Pennington et al. (2014) that combining the $\mathbf{u}_w$ and $\mathbf{v}_c$ vectors with equal weights gives a small boost in performance. This vector combination was originally motivated as an ensemble method (Pennington et al., 2014), and later Levy et al. (2015) provided a different interpretation of its effect on the cosine similarity function, and show that adding context vectors effectively adds first-order similarity terms to the second-order similarity function. In our experiments, we find that vector combination boosts the performance in word analogy task when training set is small, but when dataset is large enough (*e.g.*, 7.2 billion tokens), vector combination doesn't help anymore. More interestingly, for the word similarity task, we find that vector combination is detrimental in all the cases, sometimes even substantially[8]. Therefore, we will always use $\mathbf{u}_w$ on word similarity task, and use $\mathbf{u}_w + \mathbf{v}_c$ on word analogy task unless otherwise noted.

4.1 Evaluation

Word Similarity We use six datasets to evaluate word similarity: WS-353 (Finkelstein et al., 2002) partitioned into two subsets: WordSim Similarity and WordSim Relatedness (Agirre et al., 2009); MEN (Bruni et al., 2012); Mechanical Turk (Radinsky et al., 2011); Rare words (Luong et al., 2013); and SimLex-999 (Hill et al., 2014). They contain word pairs together with human-assigned similarity judgments. The word representations are evaluated by ranking the pairs according to their cosine similarities, and measuring the Spearman's rank correlation coefficient with the human judgments.

[7] `http://ebiquity.umbc.edu/resource/html/id/351`

[8] This is possible since we optimize a ranking loss: the absolute scores don't matter as long as they yield an ordered list correctly. Thus, *WordRank*'s $\mathbf{u}_w$ and $\mathbf{v}_c$ are less comparable to each other than those generated by *GloVe*, which employs a point-wise L_2 loss.

Word Analogies For this task, we use the Google analogy dataset (Mikolov et al., 2013a). It contains 19544 word analogy questions, partitioned into 8869 semantic and 10675 syntactic questions. A question is correctly answered only if the algorithm selects the word that is exactly the same as the correct word in the question: synonyms are thus counted as mistakes. There are two ways to answer these questions, namely, by using 3CosAdd or 3CosMul (see (Levy and Goldberg, 2014) for details). We will report scores by using 3CosAdd by default, and indicate when 3CosMul gives better performance.

4.2 The impact of $r_{w,c}$ and $\rho(\cdot)$

In Sec. 2.2 we argued the need for adding weight $r_{w,c}$ to ranking objective (3), and we also presented our framework which can deal with a variety of ranking loss functions ρ. We now study the utility of these two ideas. We report results on the 17 million token dataset in Table 2. For the similarity task, we use the WS-353 test set and for the analogy task we use the Google analogy test set. The best scores for each task are underlined. We set $t = 1.5$ for ρ_3. "Off" means that we used uniform weight $r_{w,c} = 1$, and "on" means that $r_{w,c}$ was set as in (4). For comparison, we also include the results using RobiRank (Yun et al., 2014)[9].

It can be seen from Table 2 that adding the weight $r_{w,c}$ improves performance in all the cases, especially on the word analogy task. Among the four ρ functions, ρ_0 performs the best on the word similarity task but suffers notably on the analogy task, while $\rho_1 = \log$ performs the best overall. Given these observations, which are consistent with the results on large scale datasets, in the experiments that follow we only report *WordRank* with the best configuration, *i.e.*, using ρ_1 with the weight $r_{w,c}$ as defined in (4).

4.3 Comparison to state-of-the-arts

In this section we compare the performance of *WordRank* with *word2vec*[10] and *GloVe*[11], by using the code provided by the respective authors. For a fair comparison, *GloVe* and *WordRank* are given as input the same co-occurrence matrix X; this eliminates differences in performance due to window size and other such artifacts, and the same parameters are used to *word2vec*. Moreover, the embedding dimensions used for each of the three methods is the same (see Table 1). With *word2vec*, we train the Skip-Gram with Negative Sampling (SGNS) model since it produces state-of-the-art performance, and is widely used in the NLP community (Mikolov et al., 2013b). For *GloVe*, we use the default parameters as suggested by (Pennington et al., 2014). The results are provided in Figure 2 (also see Table 4 in the supplementary material for additional details).

As can be seen, when the size of corpus increases, in general all three algorithms improve their prediction accuracy on both tasks. This is to be expected since a larger corpus typically produces better statistics and less noise in the co-occurrence matrix X. When the corpus size is small (*e.g.*, 17M, 32M, 64M, 128M), *WordRank* yields the best performance with significant margins among three, followed by *word2vec* and *GloVe*; when the size of corpus increases further, on the word analogy task *word2vec* and *GloVe* become very competitive to *WordRank*, and eventually perform neck-to-neck to each other (Figure 2(b)). This is consistent with the findings of (Levy et al., 2015) indicating that when the number of tokens is large even simple algorithms can perform well. On the other hand, *WordRank* is dominant on the word similarity task for all the cases (Figure 2(a)) since it optimizes a ranking loss *explicitly*, which aligns more naturally with the objective of word similarity than the other methods; with 17 million tokens our method performs almost as well as existing methods using 7.2 billion tokens on the word similarity benchmark.

To further evaluate the model performance on the word similarity/analogy tasks, we use the best performing models trained on the 7.2-billion-token corpus to predict on the six word similarity datasets described in Sec. 4.1. Moreover, we breakdown the performance of the models on the Google word analogy dataset into the semantic and syntactic subtasks. Results are listed in Table 3. As can be seen, *WordRank* outperforms *word2vec* and *GloVe* on 5 of 6 similarity tasks, and 1 of 2 Google analogy subtasks.

[9] We used the code provided by the authors at `https://bitbucket.org/d_ijk_stra/robirank`. Although related to RobiRank, we attribute the superior performance of *WordRank* to the use of weight $r_{w,c}$ (4), introduction of hyperparameters α and β, and many implementation details.

[10] `https://code.google.com/p/word2vec/`

[11] `http://nlp.stanford.edu/projects/glove`

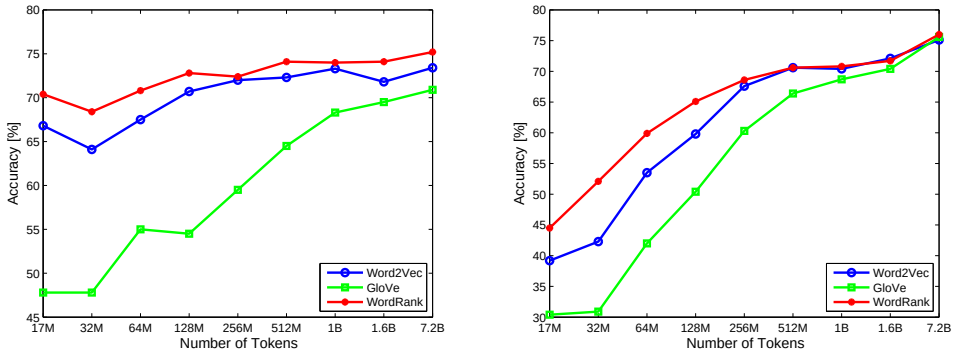


Figure 2: Performance evolution as a function of corpus size (a) on WS-353 word similarity benchmark; (b) on Google word analogy benchmark.

Model	Word Similarity						Word Analogy	
	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. MT	Luong et al. RW	Hill et al. SimLex	Goog Sem.	Goog Syn.
word2vec	73.9	60.9	75.4	66.4	45.5	36.6	78.8	72.0
GloVe	75.7	67.5	78.8	69.7	43.6	41.6	80.9	71.1
WordRank	79.4	70.5	78.1	73.5	47.4	43.5	78.4	74.7

Table 3: Performance of the best *word2vec*, *GloVe* and *WordRank* models, learned from 7.2 billion tokens, on six similarity tasks and Google semantic and syntactic subtasks.

5 Visualizing the results

To understand whether *WordRank* produces *syntatically* and *semantically* meaningful vector space, we did the following experiment: we use the best performing model produced using 7.2 billion tokens, and compute the nearest neighbors of the word "cat". We then visualize the words in two dimensions by using t-SNE (Maaten and Hinton, 2008). As can be seen in Figure 3, our ranking-based model is indeed capable of capturing both semantic (*e.g.*, cat, feline, kitten, tabby) and syntactic (*e.g.*, leash, leashes, leashed) regularities of the English language.

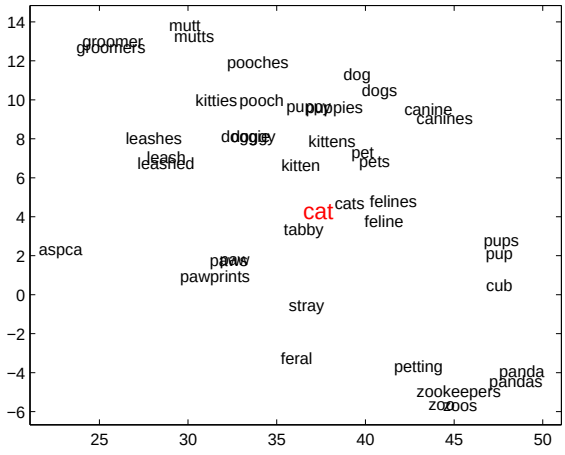


Figure 3: Nearest neighbors of "cat" found by projecting a 300d word embedding learned from *WordRank* onto a 2d space.

6 Conclusion

We proposed *WordRank*, a ranking-based approach, to learn word representations from large scale textual corpora. The most prominent difference between our method and the state-of-the-art techniques, such as *word2vec* and *GloVe*, is that *WordRank* learns word representations via a robust ranking model, while *word2vec* and *GloVe* typically model a transformation of co-occurrence count $X_{w,c}$ directly. Moreover, by a ranking loss function $\rho(\cdot)$, *WordRank* achieves its attention mechanism and robustness to noise naturally, which are usually lacking in other ranking-based approaches. These attributes significantly boost the performance of *WordRank* in the cases where training data are sparse and noisy. Our multi-node distributed implementation of *WordRank* is publicly available for general usage.

Acknowledgments

We'd like to thank Omer Levy for sharing his script for preprocessing the corpora used in the paper. We also thank the anonymous reviewers for their valuable comments and suggestions.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. *Proceedings of Human Language Technologies*, pages 19–27.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. Technical report, ArXiV. `http://arxiv.org/pdf/1502.03520.pdf`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. 2006. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Léon Bottou and Olivier Bousquet. 2011. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in technicolor. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 136–145.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

T. M. Cover and J. A. Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons, New York.

Nan Ding and S. V. N. Vishwanathan. 2010. t-logistic regression. In Richard Zemel, John Shawe-Taylor, John Lafferty, Chris Williams, and Alan Culota, editors, *Advances in Neural Information Processing Systems 23*.

J. J. Dongarra, J. Du Croz, S. Duff, and S. Hammarling. 1990. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16:1–17.

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

R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Conference on Knowledge Discovery and Data Mining*, pages 69–77.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Hugo Larochelle and Geoffrey E. Hinton. 2010. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS) 23*, pages 1243–1251.

Ching-Pei Lee and Chih-Jen Lin. 2013. Large-scale linear ranksvm. *Neural Computation*. To Appear.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Max Welling, Zoubin Ghahramani, Corinna Cortes, Neil Lawrence, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

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

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1501–1511.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

L. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *jmlr*, 9:2579–2605.

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In Chris Burges, Leon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems 26*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NIPS) 27*, pages 2204–2212.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. *Proceedings of the 20th international conference on World Wide Web*, pages 337–346.

R. T. Rockafellar. 1970. *Convex Analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, Princeton, NJ.

Nicolas Usunier, David Buffoni, and Patrick Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *Proceedings of the International Conference on Machine Learning*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ling Wang, Chu-Cheng Lin, Yulia Tsvetkov, Silvio Amir, Ramon Fernandez Astudillo, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Not all contexts are created equal: Better word representations with variable attention. In *EMNLP*.

Jason Weston, Chong Wang, Ron Weiss, and Adam Berenzweig. 2012. Latent collaborative retrieval. *arXiv preprint arXiv:1206.4603*.

Hyokun Yun, Parameswaran Raman, and S. V. N. Vishwanathan. 2014. Ranking via robust binary classification and parallel parameter estimation in large-scale data. In *nips*.

Exploring Semantic Representation in Brain Activity Using Word Embeddings

Yu-Ping Ruan[1], **Zhen-Hua Ling**[1] **and Yu Hu**[1,2]
[1]National Engineering Laboratory for Speech and Language Information Processing
University of Science and Technology of China, Hefei, China
[2]iFLYTEK Research, Hefei, China
ypruan@mail.ustc.edu.cn, zhling@ustc.edu.cn, yuhu@iflytek.com

Abstract

In this paper, we utilize distributed word representations (i.e., word embeddings) to analyse the representation of semantics in brain activity. The brain activity data were recorded using functional magnetic resonance imaging (fMRI) when subjects were viewing words. First, we analysed the functional selectivity of different cortex areas by calculating the correlations between neural responses and several types of word representations, including skip-gram word embeddings, visual semantic vectors, and primary visual features. The results demonstrated consistency with existing neuroscientific knowledge. Second, we utilized behavioural data as the semantic ground truth to measure their relevance with brain activity. A method to estimate word embeddings under the constraints of brain activity similarities is further proposed based on the semantic word embedding (SWE) model. The experimental results show that the brain activity data are significantly correlated with the behavioural data of human judgements on semantic similarity. The correlations between the estimated word embeddings and the semantic ground truth can be effectively improved after integrating the brain activity data for learning, which implies that semantic patterns in neural representations may exist that have not been fully captured by state-of-the-art word embeddings derived from text corpora.

1 Introduction

Recently, the topic of exploring semantic representation in human brain has attracted the attention of researchers from both neuroscience and computational linguistics fields. In these studies, concepts are represented in terms of neural activation patterns in the brain that can be recorded by functional magnetic resonance imaging (fMRI) (Haxby et al., 2001). It has been found that the semantic space shared among different individuals is distributed continuously across the cortex (Huth et al., 2012). A recent study proposed an efficient way to measure and visualize the semantic selectivity of different cortex areas (Huth et al., 2016).

Similar to the distributed semantic representation in the brain, describing the meaning of a word using a dense low-dimensional and continuous vector (i.e., word embedding) is currently a popular approach in computational linguistics (Hinton et al., 1986; Turney et al., 2010). Word embeddings are commonly estimated from large text corpora utilizing statistics concerning the co-occurrences of words (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014). To investigate the correlation between word embeddings and the brain activity involved in viewing words, Mitchell et al. (2008) designed a computational model to predict brain responses using hand-tailored word embeddings as input. Further, Fyshe et al. (2014) proposed a joint non-negative sparse embedding (JNNSE) method to combine fMRI data and textual data to estimate word embeddings. This work improved the correlation between word embeddings and human behavioural data, which lends support to the view that fMRI data can provide additional semantic information that may not exist in textual data.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 669–679,
Austin, Texas, November 1-5, 2016.

The factors that can influence the activities of cortex areas are diverse. Recent studies show that visual semantic features such as bag-of-visual-words (BoVW) are significantly correlated with the fMRI data captured when viewing words (Anderson et al., 2013). The primary visual features derived using Gabor wavelets can be used to determine the images presented to the subjects from their recorded brain activity (Kay et al., 2008; Naselaris et al., 2009). Some other research work also indicates that visual experiences (Nishimoto et al., 2011) and speech information (Ryali et al., 2010) can affect neural responses in cortex areas.

In this paper, we first study the semantic representation of words in brain activity by correlation analysis (Anderson et al., 2013; Carlson et al., 2014). Then, we calculate the correlations between subjects' neural responses when viewing words and three types of word representations: skip-gram word embeddings, primary visual features, and visual semantic vectors. The goal of doing this is to investigate whether these representations can account for the brain data and the functional selectivity of different cortex areas. Then, we utilize behavioural data as the semantic ground truth to measure the semantic relevance of brain activity. A method of estimating word embeddings within the constraints of similar brain activities is proposed. This method is based on the semantic word embedding (SWE) model (Liu et al., 2015) which develops from the skip-gram model. It aims at verifying whether textual data and brain activity data can be complementary to derive word embeddings that are more consistent with human judgement.

The contributions of this study are twofold. First, this study involved a comprehensive correlation analysis on brain activity data and state-of-the-art skip-gram word embeddings at both whole-brain and brain lobe levels. Primary visual features and visual semantic vectors are also introduced as auxiliary representations to better understand the functional selectivity across the cortex. Some results of this analysis are interpretable using existing neuroscience knowledge. Second, to our knowledge, this study marks the first attempt to integrate brain activity data into the skip-gram model for estimating word embeddings. The experimental results show that the correlation between the estimated word embeddings and the behavioural measure of semantics can be effectively improved after integrating brain activity data for learning.

2 Related work

The correlation between brain data and word vectors has been studied in previous work. The experiments in Carlson et al. (2014) adopted brain activity data for correlation analysis from only the ventral temporal pathway, not from the whole brain. Anderson et al. (2013) performed correlation analysis using the voxels of the whole brain and compared the HAL-based textual semantic model (Lund and Burgess, 1996) with the BoVW-based visual semantic model (Sivic and Zisserman, 2003; Csurka et al., 2004) in terms of these two model's ability to account for the patterns found in the neural data. However, the experiments in Anderson et al. (2013) failed to detect differential interactions of semantic models with brain areas. In this paper, considering the popularity of word embedding estimation approaches based on neural networks in recent years, we adopt skip-gram word embeddings (Mikolov et al., 2013a) for correlation analysis. To our knowledge, this is the first time that the association between skip-gram word embeddings and brain activity data have been studied. Furthermore, our work improves on the voxel selection strategy used in Anderson et al. (2013), leading to more interpretable results when demonstrating the functional selectivity of brain areas.

To our knowledge, the first and only attempt to integrate brain activity data into the acquisition of textual word embedding is the JNNSE method (Fyshe et al., 2014). In this method, word embeddings were estimated as latent representations using matrix factorization. The objective functions contained additional constraints for reconstructing brain activity data. In this paper, we adopt the SWE model (Liu et al., 2015) to incorporate brain activity knowledge into word embedding estimation. The SWE model was developed from the skip-gram model. In SWE, semantically related knowledge is converted into inequality constraints for learning word embeddings. The experimental results show that our proposed method using SWE can improve

the semantic consistency between word embeddings and human judgements.

3 From Skip-Gram to SWE

3.1 Skip-gram model

The skip-gram model (Mikolov et al., 2013b) adopts a neural network structure to derive the distributed representation of words from textual corpus. The word vectors are learned based on the distributional hypothesis (Harris, 1954; Miller and Charles, 1991), which assumes that words with similar contexts tend to have similar semantic meanings. For a sequence of training data of T words, denoted as $\{w_1, w_2, w_3, \cdots, w_T\}$, the skip-gram model is trained to maximize the following objective function

$$Q = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t), \quad (1)$$

where w_t and w_{w+j} are the central word and neighbouring words in a context window respectively, and c denotes the size of the context window. The conditional probability $p(w_{t+j}|w_t)$ in Eq.(1) is calculated as

$$p(w_{t+j}|w_t) = \frac{\exp(\mathbf{w}_{t+j}^{(2)} \cdot \mathbf{w}_t^{(1)})}{\sum_{k=1}^{V}\exp(\mathbf{w}_k^{(2)} \cdot \mathbf{w}_t^{(1)})}, \quad (2)$$

where $\mathbf{w}_t^{(1)}$ and $\mathbf{w}_k^{(2)}$ denote row vectors in the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ respectively, and V is the vocabulary size of the corpus. The matrix $\mathbf{W}^{(1)}$ stores the word vectors of input central words, and the matrix $\mathbf{W}^{(2)}$ stores the word vectors of predicted neighbouring words. The optimization of the objective function Q is solved by the stochastic gradient descent (SGD) method (Mikolov et al., 2013b). Finally, the learned matrix $\mathbf{W}^{(1)}$ is used as the estimated word embeddings of all words in the vocabulary.

3.2 Semantic word embedding (SWE)

The skip-gram model learns word embeddings based on the distributional hypothesis; however, this hypothesis still has some limitations. For example, antonyms often appear in similar contexts although they have opposite meanings. The semantic word embedding (SWE) model (Liu et al., 2015) has been proposed to address this issue by incorporating external semantic knowledge into the text-based learning process for word embeddings.

In this method, semantic knowledge is represented as a set of ranking inequalities. Each inequality contains a triplet (i, j, k) of three words $\{w_i, w_j, w_k\}$ with a similarity relation

$$similarity(w_i, w_j) > similarity(w_i, w_k), \quad (3)$$

which can be notated in simplified form as $s_{ij} > s_{ik}$. Then, the learning method of SWE is defined as the following constrained optimization problem

$$\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\} = arg\max_{\mathbf{W}^{(1)},\mathbf{W}^{(2)}} Q(\mathbf{W}^{(1)}, \mathbf{W}^{(2)}),$$
$$s.t.\ s_{ij} > s_{ik}, \forall (i, j, k) \in S, \quad (4)$$

where function Q is defined in Eq. (1) and S denotes the inequality set. Then, the constrained optimization problem in Eq. (4) is simplified into an unconstrained problem by introducing a penalty term into the objective function of the skip-gram model. The penalty term is defined as

$$D = \sum_{(i,j,k)\in S} f(i, j, k), \quad (5)$$

where $f(i, j, k) = \max(0, s_{ik} - s_{ij})$ is a Hinge loss function. Finally, the object function to be maximized in SWE can be written as follows:

$$Q' = Q - \beta \cdot D, \quad (6)$$

where β is a parameter to control the contribution of the penalty term. Similar to the skip-gram model, the Q' function in the SWE model is optimized using SGD to estimate word embeddings. The detailed formulae can be found in Liu et al. (2015).

3.3 Integrating brain activity into SWE

In the implementation of the SWE model in Liu et al. (2015), the ranking inequalities were collected using hypernym-hyponym and synonym-antonym relationships extracted from WordNet (Fellbaum and others, 1998). In this paper, the SWE model is utilized as a tool to explore the semantic relevance of brain activity by examining the performance of the estimated word embeddings after integrating brain-activity-related knowledge. Therefore, we construct

the ranking inequalities in Eq. (3) using brain activity data. When a subject is viewing a word, the neural response in the cortex is captured using fMRI and further stored as a vector. After collecting the fMRI data for a set of words, the inequalities in Eq. (3) can be constructed by using a similarity measure on the neural response vectors of word pairs. Here, we adopt Pearson correlation as the similarity measure. The details will be introduced in Section 5.1.

4 Data

4.1 Brain data

The fMRI data used in our experiments was recorded and preprocessed by Mitchell et al. (2008). It includes the recorded data of 9 subjects. To record the data, each of 60 concrete nouns was presented visually to each subject with a textual label and a simple line drawing. The subjects were asked to think about the properties of the objects indicated by the words during fMRI scanning. This procedure repeated 6 times, and the stimuli of the 60 nouns were presented in a random order in each run. More details about the data acquisition and preprocessing procedures can be found in Mitchell et al. (2008) and its supplement materials. Finally, an fMRI vector measuring the neural response at all voxels across the cortex was created for each word and each subject.

4.2 Behavioural data

The behavioural data collects human judgements on the semantic similarity between word pairs. The approach to behavioural data collection in our experiment is similar to the one used in the WordSim-353 dataset (Finkelstein et al., 2001). For the 60 concrete nouns used in Section 4.1, we obtained $C_{60}^2 = 1,770$ word pairs. Then, we asked 15 participants to score the semantic similarity of each word pair on a scale from 0 to 10, in which "0" signified that the two words were totally unrelated and "10" signified that the two words were highly related or had identical meanings. This collection procedure was conducted on the Amazon Mechanical Turk[1] crowdsourcing platform. We tested the average Spearman correlation coefficient among the scores given by different annotators and found that it was approximately 0.4873 with a p-value of 1.1e-02. After gathering the scores for all the word pairs, the highest and lowest scores for each word pair were discarded, and the average of the remaining 13 scores was calculated as the similarity score for each word pair[2].

To verify the reliability of the above data collection process, we also added 15 word pairs from the WordSim-353 dataset into our 1,770 word pairs during score collection. Then, we calculated the similarity scores of these 15 word pairs using the collected scores and compared them with the scores in the WordSim-353 dataset using Spearman correlation analysis. The correlation coefficient was 0.8451 with a p-value of $2.7e-04$. This high correlation verifies the reliability of our behavioural data collection.

5 Experiments

5.1 Correlations between brain activity and word vectors

We calculated the correlations between the fMRI vectors and the different types of word representations to investigate whether these representations can account for the brain activity and the functional selectivity of different cortex areas. We adopted the method of representational similarity analysis (Kriegeskorte et al., 2008) in our experiments. For a specific word representation, we calculated the cosine similarity for each word pair in a set of n words, resulting in a similarity vector with a total length of C_n^2. For the fMRI data[3], we constructed a similarity vector for each subject using the Pearson correlation coefficients between pairs of fMRI vectors (Anderson et al., 2013). Then, the 9 vectors of the 9 subjects were averaged to obtain an overall similarity vector in the fMRI space (Anderson et al., 2013). Finally, the Spearman rank correlation coefficient between the similarity vectors given by the fMRI data and each word representation was calculated together with a p-value for significance

[1] http://www.mturk.com/

[2] The behavioural data are available at http://home.ustc.edu.cn/~ypruan/work/emnlp2016/behaviour_data/

[3] Before using the fMRI data, we first regularized its mean value to 0 and variance to 1.

analysis. The p-value was calculated using a permutation test under a positive hypothesis with the word pair labels randomly shuffled 10,000 times. Empirically, two similarity vectors are considered to be correlated when $p < 0.05$, and they are considered significantly correlated when $p < 0.01$.

5.1.1 Word vectors

Three types of word representations, i.e., skip-gram word embeddings, visual semantic vectors, and primary visual features, were used in the correlation analysis. Some details about the acquisitions of these three word representations will be introduced in the following paragraphs.

Skip-gram word embeddings The Wikipedia text corpus[4], containing 130 million words, was adopted to train our skip-gram word embeddings, and the hierarchical softmax scheme was followed. The dimension of word embedding was 200. The window size, learning rate, and negative sampling number were set to 8, 0.05, and 8, respectively. The model was trained for one iteration using a single execution thread.

Visual semantic vectors On one hand, distributed word representations are usually learnt from text corpora. On the other hand, visual perception also contributes to semantic cognition according to some neuroscience research (Louwerse, 2011), and it has been utilized to complement the semantic representation learned from texts (Bruni et al., 2012). One approach to constructing visual semantic vectors is to first extract the low-level visual features from images and then convert them into higher-level semantic representations using the bag-of-visual-words (BoVW) (Grauman and Leibe, 2011) model. In our experiments, we built the BoVW representations from ImageNet (Deng et al., 2009) using the VSEM[5] toolkit. Due to coverage limitations, only 57 of the 60 concrete nouns in the fMRI data could be found in ImageNet[6] and each noun has approximately 1000 image samples. Similar to Anderson et al. (2013), we adopted the Scale Invariant Feature Transform (SIFT) (Lowe, 2004) to extract lower-level visual features; however, we did not use the "object" box to discriminate "object" and "context" areas during the extraction. Then, we clustered the SIFT features into 1000 classes to construct the visual vocabulary, and each image was divided into 8 regions. Thus, the BoVW representation of an image was a vector of 8000 dimensions. The BoVW vectors of all images in ImageNet corresponding to the same word were averaged to obtain the BoVW representation of that word. Finally, we transformed the BoVW representation matrix of the 57 nouns to nonnegative point-wise mutual information (PMI) association scores (Church and Hanks, 1990) to obtain the final visual semantic vectors.

Primary visual features As introduced in Section 4.1, a line drawing of each word was presented to subjects together with the textual label when collecting the fMRI data (Mitchell et al., 2008). This presentation led to neural responses in visual cortices that may be irrelevant to semantic representation. Because the receptive fields of simple cells in the primary visual cortex of mammalian brains can be modelled by Gabor functions (Marĉelja, 1980; Daugman, 1985), we adopted Gabor wavelets to extract the primary visual features from the line drawings of the 60 nouns and further analysed their correlations with fMRI data. The original resolution of the image stimuli used in Mitchell et al. (2008) was 500 x 500 pixels. These images were converted to 64 x 64 pixels after trimming the black borders and downsampling. The Gabor wavelet filter bank was designed using an open source tool (Haghighat et al., 2015). The number of scales and orientations were set to 5 and 8, respectively. Thus, we represented the primary visual features of each noun as a vector of 163,840 dimensions. The singular value decomposition (SVD) technique was employed to reduce the dimension of each vector to 60.

5.1.2 Correlation analysis at the whole-brain level

The fMRI recording measures the neural responses of more than 20,000 voxels across the cortex. To perform dimensionality reduction, we selected 500 voxels from all voxels for each subject according to

[4] http://mattmahoney.net/dc/enwik9.zip

[5] http://clic.cimec.unitn.it/vsem/

[6] The three missing words are *arm*, *eye* and *saw*.

word representation	rho (p-value)
skip-gram	0.0065 (4.0e-01)
BoVW	**0.3515** (0.0e-00)
Gabor	**0.3924** (0.0e-00)

Table 1: Spearman's rank correlation coefficients (**rho**) between different word representations and whole-brain fMRI data for 57 nouns and their corresponding p-values.

Lobe	Proportion (%)
frontal	5.89
temporal	6.96
parietal	10.13
occipital	58.40
other	18.62

Table 2: The proportions of the regional distributions of the 500 selected voxels.

the stability of the voxel responses across 6 runs of fMRI recordings. This selection strategy was the same as the one used in Mitchell et al. (2008) and Anderson et al. (2013). The correlation analysis followed the method described at the beginning of Section 5.1. Table 1 shows the results, where *skip-gram*, *BoVW*, and *Gabor* denote the skip-gram word embeddings, visual semantic vectors, and primary visual features introduced above, respectively.

As Table 1 shows, the visual semantic vectors and primary visual features are significantly correlated with the fMRI vectors at the whole-brain level; however, the skip-gram word embeddings are not correlated with the fMRI data. To investigate the reason for this lack of correlation, we analysed the distribution of the 500 selected voxels across the four brain lobes (i.e., frontal, temporal, parietal and occipital) using the automated anatomical labeling scheme (Tzourio-Mazoyer et al., 2002). From the results shown in Table 2, we can find that most of the selected voxels are located in the occipital lobe although it is the smallest of the four main lobes in the human brain. The occipital lobe occupies most of the anatomical area of the visual cortex and is considered to be the visual processing centre of the mammalian brain. This unbalanced distribution led to the conclusion that the semantic information related to skip-gram word embeddings is not well represented by the 500 selected voxels. Thus, an alternative strategy to select stable voxels at the brain lobe level for correlation analysis was necessary.

5.1.3 Correlation analysis at the brain lobe level

As an alternative approach, rather than selecting the 500 most stable voxels from the whole-brain data as in (Mitchell et al., 2008; Anderson et al., 2013), we selected the 100 most stable voxels at each of the four main brain lobes independently for this experiment. Then, the correlations between the fMRI vectors measuring different lobes and word representations were calculated and are shown in Table 3.

From this table, we can observe the association differences of different word representations with brain lobe level activities. First, the primary visual features (*Gabor*) are highly correlated with the occipital fMRI data and are uncorrelated with the other three lobes. This is reasonable considering that the primary visual cortex (V1) is located in the occipital lobe. Second, the skip-gram word embeddings are significantly correlated with the fMRI data of all brain lobes except the occipital lobe. Previous neuroscience research has revealed that the frontal, temporal, and parietal lobes all play important roles in semantic cognition, including high-level and abstract knowledge processing (Miller et al., 2002), integration of lexical information (Hagoort, 2005), speech comprehension (Hickok and Poeppel, 2007), and knowledge retrieval (Binder et al., 2009). This indicates that the skip-gram word embeddings can partly account for the semantic processing in the cortex and contain little visual information about words. Third, the visual sematic vectors (*BoVW*) are significantly correlated with all four brain lobes. It has been found that the temporal lobe plays a key role in both the formation of long-term visual memories (Smith and Kosslyn, 2007) and in the recognition of visual stimuli and objects (Chao et al., 1999; Kanwisher and Yovel, 2006). The parietal lobe is relevant to high-level vision and is part of the dorsal visual stream correlated with spatial cognition (Sack, 2009; Vannini et al., 2004). This indicates that the visual sematic vectors used in our experiment may contain not only low-level but also high-level and semantically related visual information.

	Frontal	Temporal	Parietal	Occipital
Skip-gram	**0.1450** (0.0e+00)	**0.1483** (0.0e+00)	**0.2317** (0.0e+00)	-0.0385 (9.4e-01)
BoVW	**0.0601** (8.2e-03)	**0.2053** (0.0e+00)	**0.2750** (0.0e+00)	**0.3120** (0.0e+00)
Gabor	-0.0823 (1.0e+00)	-0.0879 (1.0e+00)	0.0111 (3.4e-01)	**0.5116** (0.0e+00)

Table 3: Spearman's rank correlation coefficients (**rho**) between different word representations and the fMRI data at four main brain lobes and their corresponding p-values.

fMRI data	rho (p-value)
whole brain	**0.1266** (0.0e+00)
frontal lobe	0.0160 (2.5e-01)
temporal lobe	**0.0694** (1.7e-03)
parietal lobe	**0.0698** (1.6e-03)
occipital lobe	**0.0814** (4.0e-04)

Table 4: Spearman's rank correlation coefficients (**rho**) between the behaviour data and the fMRI data of different brain lobes.

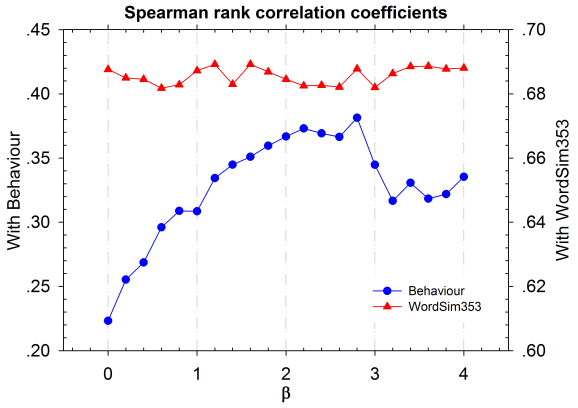


Figure 1: Spearman's rank correlation coefficients between the estimated word embeddings with different β values and the behavioural data of two datasets.

5.2 Correlations between brain activity and behavioural data

After analysing the correlation between brain activity and the three types of word vectors in the previous experiments, we further examined the correlations between brain activity and the behavioural data introduced in Section 4.2. Here, the behavioural data were used as the semantic ground truth to evaluate the semantic relevance of the brain activity and word embeddings. The results are shown in Table 4. In this subsection, the fMRI data at the whole-brain and brain lobe levels adopted the voxel selection strategies introduced in Sections 5.1.2 and 5.1.3, respectively. As Table 4 shows, the behavioural data are significantly correlated with the fMRI data of the whole brain and the occipital lobe, and they are also correlated with the fMRI data of the temporal and parietal lobes.

Furthermore, we utilized the SWE model introduced in Section 3.2 to explore the semantic relevance of brain activity by examining the performance of the estimated word embeddings after integrating brain activity related knowledge. The inequality set used in Eq. (3) was created using the fMRI data, where the similarity score s_{ij} was calculated as the Pearson correlation coefficient between the fMRI vectors of the i-th and the j-th words. For the 60 nouns (a total of 12 categories with 5 words in each category), we produced $12 \times 3 \times C_5^3 = 360$ intra-category inequalities and $3 \times C_{12}^3 = 660$ inter-category inequalities. To collect the inter-category inequalities, we first used the label words of each category and averaged the fMRI vectors of the 5 words belonging to each category to obtain the fMRI data for these label words. Then, the inter-category inequalities were produced from the triplets of these label words. The text corpus and parameter settings we used to train SWE were the same as those used for training the skip-gram word embeddings as described in Section 5.1.1. The penalty term β in Eq. (6) was tuned through experiments.

We evaluated the word embeddings estimated with brain activity constraints using the collected behavioural data for the 60 nouns and the WordSim-353 dataset. WordSim-353 is a behavioural dataset containing semantic similarity scores for 353 word-pairs (Finkelstein et al., 2001). We checked to ensure these word-pairs have no overlap with the 60 nouns used in our experiments. The purpose of using the WordSim-353 dataset is to explore the effects of utilizing the brain data of the 60 nouns on other words for which we had no brain data.

	60 nouns	WordSim353
skip-gram	0.2232	0.6876
SWE (whole brain)	0.3814	0.6878
SWE (frontal)	0.3173	0.6822
SWE (temporal)	0.3613	0.6890
SWE (parietal)	0.3516	0.6706
SWE (occipital)	0.3348	0.6803
JNNSE	0.3006	0.1795

Table 5: Spearman's rank correlation coefficients between different word embeddings and the behavioural data of the two datasets.

The performance of the word embeddings estimated using the SWE model and the whole-brain fMRI data are shown in Figure 1. In this figure, the SWE model becomes a conventional skip-gram model when $\beta = 0$. The correlation coefficient between the skip-gram word embeddings and the behavioural data of the 60 nouns was 0.2232. As β was increased, this correlation coefficient increased significantly. The maximum correlation efficient was 0.3814 when $\beta = 2.8$. This result implies that textual data and brain activity data can be used in a complementary fashion to derive word embeddings that are more consistent with human judgements. On one hand, semantic patterns may exist in neural representations that have not been fully captured by state-of-the-art word embeddings derived from text corpora. On the other hand, we can see that the variation of the correlation coefficients for the WordSim-353 dataset with different β values is small. This indicates that our SWE training didn't negatively affect the word embeddings without fMRI observations.

Furthermore, we produced ranking inequalities using the fMRI data measuring each brain lobe to estimate word embeddings under the SWE framework. The correlations between the learned word embeddings and the behavioural data of the two datasets were calculated and are shown in Table 5. For each SWE model in this table, the value of β was tuned to obtain the highest correlation on the 60 nouns. Comparing the correlation coefficients of the different models on the 60 nouns, we can see that the fMRI data at all brain lobes can contribute to learning more semantically related word embeddings using the SWE model. The improvement from using the fMRI data of the temporal lobe is the most significant among the four lobes, but the highest correlation coefficient is achieved when utilizing the fMRI data of whole brain.

Finally, we compared the performance of our SWE models with the JNNSE model proposed by Fyshe et al. (2014) on the two datasets. The word embeddings estimated by the JNNSE model utilized either fMRI or magnetoencephalography (MEG) measures of the 60 nouns. We adopted the best JNNSE word embeddings reported by the authors[7] for these comparisons, and the results are shown in the last row of Table 5[8]. As Table 5 shows, the performance of the JNNSE word embeddings on the WordSim-353 dataset is not as good as those of the skip-gram and SWE results. Examining the correlation coefficients on the 60 nouns with brain activity data, we can see that the JNNSE model achieves better performance than the skip-gram model, but is still below that of the SWE models. It should be noted that it is unfair to directly compare the SWE models and the JNNSE model because they used different training corpora and word embedding dimensions. Moreover, the β values of the SWE models were tuned to achieve the best performance on these 60 nouns. Here, the motivation behind introducing the JNNSE model as a reference is to help readers better understand the effects of integrating brain data into SWE training. These experimental results demonstrate that the SWE model is an effective model structure for integrating external knowledge into the estimation of word embeddings.

6 Conclusion

This study utilized word embeddings to investigate the semantic representations in brain activity as measured by fMRI. First, the functional selectivity of different cortex areas is explored by calculating the correlations between neural responses and three types of word vectors: skip-gram word embeddings, visual semantic vectors, and primary visual features.

[7] http://www.cs.cmu.edu/~afyshe/papers/acl2014/

[8] Because there were 32 word-pairs in the WordSim-353 dataset that were not covered by the vocabulary of the JNNSE word embeddings, the value 0.1795 in the last row of Table 5 was calculated using only 321 word-pairs.

Experimental results demonstrate the differences between the associations of different word vectors with brain-lobe-level brain activities. The skip-gram word embeddings are significantly correlated with the fMRI data of all brain lobes except the occipital lobe. Furthermore, we utilized behavioural data as the semantic ground truth to measure its relevance to brain activity. The SWE model was employed to explore the semantic relevance of brain activity by examining the performances of word embeddings after integrating brain-activity-related knowledge into their estimations. Experimental results show that whole-brain fMRI data are significantly correlated with human judgement with respect to semantic similarity. The correlations between the estimated word embeddings and the human-assigned similarity scores are effectively improved after integrating brain activity data into SWE training.

The experiments in this paper provide information about how semantic features correlate with brain activities, laying foundations for further investigations of higher-level semantic processing in the human brain. Furthermore, our experiments with SWE modelling show the potential of applying fMRI data to obtain better word embeddings. Although this approach is still far from being a practical engineering application due to issues such as the high costs and low signal-to-noise ratio of fMRI recordings and the diversity among individuals, it provides us with an alternative method for verifying the semantic relevance of brain activities and with evidence for recognizing the limitations of estimating word embeddings using only text corpora.

Acknowledgements

This work was supported in part by the Science and Technology Development of Anhui Province, China (Grant No. 2014z02006), the Fundamental Research Funds for the Central Universities (Grant No. WK2350000001) and the CAS Strategic Priority Research Program (Grant No. XDB02070006). The authors also want to thank Quan Liu for his help and wonderful suggestions during the experiments.

References

[Anderson et al.2013] Andrew J Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio, and Marco Baroni. 2013. Of words, eyes and brains: Correlating image-based distributional semantic models with neural representations of concepts. In *EMNLP*, pages 1960–1970.

[Binder et al.2009] Jeffrey R Binder, Rutvik H Desai, William W Graves, and Lisa L Conant. 2009. Where is the semantic system? a critical review and meta-analysis of 120 functional neuroimaging studies. *Cerebral Cortex*, 19(12):2767–2796.

[Bruni et al.2012] Elia Bruni, Jasper Uijlings, Marco Baroni, and Nicu Sebe. 2012. Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1219–1228. ACM.

[Carlson et al.2014] Thomas A Carlson, Ryan A Simmons, Nikolaus Kriegeskorte, and L Robert Slevc. 2014. The emergence of semantic meaning in the ventral temporal pathway. *Journal of cognitive neuroscience*, 26(1):120–131.

[Chao et al.1999] Linda L Chao, James V Haxby, and Alex Martin. 1999. Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects. *Nature neuroscience*, 2(10):913–919.

[Church and Hanks1990] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

[Csurka et al.2004] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. 2004. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, pages 1–2. Prague.

[Daugman1985] John G Daugman. 1985. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169.

[Deng et al.2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

[Fellbaum and others1998] Christiane Fellbaum et al. 1998. Wordnet: An electronic lexical database mit press. *Cambridge MA*.

[Finkelstein et al.2001] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In

Proceedings of the 10th international conference on World Wide Web, pages 406–414. ACM.

[Fyshe et al.2014] Alona Fyshe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2014. Interpretable semantic vectors from a joint model of brain-and text-based meaning. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2014, page 489. NIH Public Access.

[Grauman and Leibe2011] Kristen Grauman and Bastian Leibe. 2011. Visual object recognition. *Synthesis lectures on artificial intelligence and machine learning*, 5(2):1–181.

[Haghighat et al.2015] Mohammad Haghighat, Saman Zonouz, and Mohamed Abdel-Mottaleb. 2015. Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21):7905–7916.

[Hagoort2005] Peter Hagoort. 2005. On broca, brain, and binding: a new framework. *Trends in cognitive sciences*, 9(9):416–423.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

[Haxby et al.2001] James V Haxby, M Ida Gobbini, Maura L Furey, Alumit Ishai, Jennifer L Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430.

[Hickok and Poeppel2007] Gregory Hickok and David Poeppel. 2007. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393–402.

[Hinton et al.1986] Geoffrey E Hinton, James L Mcclelland, and David E Rumelhart. 1986. Distributed representations, parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations.

[Huth et al.2012] Alexander G Huth, Shinji Nishimoto, An T Vu, and Jack L Gallant. 2012. A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210–1224.

[Huth et al.2016] Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458.

[Kanwisher and Yovel2006] Nancy Kanwisher and Galit Yovel. 2006. The fusiform face area: a cortical region specialized for the perception of faces. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 361(1476):2109–2128.

[Kay et al.2008] Kendrick N Kay, Thomas Naselaris, Ryan J Prenger, and Jack L Gallant. 2008. Identifying natural images from human brain activity. *Nature*, 452(7185):352–355.

[Kriegeskorte et al.2008] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. 2008. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

[Liu et al.2015] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. *Proceedings of ACL, Beijing, China*.

[Louwerse2011] Max M Louwerse. 2011. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 3(2):273–302.

[Lowe2004] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Lund and Burgess1996] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

[Marĉelja1980] S Marĉelja. 1980. Mathematical description of the responses of simple cortical cells*. *JOSA*, 70(11):1297–1300.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

[Miller and Charles1991] George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

[Miller et al.2002] Earl K Miller, David J Freedman, and Jonathan D Wallis. 2002. The prefrontal cortex: categories, concepts and cognition. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 357(1424):1123–1136.

[Mitchell et al.2008] Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195.

[Naselaris et al.2009] Thomas Naselaris, Ryan J Prenger, Kendrick N Kay, Michael Oliver, and Jack L Gallant. 2009. Bayesian reconstruction of natural images from human brain activity. *Neuron*, 63(6):902–915.

[Nishimoto et al.2011] Shinji Nishimoto, An T Vu, Thomas Naselaris, Yuval Benjamini, Bin Yu, and

Jack L Gallant. 2011. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641–1646.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

[Ryali et al.2010] Srikanth Ryali, Kaustubh Supekar, Daniel A Abrams, and Vinod Menon. 2010. Sparse logistic regression for whole-brain classification of fMRI data. *NeuroImage*, 51(2):752–764.

[Sack2009] Alexander T Sack. 2009. Parietal cortex and spatial cognition. *Behavioural brain research*, 202(2):153–161.

[Sivic and Zisserman2003] Josef Sivic and Andrew Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE.

[Smith and Kosslyn2007] EE Smith and SM Kosslyn. 2007. *Cognitive Psychology: Mind and Brain*. Pearson Prentice Hall, Upper Saddle River, NJ.

[Turney et al.2010] Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

[Tzourio-Mazoyer et al.2002] Nathalie Tzourio-Mazoyer, Brigitte Landeau, Dimitri Papathanassiou, Fabrice Crivello, Olivier Etard, Nicolas Delcroix, Bernard Mazoyer, and Marc Joliot. 2002. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the mni mri single-subject brain. *Neuroimage*, 15(1):273–289.

[Vannini et al.2004] Patrizia Vannini, Ove Almkvist, Anders Franck, Tomas Jonsson, Umberto Volpe, Maria Kristoffersen Wiberg, Lars-Olof Wahlund, and Thomas Dierks. 2004. Task demand modulations of visuospatial processing measured with functional magnetic resonance imaging. *Neuroimage*, 21(1):58–68.

AMR Parsing with an Incremental Joint Model

Junsheng Zhou[†], **Feiyu Xu**[‡], **Hans Uszkoreit**[‡], **Weiguang Qu**[†], **Ran Li**[†] and **Yanhui Gu**[†]
† Language Information Processing and Social Computing Lab
School of Computer Science and Technology, Nanjing Normal University, China
{zhoujs, wgqu, gu}@njnu.edu.cn, liran3277@sina.com
‡ Language Technology Lab, DFKI, Germany
{feiyu, uszkoreit}@dfki.de

Abstract

To alleviate the error propagation in the traditional pipelined models for Abstract Meaning Representation (AMR) parsing, we formulate AMR parsing as a joint task that performs the two subtasks: concept identification and relation identification simultaneously. To this end, we first develop a novel component-wise beam search algorithm for relation identification in an incremental fashion, and then incorporate the decoder into a unified framework based on multiple-beam search, which allows for the bi-directional information flow between the two subtasks in a single incremental model. Experiments on the public datasets demonstrate that our joint model significantly outperforms the previous pipelined counterparts, and also achieves better or comparable performance than other approaches to AMR parsing, without utilizing external semantic resources.

1 Introduction

Producing semantic representations of text is motivated not only by theoretical considerations but also by the hypothesis that semantics can be used to improve many natural language tasks such as question answering, textual entailment and machine translation. Banarescu et al. (2013) described a semantics bank of English sentences paired with their logical meanings, written in Abstract Meaning Representation (AMR), which is rapidly emerging as an important practical form of structured sentence semantics. Recently, some literatures reported some promising applications of AMR. Pan et al. (2015) presented an unsupervised entity linking system with AMR, achieving the performance comparable to the supervised state-of-the-art. Liu et al. (2015) demonstrated a novel abstractive summarization framework driven by the AMR graph that shows promising results. Garg et al. (2016) showed that AMR can significantly improve the accuracy of a biomolecular interaction extraction system compared to only using surface- and syntax-based features. Mitra and Baral (2016) presented a question-answering system by exploiting the AMR representation, obtaining good performance.

Automatic AMR parsing is still in a nascent stage. Flanigan et al. (2014) built the first AMR parser, JAMR, based on a pipelined approach, which breaks down the whole task into two separate subtasks: concept identification and relation identification. Considering that node generation is an important limiting factor in AMR parsing, Werling et al. (2015) proposed an improved approach to the concept identification subtask by using a simple classifier over actions which generate these subgraphs. However, the overall architecture is still based on the pipelined model.

As a common drawback of the staged architecture, errors in upstream component are often compounded and propagated to the downstream prediction. The downstream components, however, cannot impact earlier decision. For example, for the verb "affect" in the example shown in Figure 1, there exist two possible concepts: "affect-01" and "affect-02". Comparatively, the first concept has more common use cases than the second one. But, when the verb "affect" is followed by the noun "ac-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 680–689,
Austin, Texas, November 1-5, 2016.

cent", it should evoke the concept "affect-02". Obviously, the correct concept choice for the verb "affect" should exploit a larger context, and even the whole semantic structure of the sentence, which is more probable to be unfolded at the downstream relation identification stage. This example indicates that it is necessary to allow for the interaction of information between the two stages.

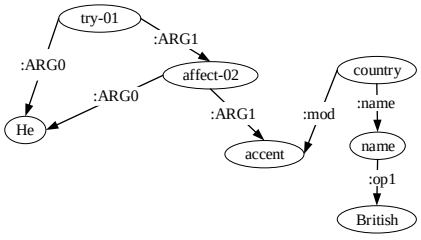


Figure 1: The AMR graph for the sentence "He tries to affect a British accent."

To address this problem, in this paper we reformulate this task as a joint parsing problem by exploiting an incremental parsing model. The underlying learning algorithm has shown the effectiveness on some other Natural Language Processing (NLP) tasks, such as dependency parsing and extraction of entity mentions and relations (Collins and Roark, 2004; Hatori et al., 2012; Li and Ji, 2014). However, compared to these NLP tasks, the AMR parsing is more challenging in that the AMR graph is more complicated. In addition, the nodes in the graph are latent.

One main challenge to search for concept fragments and relations incrementally is how to combine the two subtasks in a unified framework. To this end, we first develop a novel Component-Wise Beam Search (CWBS) algorithm for incremental relation identification to examine the accuracy loss in a fully incremental fashion compared to the global fashion in which a sequence of concept fragments derived from the whole sentence are required as input, as the MSCG algorithm in JAMR. Secondly, we adopt a segment-based decoder similar to the multiple-beam algorithm (Zhang and Clark, 2008b) for concept identification, and then incorporate the CWBS algorithm for relation identification into this framework, combining the two subtasks in a single incremental model. For parameter estimation, "violation-fixing" perceptron is adopted since it is designed specifically for inexact search in structured learning (Huang et al., 2012).

Experimental results show that the proposed joint framework significantly outperforms the pipelined counterparts, and also achieves better or comparable performance than other AMR parsers, even without employing external semantic resources.

2 Background

2.1 AMR Parsing Task

Nodes of an AMR graph are labeled with concepts, and edges are labeled with relations. Concepts can be English words ("He"), PropBank event predicates ("try-01", "affect-02"), or special keywords ("British"). For example, "affect-02" represents a PropBank roleset that corresponds to the first sense of "affect". According to (Banarescu et al., 2013), AMR uses approximately 100 relations. The rolesets and core semantic relations (e.g., ARG0 to ARG5) are adopted from the PropBank annotations in OntoNotes. Other semantic relations include "mode", "name", "time", "topic" and so on. The AMR guidelines provide more detailed descriptions.

2.2 The Pipelined Models for AMR Parsing

The AMR parser JAMR is a two-stage algorithm that first identifies concepts and then identifies the relations that obtain between these.

The concept identification stage maps spans of words in the input sentence to a sequence of concept graph fragments. Note that these graph fragments, in some cases, are subgraphs with multiple nodes and edges, not just one labeled concept node. The relation identification stage adds edges among the concept subgraph fragments identified in the first stage. JAMR requires the output subgraph $G =< V_G, E_G >$ should respect the following constraints:

(1) Simple: For any two vertices u and $v \in V_G$, E_G includes at most one edge between u and v.

(2) Connected: G must be weakly connected (every vertex reachable from every other vertex, ignoring the direction of edges).

(3) Deterministic: For each node $u \in V_G$, and for each label $l \in \{\text{ARG0}, \ldots, \text{ARG5}\}$, there is at

most one outgoing edge in E_G from u with label l.

To find a maximum spanning AMR graph, JAMR proposed a two-step approach[1] . First, a graph that ignores constraint (3) but respects the others was created, by searching for the maximum spanning connected subgraph from an edge-labeled, directed graph representing all possible relations between the identified concepts; Second, a Lagrangian relaxation was adopted to iteratively adjust the edge scores so as to enforce constraint (3).

In order to train the parser, JAMR built an automatic aligner that uses a set of rules to greedily align concepts to spans of words in the training data to generate an alignment table.

3 Algorithms

Based on the hypothesis that concept identification and relation identification are interrelated, we propose to jointly perform the two subtasks in a single model. To this end, we present an incremental model for AMR parsing. Evidence from psycholinguistic research also suggests that human language comprehension is incremental. Comprehenders do not wait until the end of the sentence before they build a syntactic or semantic representation for the sentence.

However, the challenges of successfully applying the incremental joint model to this problem formulation are: 1) how can we design an effective decoding algorithm for identifying the relations between the nodes in an incremental fashion, given a partial sequence of spans, i.e., a partial sequence of gold-standard concept fragments; 2) further, if given a sentence, how can we design an incremental framework to perform concept identification and relation identification simultaneously. In the following subsections we introduce our solutions to these challenges in detail.

3.1 An Incremental Decoding Algorithm for Relation Identification

We define the relation identification problem as finding the highest scoring graph y from all possible outputs given a sequence of concept fragments c:

$$F(c) = \arg\max_{Gen(c)} Score(y) \quad (1)$$

where $Gen(c)$ denotes the set of possible AMR graph for the input c. The score of an output parse y is defined to be decomposed by edges, and with a linear model:

$$Score(y) = \sum_{e \in E_y} w^T \cdot \varphi(e) \quad (2)$$

where $\varphi(e)$ is the feature vector over the edge e, and w is weight vector of the model.

The AMR graph is a directed graph that respects three constraints (see section 2.2) and has a node marked as the focus node. Obviously, finding such a maximum spanning graph in AMR parsing in fact carries more complexities than that of maximum spanning tree (MST) decoding for syntactic parsing. Especially, performing the task incrementally is substantially harder than doing it non-incrementally. In both cases, parsing is in general intractable and we provide an approximate inference algorithm to make these cases tractable.

Inspired by the graph-based dependency parser under the framework of beam-search, which yields a competitive performance compared to the exact-search-based counterpart (Zhang and Clark, 2008a), we develop a CWBS algorithm for the relation identification task.

Basically, the decoder works incrementally, building a state item (i.e. a partial AMR graph) fragment by fragment. When each concept fragment is processed, edges are added between the current concept fragment and its predecessors. However, how to treat its predecessors is a difficult problem. In our experiments, we found that if we consider every preceding concept fragment to the left of the current fragment in a right-to-left order in the search process, the decoder suffers from low efficiency and poor performance. Unlike the beam-search for dependency parsing, which can greatly reduce the search space by exploiting the projectivity property of the dependency tree (Covington, 2001; Zhang and Clark, 2008a), this naive search process in this context inevitably leads to huge search space, and furthermore is difficult to guarantee the connectivity of

[1] In this paper, we refer to this two-step approach for relation identification as MSCG algorithm.

output graph. Instead, we propose a component-wise beam search scheme, which can not only alleviate much noisy partial candidate, but also ensure that the final output graph is *connected*.

Algorithm 1 shows the pseudocode for the complete procedure of the decoder. In a nutshell, the algorithm builds the AMR graph in one left-to-right pass over the sequence of concept fragments. Beam search is applied by keeping the B-best[2] items in the agenda at each processing stage, according to the scores of partial graph up to the current concept fragment. Lets take an illustrative diagram to demonstrate the procedure (see Figure 2). When appending the current concept fragment to the left partial graph to extend it, we just need to consider the relations between current concept and each preceding connected component. However, even at this single step, picking B-best extended partial graphs is still a difficult task due to the large combination space. Here, we adopt an effective *nested beam search strategy* at this step. In other words, edges are added between the current concept fragment and its preceding connected components by iterating through these components in a right-to-left order[3] using an *inner beam-search*. When examining the edges between the current concept fragment and some preceding component, four elementary actions are used:

(1) **SHIFT** (lines 12-14): Add only current concept to the partial graph.

(2) **LEFT-ARC** (lines 16-19): Add current concept and a highest-scoring edge from a node in the current concept to a node in some preceding connected component to the partial graph.

(3) **RIGHT-ARC** (lines 21-24): Add current concept and a highest-scoring edge from a node in some preceding connected component to a node in current concept to the partial graph.

(4) **LEFT & RIGHT-ARCS** (lines 26-27): Add current concept and highest-scoring left arc and right arc to the partial graph.

The first three actions are similar in form to those in the Arc-Standard algorithm for transition-based dependency parsing (Nivre, 2008; Zhang and Clark, 2008a). The last one is defined to cope with the cases where there may be multiple parents for some nodes in an AMR graph. Note that the "SHIFT" action does not add any edges. This operation is particularly necessary because the partial graphs are not always connected during the search process. In our experiments, we also found that the number of connected components during search process is relatively small, which is generally less than 6. It is important to note that, in order to guarantee the output graph *connected*, when the last concept fragment is encountered, the "SHIFT" action is skipped (see line 10 in Algorithm 1), and the other three 'arc' actions will add edges to connect the last concept fragment with all preceding connected components to yield a connected graph.

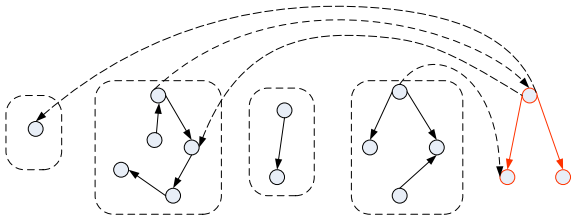


Figure 2: An illustrative diagram for CWBS algorithm. Each dotted box corresponds to a connected component in the partial graph, each of which consists one or multiple concept fragments. The rightmost subgraph corresponds to the current concept fragment.

For purpose of brevity, we introduce some functional symbols in Algorithm 1. Function CalEdgeScores($state, c_i$) calculates the scores of all candidate edges between the nodes in current concept fragment c_i and the nodes in the partial graph in state covering $(c_1, c_2, \ldots, c_{i-1})$. For computing the scores of edges, we use the same features as JAMR (refer to Flanigan et al. (2014) for more details). Function FindComponents($state$) returns all connected components $(p_1, p_2, \ldots, p_m)$ in the partial graph in $state$, sorted by the maximum end position of spans including in every component. The AddItem function adds the current concept fragment and left/right arc to the partial graph. Function AppendItem($buf, item$) inserts the partial graph $item$ into buf by its score. Functions GetMaxLeftEdge(c_i, p_j) and

[2]The constant B denotes the beam size.

[3]The right-to-left order reflects the principle of local priority.

Algorithm 1 The incremental decoding algorithm for relation identification.

Input: A sequence of concept fragments $(c_1, c_2, \ldots, c_n)$
Output: Best AMR graph including $(c_1, c_2, \ldots, c_n)$

$agenda \leftarrow$ {Empty-graph}
for $i \leftarrow 1 \ldots n$ **do**
  for $state$ in $agenda$ **do**
    CalEdgeScores($state$, c_i)
    $(p_1, p_2, \ldots, p_m) \leftarrow$ FindComponents($state$)
    $innerAgenda \leftarrow state$
    for $j \leftarrow m \ldots 1$ **do**
      $buf \leftarrow NULL$
      for $item$ in $innerAgenda$ **do**
        if $i < n$ **then**
          //Add only c_i to the item
          $newitem \leftarrow item$
          AddItem($newitem, c_i$)
          AppendAgenda($buf, newitem, i, n$)
        // Add a left arc from c_i to p_j to the $item$
        $newitem \leftarrow item$
        $le \leftarrow$ GetMaxLeftEdge(c_i, p_j)
        AddItem($newitem, c_i, le$)
        AppendAgenda($buf, newitem, i, n$)
        //Add a right arc from p_j to c_i the $item$
        $newitem \leftarrow item$
        $re \leftarrow$ GetMaxRightEdge(p_j, c_i)
        AddItem($newitem, c_i, le$)
        AppendAgenda($buf, newitem, i, n$)
        //Add both left and right arc to the $item$
        AddItem($item, c_i, le, re$)
        AppendAgenda($buf, item, i, n$)
      $innerAgenda \leftarrow B$-best(buf)
  $agenda \leftarrow innerAgenda$
return $agenda$[0]
function AppendAgenda($buf, item, i, n$)
  //parameter n represents the terminal position
  if $i = n$ **then**
    CalRootFeatures($item$)
  AppendItem($buf, item$)

GetMaxRightEdge(p_j, c_i) pick the highest-scoring left-arc and right-arc linking current fragment c_i and the connected component p_j by the scores returned from the CalEdgeScores function, respectively.

Finally, the function CalRootFeatures(g) first computes the scores for all nodes in the output graph g by treating them as the candidate root respectively, and then pick the node with the highest score as the focus node of the graph. When computing the score for each candidate node, similar to JAMR, two types of features were used: the concept of the node, and the shortest dependency path from a word in the span to the root of the dependency tree.

The time complexity of the above algorithm is $O(MB^2n)$, where M is the maximum number of connected components during search, B is beam size and n is the number of concept fragments. It is linear in the length of sequence of concept fragments. However, the constant in the O is relatively large. In practice, the search space contains a large number of invalid partial candidates. Therefore, we introduce three partial output pruning schemes which are helpful in reducing search space as well as making the input for parameter update less noisy.

Firstly, we limit the number of children and parents of every node. By observing the training data, we set the maximum numbers of children and parents of every node as 7 and 4, respectively. Secondly, due to the fact that all frame arguments ARG0-ARG5 are derived from the verb framesets, the edges with label $l \in \{ARG0, \ldots, ARG5\}$ that do not outgo from a verb node will be skipped.

Finally, consider the determinism constraint (as illustrated in section 2.2) that should be satisfied by an AMR representation. When one edge has the same label $l \in \{ARG0, \ldots, ARG5\}$ as one of edges outgoing from the same parent node, this edge will also be skipped. Obviously, this type of pruning can enforce the *determinism* constraint for every decoding output.

3.2 Joint Decoding for Concept Identifica-tion and Relation Identification

In this section, we further consider the joint decoding problem for a given sentence x, which maps the sentence x to an output AMR graph y. The objective function for the joint decoding is as follows:

$$\hat{y} = \arg\max_{y' \in Gen(x)} (w^T \cdot \phi(x, y') + w^T \cdot \mathbf{f}(y')) \quad (3)$$

where the first term is to calculate the score over all concept fragments derived from the words in the sentence x, and the second one is to calculate the score over all edges linking the concept fragments. Maximizing Equation (3) amounts to concurrently maximizing the score over the concept fragments and the score over the edges. Admittedly, the joint decoding problem is more intricate and in general

intractable. Therefore, we use a beam-search-based incremental decoder for approximate joint inference during training and testing.

In order to combine the two subtasks in a unified framework, we first relax the exact-search for concept identification in JAMR by beam search, resulting in a segment-based decoder similar to the multiple-beam algorithm in (Zhang and Clark, 2008b; Li and Ji, 2014), and then incorporate the CWBS algorithm for relation identification (as depicted in section 3.1) into this framework, which provides a natural formulation for combining the two subtasks in a single incremental model.

Algorithm 2 shows the joint decoding algorithm. In short, during performing joint decoding incrementally for the input sentence, for each word index i in the input sentence, it maintains a beam for the partial graphs whose last segments end at the i-th word, which is denoted as $agendas[i]$ in the algorithm. When the i-th word is processed, it either triggers concepts starting from this word by looking up the alignment table generated from the training data, or evokes no concept (we refer to this type of words as *function words*). If the current word triggers multiple concepts, we first append each candidate concept to the partial graphs in the beam $agendas[i-1]$, by using a component-wise beam search way (see section 3.1), and then pick B-best extended partial graphs by exploiting the features from *both the concept level* and *relation level* to compute the overall scores.

In particular, judging whether a word is a *function word* is an important and difficult task. For example, the word "*make*" corresponds to multiple candidate concepts in the alignment table, such as "*make-01*" and "*make-02*". However, it can also act as a functional word in some cases. To resolve the judgement problem, we view each word as a *function word* and a *non-function word* at the same time to allow them to compete against each other by their scores. For instance, for the i-th word, this is done by combining all partial graphs in the beam $agendas[i-1]$ with those in the beam $agendas[i]$ to select B-best items and then record them in $agendas[i]$, which is represented as the Union function in Algorithm 2.

After all words are processed, the highest-scoring graph in the beam corresponding to the terminal position of the sentence is selected as the output.

Algorithm 2 The joint decoding algorithm.

Input: Input sentence $x = (w_1, w_2, \ldots, w_n)$
Output: Best AMR graph derived from x

```
agendas[0] ← ∅
last ← Scan(x)
for i ← 1 ... n do
    list ← Lookup(x, i)
    if list.size > 0 then
        preAgenda ← agendas[i − 1]
        for cf ∈ list do
            end ← i + cf.size − 1
            if preAgenda.size = 0 then
                g ← Graph.empty
                CalConceptFeatures(g, cf)
                AppConcept(agendas, end, g, cf, last)
            else
                for item ∈ preAgenda do
                    g ← item
                    CalConceptFeatures(g, cf)
                    AppConcept(agendas, end, g, cf, last)
        Union(agendas, i, i − 1)
    else
        agendas[i] ← agendas[i − 1]
bestGraph ← agendas[last][0]
return bestGraph
```

In algorithm 2, function Scan(x) is used to search the terminal position corresponding to the last concept fragment in the sentence x, which will be passed as a parameter to the function AppConcept. The Scan function can be efficiently implemented by calling the function Lookup in a right-to-left order. Function Lookup(x, i) maps a sequence of words starting from the index i in sentence x to a set of candidate concept fragments, by looking up the alignment table that was generated from the training data. The alignments are accomplished using an aligner from JAMR. Motivated by Werling et al. (2015), we also adopt two additional actions to generate the candidate concept fragments: LEMMA and VERB. The action LEMMA is executed by using the lemma of the source token as the generated node title, and the action VERB is to find the most similar verb in PropBank based on Jaro-Winkler distance, and adopt its most frequent sense.

Function CalConceptFeatures(g, cf) calculates the feature vector for the candidate concept fragment cf and the partial graph g, using the features

defined in Table 1. Among them, features 1-4 are from JAMR. Additional features 5-16 aim to capture the association between the current concept and the context in which it appears. Function AppConcept($agendas, end, g, cf, last$) appends the current concept cf to the partial graph g, and then inserts the extended partial graph into $agendas[end]$. Note that when the parameter end equals to the parameter $last$, this function will call the function CalRootFeatures to select the focus node, as illustrated in Algorithm 1.

	Name	Description
1	Fragment given words	Relative frequency estimates of the probability of a concept fragment given the span of words.
2	Span length	The length of the span.
3	NER	1 if the span corresponds to a named entity, 0 otherwise.
4	Bias	1 for any concept fragment from the alignment table, 0 otherwise.
5	c	c represents the current concept label, w represents the current words, lem represents the current lemmas, pos represents the current POS tags. w_{-1} denotes the first word to the left of current word, w_{+1} denotes the first word to the right of current word, and so on.
6	$c + w$	
7	$c + lem$	
8	$c + pos$	
9	$c + w_{-1}$	
10	$c + w_{+1}$	
11	$c + pos_{-1}$	
12	$c + pos_{+1}$	
13	$c + w_{-2}$	
14	$c + w_{+2}$	
15	$c + pos_{-2}$	
16	$c + pos_{+2}$	

Table 1: Features associated with the concept fragments.

3.3 Violation-Fixing Perceptron for Training

Online learning is an attractive method for the structured learning since it quickly converges within a few iterations (Collins, 2002). Particularly, Huang et al. (2012) establish a theoretical framework called "violation-fixing perceptron" which is tailored for structured learning with inexact search and has provable convergence properties. Since our incremental decoding for AMR parsing is an approximate inference, it is very natural to employ violation-fixing perceptron here for AMR parsing training.

Specifically, we use an improved update method "max-violation" which updates at the worst mistake, and converges much faster than early update with similar or better accuracy. We adopt this idea here as follows: decode the whole sentence, and find the word index i^* where the difference between the candidate partial graph and gold-standard one is the biggest. Only part of the graph ending at the word index i^* is used to calculate the weight update, in order to account for search errors.

To reduce overfitting, we used averaged parameters after training to decode test instances in our experiments. The resulting model is called averaged perceptron (Collins, 2002).

Additionally, in our training algorithms, the implementation of the oracle function is rela-tively straightforward. Specifically, when the i-th span is processed in the incremental parsing process, the partial gold-standard AMR graph up to the i-th span consists of the edges and nodes that appear before the end position of the i-th span, over which the gold-standard feature vectors are calculated.

4 Experiments

4.1 Dataset and Evaluation Metric

Following previous studies on AMR parsing, our experiments were performed on the newswire sections of LDC2013E117 and LDC2014T12, and we also follow the official split for training, development and evaluation. Finally, we also show our parsers performance on the full LDC2014T12 dataset. We evaluate the performance of our parser using Smatch v2.0 (Cai and Knight, 2013), which counts the precision, recall and F1 of the concepts and relations together.

4.2 Development Results

Generally, larger beam size will increase the computational cost while smaller beam size may reduce the performance. As a tradeoff, we set the beam size as 4 throughout our experiments. Figure 3 shows the training curves of the averaged violation-fixing perceptron with respect to the performance on the both development sets. As we can see the curves converge very quickly, at around iteration 3.

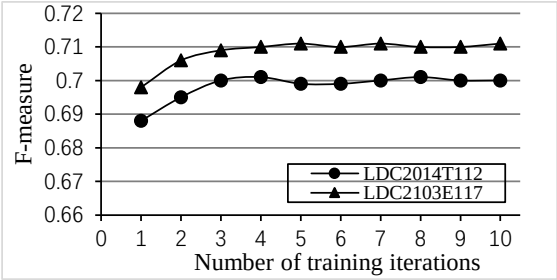


Figure 3: Learning curves on development sets.

Dataset	System	P	R	F1
LDC2013E117	MSCG	.85	.77	.81
	CWBS	.85	.78	.81
LDC2014T12	MSCG	.84	.77	.80
	CWBS	.84	.77	.80

Table 2: Results of two different relation identification algorithms.

4.3 Incremental Relation Identification Performance

Before performing joint decoding, we should first verify the effectiveness of our incremental algorithm CWBS. The first question about CWBS is whether the component-wise search is a valid scheme for deriving the gold-standard AMR graph given the sequence of gold-standard concepts. Therefore, we first implement an oracle function by performing the incremental component-wise search for each fragment sequence c to get a "pseudo-gold" graph $G_c^{'}$; Then we compare with gold-standard AMR graph G_c . On the training data of LDC2013E117 and LDC2014T12, we respectively got an overall 99.6% and 99.7% F-scores for all $< G_c^{'}, G_c >$ pairs, which indicates that our component-wise search is an effective incremental search scheme.

Further, we train a perceptron model using the max-violation update to approximate the oracle search procedure. As shown in Table 2, our incremental algorithm CWBS achieves almost the same performance as the non-incremental algorithm MSCG in JAMR, using the same features as MSCG. The results indicate that CWBS is a competitive alternative to MSCG.

4.4 Joint Model vs. Pipelined Model

In this section, we compare the overall performance of our joint model to the pipelined model, JAMR[4]. To give a fair comparison, we first implemented ***system 1*** only using the same features (i.e., features 1-4 in Table 1) as JAMR for concept fragments. Table 3 gives the results on the two datasets. In terms of F-measure, we gain a 6% absolute improvement, and a 5% absolute improvement over the results of JAMR on the two different experimental setups respectively.

Next, we implemented ***system 2*** by using more lexical features to capture the association between concept and the context (i.e., features 5-16 in Table 1). Intuitively, these lexical contextual features should be helpful in identifying concepts in parsing process. As expected, the results in Table 3 show that we gain 3% improvement over the two different datasets respectively, by adding only some additional lexical features.

Dataset	System	P	R	F1
LDC2013E117	JAMR(fixed)	.67	.58	.62
	System 1	.72	.65	.68
	System 2	.73	.69	.71
LDC2014T12	JAMR(fixed)	.68	.59	.63
	System 1	.74	.63	.68
	System 2	.73	.68	.71

Table 3: Comparison between our joint approaches and the pipelined counterparts.

Dataset	System	P	R	F1
LDC2013E117	CAMR*	.69	.67	.68
	CAMR	.71	.69	.70
	Our approach	**.73**	**.69**	**.71**
LDC2014T12	CAMR*	.70	.66	.68
	CAMR	.72	.67	.70
	CCG-based	.67	.66	.66
	Our approach	**.73**	**.68**	**.71**

Table 4: Final results of various methods.

4.5 Comparison with State-of-the-art

We give a comparison between our approach and other state-of-the-art AMR parsers, including CCG-based parser (Artzi et al., 2015) and dependency-based parser (Wang et al., 2015b). For comparison

[4]We use the latest, fixed version of JAMR, available at `https://tiny.cc/jamr`.

purposes, we give two results from two different versions of dependency-based AMR parser[5]: CAMR* and CAMR. Compared to the latter, the former denotes the system that does not use the extended features generated from the semantic role labeling system, word sense disambiguation system and so on, which is directly comparable to our system.

From Table 4 we can see that our parser achieves better performance than other approaches, even without utilizing any external semantic resources.

We also evaluate our parser on the full LDC2014T12 dataset. We use the training/development/test split recommended in the release: 10,312 sentences for training, 1,368 sentences for development and 1,371 sentences for testing. For comparison, we include the results of JAMR, CAMR*, CAMR and SMBT-based parser (Pust et al., 2015), which are also trained on the same dataset. The results in Table 5 show that our approach outperforms CAMR*, and obtains comparable performance with CAMR. However, our approach achieves slightly lower performance, compared to the SMBT-based parser, which adds data and features drawn from various external semantic resources.

Dataset	System	P	R	F1
LDC2014T12	JAMR(fixed)	.64	.53	.58
	CAMR*	.68	.60	.64
	CAMR	.70	.62	.66
	SMBT-based	-	-	**.67**
	Our approach	.70	.62	.66

Table 5: Final results on the full LDC2014T12 dataset.

5 Related Work

Our work is motivated by JAMR (Flanigan et al., 2014), which is based on a pipelined model, resulting in a large drop in overall performance when moving from gold concepts to system concepts.

Wang et al. (2015a) uses a two-stage approach; dependency parses are modified by executing a sequence of actions to resolve dis-crepancies between dependency tree and AMR structure. Goodman et al. (2016) improves the transition-based parser with the imitation learning algorithms, achieving almost the same performance as that of Wang et al. (2015b), which exploits the extended features from additional trained analysers, including co-reference and semantic role labelers. Artzi et al. (2015) introduces a new CCG grammar induction algorithm for AMR parsing, combined with a factor graph to model non-compositional phenomena. Pust et al. (2015) adapts the SBMT parsing framework to AMR parsing by designing an AMR transformation, and adding external semantic resources. More recently, Damonte et al. (2016) also presents an incremental AMR parser based on a simple transition system for dependency parsing. However, compared to our parser, their parser cannot parse non-projective graphs, resulting in a limited coverage.

Our work is also inspired by a new computational task of incremental semantic role labeling, in which semantic roles are assigned to incomplete input (Konstas et al., 2014).

6 Conclusions and Future Work

In this paper, we present a new approach to AMR parsing by using an incremental model for performing the concept identification and relation identification jointly, which alleviates the error propagation in the pipelined model.

In future work, we plan to improve the parsing performance by exploring more features from the coreference resolution, word sense disambiguation system and other external semantic resources. In addition, we are interested in further incorporating the incremental semantic role labeling into our incremental framework to allow bi-directional information flow between the two closely related tasks.

Acknowledgments

This research is supported by projects 61472191, 61272221 under the National Natural Science Foundation of China, projects 14KJB520022, 15KJA420001 under the Natural Science Research of Jiangsu Higher Education Institutions of China, and partially supported by the German Federal Ministry of Education and Research (BMBF) through the project ALL SIDES (01IW14002) and BBDC (contract 01IS14013E). We would also like to thank the insightful comments from the three anonymous reviewers.

[5]The code is available at `https://github.com/Juicechuan/AMRParsing`

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proc. of EMNLP*, pages 1699–1710.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, , and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proc. of the Linguistic Annotation Workshop and Interoperability with Discourse*.

Shu Cai and Kevin Knight. 2013. Smatch: an Evaluation Metric for Semantic Feature Structures. In *Proc. of ACL*, pages 748–752.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proc. of ACL*, pages 111–118.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron algorithms. In *Proc. of EMNLP*, pages 1–8.

Michael A. Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proc. of ACM Southeast Conference*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2016. An Incremental Parser for Abstract Meaning Representation. arXiv preprint at arXiv:1608.06111.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proc. of ACL*, pages 1426–1436.

Sahil Garg, Aram Galstyan, Ulf Hermjakob, and Daniel Marcu. 2016. Extracting Biomolecular Interactions Using Semantic Parsing of Biomedical Text. In *Proc. of AAAI*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise Reduction and Targeted Exploration in Imitation Learning for Abstract Meaning Representation Parsing. In *Proc. of ACL*, pages 1–11.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2012. Incremental Joint Approach to Word Segmentation, POS Tagging, and Dependency Parsing in Chinese. In *Proc. of ACL*, pages 1045–1053.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured Perceptron with Inexact Search. In *Proc. of HLT-NAACL*, pages 142–151.

Ioannis Konstas, Frank Keller, Vera Demberg, and Mirella Lapata. 2014. Incremental Semantic Role Labeling with Tree Adjoining Grammar. In *Proc. of EMNLP*, pages 301–312.

Qi Li and Heng Ji. 2014. Incremental Joint Extraction of Entity Mentions and Relations. In *Proc. of ACL*, pages 402–412.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Proc. of NAACL*, pages 1086–1077.

Arindam Mitra and Chitta Baral. 2016. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. In *Proc. of AAAI*.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised Entity Linking with Abstract Meaning Representation. In *Proc. of NAACL*, pages 1130–1139.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation. In *Proc. of EMNLP*, pages 1143–1154.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. A Transition-based Algorithm for AMR Parsing. In *Proc. of NAACL*, pages 366–375.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. Boosting Transition-based AMR Parsing with Re-fined Actions and Auxiliary Analyzers. In *Proc. of ACL*, pages 857–862.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust Subgraph Generation Improves Abstract Meaning Representation Parsing. In *Proc. of ACL*, pages 982–991.

Yue Zhang and Stephen Clark. 2008a. A Tale of Two Parsers: Investigating and Combining Graph-Based And transition-Based Dependency Parsing Using Beam-search. In *Proc. of EMNLP*, pages 562–571.

Yue Zhang and Stephen Clark. 2008b. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In *Proc. of ACL*, pages 888–896.

Identifying Dogmatism in Social Media: Signals and Models

Ethan Fast and **Eric Horvitz**
ethaen@stanford.edu, horvitz@microsoft.com

Abstract

We explore linguistic and behavioral features of dogmatism in social media and construct statistical models that can identify dogmatic comments. Our model is based on a corpus of Reddit posts, collected across a diverse set of conversational topics and annotated via paid crowdsourcing. We operationalize key aspects of dogmatism described by existing psychology theories (such as over-confidence), finding they have predictive power. We also find evidence for new signals of dogmatism, such as the tendency of dogmatic posts to refrain from signaling cognitive processes. When we use our predictive model to analyze millions of other Reddit posts, we find evidence that suggests dogmatism is a deeper personality trait, present for dogmatic users across many different domains, and that users who engage on dogmatic comments tend to show increases in dogmatic posts themselves.

1 Introduction

> "I'm supposed to trust the opinion of a MS minion? The people that produced Windows ME, Vista and 8? They don't even understand people, yet they think they can predict the behavior of new, self-guiding AI?" *–anonymous*
>
> "I think an AI would make it easier for Patients to confide their information because by nature, a robot cannot judge them. Win-win? :D"' *–anonymous*

Dogmatism describes the tendency to lay down opinions as incontrovertibly true, without respect for conflicting evidence or the opinions of others (Oxford Dictionary, 2016). Which user is more dogmatic in the examples above? This question is simple for humans. Phrases like "they think" and "they don't even understand," suggest an intractability of opinion, while "I think" and "win-win?" suggest the opposite. Can we train computers to draw similar distinctions? Work in psychology has called out many aspects of dogmatism that can be modeled computationally via natural language, such as over-confidence and strong emotions (Rokeach, 1954).

We present a statistical model of dogmatism that addresses two complementary goals. First, we validate psychological theories by examining the predictive power of feature sets that guide the model's predictions. For example, do linguistic signals of certainty help to predict a post is dogmatic, as theory would suggest? Second, we apply our model to answer four questions:

R1: What kinds of topics (e.g., guns, LGBT) attract the highest levels of dogmatism?

R2: How do dogmatic beliefs cluster?

R3: How does dogmatism influence a conversation on social media?

R4: How do other user behaviors (e.g., frequency and breadth of posts) relate to dogmatism?

We train a predictive model to classify dogmatic posts from Reddit, one of the most popular discussion communities on the web.[1] Posts on Reddit capture discussion and debate across a diverse set of domains and topics – users talk about everything from climate change and abortion, to world news and relationship advice, to the future of artificial intelligence. As a prerequisite to training our model, we have created a corpus of 5,000 Reddit posts annotated with levels of dogmatism, which we are releasing to share with other researchers.

[1] http://www.reddit.com

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 690–699,
Austin, Texas, November 1-5, 2016.

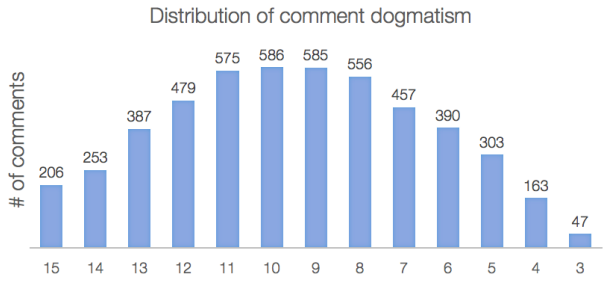


Figure 1: We crowdsourced dogmatism labels for 5000 comments. The distribution is slightly skewed towards higher levels of dogmatism. For example, crowdworkers unanimously labeled 206 comments as highly dogmatic ($5 \times 3 = 15$), but only 47 as minimally dogmatic ($1 \times 3 = 3$).

Using the model, we operationalize key domain-independent aspects of psychological theories of dogmatism drawn from the literature. We find these features have predictive power that largely supports the underlying theory. For example, posts that use less confident language tend to be less dogmatic. We also discover evidence for new attributes of dogmatism. For example, dogmatic posts tend not to verbalize cognition, through terms such as "I think," "possibly," or "might be."

Our model is trained on only 5,000 annotated posts, but once trained, we use it to analyze millions of other Reddit posts to answer our research questions. We find a diverse set of topics are colored by dogmatic language (e.g., people are dogmatic about religion, but also about LGBT issues). Further, we find some evidence for dogmatism as a deeper personality trait – people who are strongly dogmatic about one topic are more likely to express dogmatic views about others as well. Finally, in conversation, we discover that one user's dogmatism tends to bring out dogmatism in their conversational partner, forming a vicious cycle.

2 Dogmatism data

Posts on Reddit capture debate and discussion across a diverse set of topics, making them a natural starting point for untangling domain-independent linguistic features of dogmatism.

Data collection. Subreddits are sub-communities on Reddit oriented around specific interests or topics, such as *technology* or *politics*. Sampling from Reddit as a whole would bias the model towards the most commonly discussed content. But by sampling posts from individual subreddits, we can control the kinds of posts we use to train our model. To collect a diverse training dataset, we have randomly sampled 1000 posts each from the subreddits *politics*, *business*, *science*, and *AskReddit*, and 1000 additional posts from the Reddit frontpage. All posts in our sample appeared between January 2007 and March 2015, and to control for length effects, contain between 300 and 400 characters. This results in a total training dataset of 5000 posts.

Dogmatism annotations. Building a useful computational model requires labeled training data. We labeled the Reddit dataset using crowdworkers on Amazon Mechanical Turk (AMT), creating the first public corpus annotated with levels of dogmatism. We asked crowdworkers to rate levels of dogmatism on a 5-point Likert scale, as supported by similar annotation tasks in prior work (Danescu-Niculescu-Mizil et al., 2013). Concretely, we gave crowdworkers the following task:

> Given a comment, imagine you hold a well-informed, different opinion from the commenter in question. We'd like you to tell us how likely that commenter would be to engage you in a constructive conversation about your disagreement, where you each are able to explore the other's beliefs. The options are:
>
> **(5)**: It's unlikely you'll be able to engage in any substantive conversation. When you respectfully express your disagreement, they are likely to ignore you or insult you or otherwise lower the level of discourse.
>
> **(4)**: They are deeply rooted in their opinion, but you are able to exchange your views without the conversation degenerating too much.
>
> **(3)**: It's not likely you'll be able to change their mind, but you're easily able to talk and understand each other's point of view.
>
> **(2)**: They may have a clear opinion about the subject, but would likely be open to discussing alternative viewpoints.
>
> **(1)**: They are not set in their opinion, and it's possible you might change their mind. If the comment does not convey an opinion of any kind, you may also select this option.

To ensure quality work, we restricted the task to Masters workers and provided examples corresponding to each point on the scale. Including examples in a task has been shown to significantly increase the agreement and quality of crowdwork

(Doroudi et al., 2016). For instance, here is an example of a highly dogmatic *(5)* comment:

> I won't be happy until I see the executive suite of BofA, Wells, and all the others, frog-marched into waiting squad cars. It's ALREADY BEEN ESTABLISHED that...

And a minimally dogmatic *(1)* comment:

> I agree. I would like to compile a playlist for us trance yogi's, even if you just would like to experiment with it. Is there any preference on which platform to use?

Each comment has been annotated by three independent workers on AMT, which is enough to produce reliable results in most labeling tasks (Sheng et al., 2008). To compute an aggregate measure of dogmatism for each comment, we summed the scores of all three workers. We show the resulting distribution of annotations in Figure 1.

Inter-annotator agreement. To evaluate the reliability of annotations we compute Krippendorff's α, a measure of agreement designed for variable levels of measurement such as a Likert scale (Hayes and Krippendorff, 2007). An α of 0 indicates agreement indistinguishable from chance, while an α of 1 indicates perfect agreement. Across all annotations we find $\alpha = 0.44$. While workers agree much more than chance, clearly dogmatism is also subjective. In fact, when we examine only the middle two quartiles of the dogmatism annotations, we find agreement is no better than chance. Alternatively, when we measure agreement only among the top and bottom quartiles of annotations, we find agreement of $\alpha = 0.69$. This suggests comments with scores that are only slightly dogmatic are unreliable and often subject to human disagreement. For this reason, we use only the top and bottom quartiles of comments when training our model.

3 Approaches to Identifying Dogmatism

We now consider strategies for identifying dogmatism based on prior work in psychology. We start with the Linguistic Inquiry and Word Count (LIWC), a lexicon popular in the social sciences (Pennebaker et al., 2001). LIWC provides human validated lists of words that correspond to high-level psychological categories such as *certainty* or *perception*. In other studies, LIWC has uncovered linguistic signals relating to politeness (Danescu-Niculescu-Mizil et al., 2013), deception (Yoo and Gretzel, 2009), or authority in texts (Gilbert, 2012). Here, we examine how dogmatism relates to 17 of LIWC's categories (Table 1).

To compute the relationships between LIWC categories and dogmatism, we first count the relevant category terms that appear in each annotated Reddit comment, normalized by its word count. We then calculate odds ratios on the aggregate counts of each LIWC category over the top and bottom quartiles of dogmatic comments. As we have discussed, using the top and bottom quartiles of comments provides a more reliable signal of dogmatism. We check for significant differences in categories between dogmatic and non-dogmatic comments using the Mann-Whitney U test and apply Holmes method for correction. All odds we report in this section are significant after correction.

Dogmatic statements tend to express a high degree of certainty (Rokeach, 1954). Here we consider LIWC categories that express certainty both positively (*certainty*) and negatively (*tentativeness*). For example, the word "always" is certain, while "possibly" is tentative. Conforming to existing theory, *certainty* is more associated with dogmatic comments (1.52 odds), while *tentativeness* is more associated with the absence of dogmatism (0.88 odds).

Terms used to verbalize cognition can act as a hedge that often characterizes non-dogmatic language. LIWC's *insight* category captures this effect through words such as "think," "know," or "believe." These words add nuance to a statement (Pennebaker and Francis, 1996), signaling it is the product of someone's mind ("*I think* you should give this paper a good review") and not meant to be interpreted as an objective truth. Along these lines, we find the use of terms in the *insight* category is associated with non-dogmatic comments (0.83 odds).

Sensory language, with its focus on description and detail, often signals a lack of any kind of opinion, dogmatic or otherwise. LIWC's *perception* category captures this idea through words associated with hearing, feeling, or seeing. For example, these words might occur when recounting a personal experience ("I saw his incoming fist"), which even if emotionally charged or negative, is less likely to be dogmatic. We find perception is associated with

Strategy	Odds	Example
Certainty	1.33*	Be a hate monger **all** you want... Your life will **never** truly be happy though, and you will **never** know peace.
Tentativeness	0.88*	Most are **likely** to be more technically advanced and, **if** still using radio, **might** very well be emitting signals we could detect
Insight	0.83*	I **think** stating the obvious is a necessary function. Information like this is important to **consider**...
Perception	0.77*	I **saw** four crows on that same branch, **staring** at the deceased. The **silence** of the crows was **deafening**.
Relativity	0.82*	I've known a number to go **into** shock **during** the procedure
Comparison	0.91	This may be **more than** a coincidence.
I (pronouns)	0.68*	Like **I** said, **I** want to believe the former. **I'm** glad it worked out.
You (pronouns)	2.18*	I don't give a fuck what **you** do. **You** can get drink **yourself** to death, **you** can get **yourself** pregnant...
We (pronouns)	0.96	**We** need a bigger, better, colder fridge. **We** have worked hard...
They (pronouns)	1.63*	**They** want the ability to prosecute who they please.
Past	0.69*	I **was** walking past and **thought** about asking if they **needed** help.
Present	1.11*	Can I **steal** your organs and nutrients if I **need** them and you don't **want** to give them up?
Future	1.06	Trump's thugs **will** be pretending to be Bernie supporters and **will** set fire to Philadelphia.
Interrogatory	1.12*	Gee, **where** was the NY Times back in the day? **Why** didn't we hear of the Kennedys, LBJ and FDR?
Negation	1.35*	If you **didn't** know the woman well enough to know she **didn't** take BC regularly, you certainly **don't** know her well enough to know she **doesn't** have an std.
Negative emotion	2.32*	A prank?!? You **arrogant** son of a **bitch**
Positive emotion	0.96	They were **excellent** fishermen - they built **fine** boats.

Table 1: Linguistic features that capture high level psychological categories and their relationship with dogmatic comments. *Strategy* describes the psychological category. *Odds* describes the likelihood that a category will appear more often in a dogmatic comment (e.g., dogmatic comments are 2.18 times more likely to mention *you*-oriented phrases). *Example* illustrates a comment that matches the category. * indicates significance ($p < 0.05$) after correction with Holmes method.

non-dogmatic comments at 0.77 odds.

Drawing comparisons or qualifying something as relative to something else conveys a nuance that is absent from traditionally dogmatic language. The LIWC categories *comparison* and *relativity* capture these effects through comparison words such as "than" or "as" and qualifying words such as "during" or "when." For example, the statement "I hate politicians" is more dogmatic than "I hate politicians *when* they can't get anything done.' *Relativity* is associated with non-dogmatic comments at 0.80 odds, but *comparison* does not reach significance.

Pronouns can be surprisingly revealing indicators of language: for example, signaling one's gender or hierarchical status in a conversation (Pennebaker, 2011). We find first person singular pronouns are a useful negative signal for dogmatism (0.46 odds), while second person singular pronouns (2.18 odds) and third person plural (1.63 odds) are a useful positive signal. Looking across the corpus, we see *I* often used with a hedge ("I think" or "I know"), while *you* and *they* tend to characterize the beliefs of oth-

ers, often in a strongly opinionated way ("you are a moron" or "they are keeping us down"). Other pronoun types do not show significant relationships.

Like pronouns, verb tense can reveal subtle signals in language use, such as the tendency of medical inpatients to focus on the past (Wolf et al., 2007). On social media, comments written in the *present tense* are more likely to be oriented towards a user's current interaction ("this *is* all so stupid"), creating opportunities to signal dogmatism. Alternatively, comments in the *past tense* are more likely to refer to outside experiences ("it *was* an awful party"), speaking less to a user's stance towards an ongoing discussion. We find *present tense* is a positive signal for dogmatism (1.11 odds) and *past tense* is a negative signal (0.69 odds).

Dogmatic language can be either positively or negatively charged in sentiment: for example, consider the positive statement "*Trump is the SAVIOR of this country!!!*" or the negative statement "*Are you REALLY that stupid?? Education is the only way out of this horrible mess. It's hard to imagine how anyone could be so deluded.*" In diverse communities, where people hold many different kinds of opinions, dogmatic opinions will often tend to come into conflict with one another (McCluskey and Hmielowski, 2012), producing a greater likelihood of negative sentiment. Perhaps for this reason, *negative emotion* (2.09 odds) is a useful positive signal of dogmatism, while *positive emotion* shows no significant relationship.

Finally, we find that *interrogative* language (1.12 odds) and *negation* (1.35 odds) are two additional positive signals of dogmatism. While interrogative words like "how" or "what" have many benign uses, they disproportionately appear in our data in the form of rhetorical or emotionally charged questions, such as "how can anyone be that dumb?"

Many of these linguistic signals are correlated with each other, suggesting that dogmatism is the cumulative effect of many component relationships. For example, consider the relatively non-dogmatic statement: "I think the reviewers are wrong in this instance." Removing signals of *insight*, we have: "the reviewers are wrong in this instance," which is slightly more dogmatic. Then removing *relativity*, we have: "the reviewers are wrong." And finally, adding *certainty*, we have a dogmatic statement: "the reviewers are always wrong."

Classifier	In-domain	Cross-domain
BOW	0.853	0.776
SENT	0.677	0.646
LING	0.801	0.728
BOW + SENT	0.860	0.783
BOW + LING	0.881	0.791

Table 2: The AUC scores for dogmatism classifiers within and across domains. BOW (bag-of-words) and SENT (sentiment signals) are baselines, and LING uses the linguistic features from Table 1. We compute in-domain accuracy using 15-fold cross-validation on the Reddit dataset, and cross-domain accuracy by training on Reddit and evaluating on comments on articles from the New York Times. Chance AUC is 0.5.

4 Predicting dogmatism

We now show how we can use the linguistic feature sets we have described to build a classifier that predicts dogmatism in comments. A predictive model further validates our feature sets, and also allows us to analyze dogmatism in millions of other Reddit comments in a scalable way, with multiple uses in ongoing, downstream analyses.

Prediction task. Our goal is (1) to understand how well we can use the strategies in Section 3 to predict dogmatism, and (2) to test the *domain-independence* of these strategies. First, we test the performance of our model under cross-validation within the Reddit comment dataset. We then evaluate the Reddit-based model on a held out corpus of New York Times comments annotated using the technique in Section 2. We did not refer to this second dataset during feature construction.

For classification, we consider two classes of comments: *dogmatic* and *non-dogmatic*. As in the prior analysis, we draw these comments from the top and bottom quartiles of the dogmatism distribution. This means the classes are balanced, with 2,500 total comments in the Reddit training data and 500 total comments in the New York Times testing data.

We compare the predictions of logistic regression models based on unigram bag-of-words features (BOW), sentiment signals[2] (SENT), the linguistic

[2]For SENT, we use normalized word counts from LIWC's positive and negative emotional categories.

features from our earlier analyses (LING), and combinations of these features. BOW and SENT provide baselines for the task. We compute BOW features using term frequency-inverse document frequency (TF-IDF) and category-based features by normalizing counts for each category by the number of words in each document. The BOW classifiers are trained with regularization (L2 penalties of 1.5).

Classification results. We present classification accuracy in Table 2. BOW shows an AUC of 0.853 within Reddit and 0.776 on the held out New York Times comments. The linguistic features boost classification results within Reddit (0.881) and on the held out New York Times comments (0.791). While linguistic signals by themselves provide strong predictive power (0.801 AUC within domain), sentiment signals are much less predictive.

These results suggest that linguistic features inspired by prior efforts in psychology are useful for predicting dogmatism in practice and generalize across new domains.

5 Dogmatism in the Reddit Community

We now apply our dogmatism classifier to a larger dataset of posts, examining how dogmatic language shapes the Reddit community. Concretely, we apply the BOW+LING model trained on the full Reddit dataset to millions of new unannotated posts, labeling these posts with a probability of dogmatism according to the classifier (0=non-dogmatic, 1=dogmatic). We then use these dogmatism annotations to address four research questions.

5.1 What subreddits have the highest and lowest levels of dogmatism? (R1)

A natural starting point for analyzing dogmatism on Reddit is to examine how it characterizes the site's sub-communities. For example, we might expect to see that subreddits oriented around topics such as abortion or climate change are more dogmatic, and subreddits about cooking are less so.

To answer this question, we randomly sample 1.6 million posts from the entire Reddit community between 2007 and 2015. We then annotate each of these posts with dogmatism using our classifier, and compute the average dogmatism level for each subreddit in the sample with at least 100 posts.

Highest	Score	Lowest	Score
cringepics	0.553	photography	0.399
DebateAChristian	0.551	DIY	0.399
DebateReligion	0.540	homebrewing	0.401
politics	0.536	cigars	0.402
ukpolitics	0.533	wicked_edge	0.404
atheism	0.529	guitar	0.406
lgbt	0.527	gamedeals	0.406
TumblrInAction	0.524	buildapc	0.407
islam	0.523	techsupport	0.410
SubredditDrama	0.520	travel	0.410

Table 3: Subreddits with the highest and lowest dogmatism scores. Politics and religion are common themes among the most dogmatic subreddits, while hobbies (e.g., photography, homebrewing, buildapc) show the least dogmatism.

We present the results of this analysis in Table 3. The subreddits with the highest levels of dogmatism tend to be oriented around politics and religion (*DebateAChristian* or *ukpolitics*), while those with the lowest levels tend to focus on hobbies (*photography* or *homebrewing*). The subreddit with the highest average dogmatism level, *cringepics*, is a place to make fun of socially awkward messages, often from would-be romantic partners. Dogmatism here tends to take the form of "how could someone be that stupid" and is directed at the subject of the post, as opposed to other members of the community.

Similarly, *SubredditDrama* is a community where people come to talk about fights on the internet or social media. These fights are often then extended in discussion, for example: *"If the best you can come up with is that something you did was legal, it's probably time to own up to being an ass."* The presence of this subreddit in our analysis provides a further sanity check that our model is capturing a robust signal of dogmatism.

5.2 How do dogmatic beliefs cluster? (R2)

Dogmatism is widely considered to be a domain-specific attitude (for example, oriented towards religion or politics) as opposed to a deeper personality trait (Rokeach, 1954). Here we use Reddit as a lens to examine this idea more closely. Are users who are dogmatic about one topic likely to be dogmatic about others? Do clusters of dogmatism exist around particular topics? To find out, we examine the re-

Libertarianism	business	conspiracy	science	Christianity	lgbt
Anarcho_Capitalism	Bitcoin	Republican	Christianity	DebateAChristian	feminisms
Bitcoin	economy	conspiritard	relationship_advice	DebateReligion	Equality
ronpaul	entertainment	ronpaul	worldpolitics	science	SubredditDrama
Conservative	TrueReddit	collapse	MensRights	videos	TwoXChromosomes
Android	socialism	guns	IAmA	news	MensRights
ukpolitics	bestof	worldpolitics	TwoXChromosomes	Libertarianism	offbeat
Equality	philosophy	occupywallstreet	WTF	atheism	fffffffuuuuuuuuuuu

Table 4: Clusters of subreddits that share dogmatic users. For example, users who are dogmatic on the *conspiracy* subreddit (a place to discuss conspiracy theories) are also likely to be dogmatic on *guns* or *occupywallstreet*.

lationships between subreddits over which individual users are dogmatic. For example, if many users often post dogmatic comments on both the *politics* and *Christianity* subreddits, but less often on *worldnews*, that would suggest *politics* and *Christianity* are linked per a boost in likelihood of individuals being dogmatic in both.

We sample 1000 Reddit users who posted at least once a year between 2007 and 2015 to construct a corpus of 10 million posts that constitute their entire post history. We then annotate these posts using the classifier and compute the average dogmatism score per subreddit per user. For example, one user might have an average dogmatism level of 0.55 for the *politics* subreddit and 0.45 for the *economics* subreddit. Most users do not post in all subreddits, so we track only subreddits for which a user had posted at least 10 times. Any subreddits with an average dogmatism score higher than 0.50 we consider to be a user's dogmatic subreddits. We then count all pairs of these dogmatic subreddits. For example, 45 users have *politics* and *technology* among their dogmatic subreddits, so we consider *politics* and *technology* as linked 45 times. We compute the mutual information (Church and Hanks, 1990) between these links, which gives us a measure of the subreddits that are most related through dogmatism.

We present the results of this analysis in Table 4, choosing clusters that represent a diverse set of topics. For example, *Libertarianism* is linked through dogmatism to other political communities like *Anarcho_Capitalism*, *ronpaul*, or *ukpolitics*, as well as other topical subreddits like *guns* or *economy*. Similarly, people who are dogmatic in the *business* subreddit also tend to be dogmatic in subreddits for *Bitcoin*, *socialism*, and *technology*. Notably, when we apply the same mutual information analysis to links defined by subreddits posted in by the same user, we see dramatically different results. For example, the subreddits most linked to *science* through user posts are *UpliftingNews*, *photoshopbattles*, and *firstworldanarchist*, and *millionairemakers*.

Feature	Direction
total user posts	↑
proportion of posts in most active subreddit	↑
number of subreddits posted in	↓
average number of posts in active articles	↓

Table 5: User behavioral features that are positively and negatively associated with dogmatism. ↑ means the feature is positively predictive with dogmatism, and ↓ means the feature is negatively predictive. For example, the more subreddits a user posts in, the less likely they are to be dogmatic. All features are statistically significant ($p < 0.001$).

Finally, we see less obvious connections between subreddits that suggest some people may be dogmatic by nature. For example, among the users who are dogmatic on *politics*, they are also disproportionately dogmatic on unrelated subreddits such as *science* ($p < 0.001$), *technology* ($p < 0.001$), *IAmA* ($p < 0.001$), and *AskReddit* ($p < 0.05$), with p-values computed under a binomial test.

5.3 What user behaviors are predictive of dogmatism? (R3)

We have shown dogmatism is captured by many linguistic features, but can we discover other high-level user behaviors that are similarly predictive?

To find out, we compute metrics of user behavior using the data sample of 1000 users and 10 million posts described in Section 5.2. Specifically, we calculate (1) *activity*: a user's total number of posts, (2) *breadth*: the number of subreddits a user has posted in, (3) *focus*: the proportion of a user's posts that appear in the subreddit where they are most active, and (4) *engagement*: the average number of posts a user contributes to each discussion they engage in.

We then fit these behavioral features to a linear regression model where we predict each user's average dogmatism level. Positive coefficients in this model are positively predictive of dogmatism, while negative coefficients are negatively predictive.

We find this model is significantly predicitive of dogmatism ($R^2 = 0.1$, $p < 0.001$), with all features reaching statistical significance ($p < 0.001$). *Activity* and *focus* are positively associated with dogmatism, while *breadth* and *engagement* are negatively associated (Table 5). Together, these results suggest dogmatic users tend to post frequently and in specific communities, but are not as inclined to continue to engage with a discussion, once it has begun.

5.4 How does dogmatism impact a conversation? (R4)

How does interacting with a dogmatic comment impact a conversation? Are users able to shrug it off? Or do otherwise non-dogmatic users become more dogmatic themselves?

To answer this question, we sample 600,000 conversations triples from Reddit. These conversations consist of two people (A and B) talking, with the structure: A1 $\rightarrow$ B $\rightarrow$ A2. This allows us to measure the impact of B's dogmatism on A's response, while also controlling for the dogmatism level initially set by A. Concretely, we model the impact of dogmatism on these conversations through a linear regression. This model takes two features, the dogmatism levels of A1 and B, and predicts the dogmatism response of A2. If B's dogmatism has no effect on A's response, the coefficient that corresponds to B will not be significant in the model. Alternatively, if B's dogmatism does have some effect, it will be captured by the model's coefficient.

We find the coefficient of the B feature in the model is positively associated with dogmatism ($p < 0.001$). In other words, engagement with a dogmatic comment tends to make a user more dogmatic themselves. This effect holds when we run the same model on data subsets consisting only of dogmatic or non-dogmatic users, and also when we conservatively remove all words used by B from A's response (i.e., controlling for quoting effects).

6 Related Work

In contrast to the computational models we have presented, dogmatism is usually measured in psychology through survey scales, in which study participants answer questions designed to reveal underlying personality attributes (Rokeach, 1954). Over time, these surveys have been updated (Shearman and Levine, 2006) and improved to meet standards of psychometric validity (Crowson, 2009).

These surveys are often used to study the relationship between dogmatism and other psychological phenomena. For example, dogmatic people tend to show an increased tendency for confrontation (El-Nawawy and Powers, 2010) or moral conviction and religiosity (Swink, 2011), and less likelihood of cognitive flexibility (Martin et al., 2011), even among stereotypically non-dogmatic groups like atheists (Gurney et al., 2013). From a behavioral standpoint, dogmatic people solve problems differently, spending less time framing a problem and expressing more certainty in their solution (Lohman, 2010). Here we similarly examine how user behaviors on Reddit relate to a language model of dogmatism.

Ertel sought to capture dogmatism linguistically, though a small lexicon of words that correspond with high-level concepts like certainty and compromise (1985). McKenny then used this dictionary to relate dogmatism to argument quality in student essays (2005). Our work expands on this approach, applying supervised models based on a broader set of linguistic categories to identify dogmatism in text.

Other researchers have studied topics similar to dogmatism, such as signals of cognitive style in right-wing political thought (Van Hiel et al., 2010), the language used by trolls on social media (Cheng et al., 2015), or what makes for impartial language on twitter (Zafar et al., 2016). A similar flavor of work has examined linguistic models that capture politeness (Danescu-Niculescu-Mizil et al., 2013), deception (Ott et al., 2011), and authority (Gilbert, 2012). We took inspiration from these models when constructing the feature sets in our work.

Finally, while we examine what makes an opinion dogmatic, other work has pushed further into the structure of arguments, for example classifying their justifications (Hasan and Ng, 2014), or what makes an argument likely to win (Tan et al., 2016). Our

model may allow future researchers to probe these questions more deeply.

7 Conclusion

We have constructed the first corpus of social media posts annotated with dogmatism scores, allowing us to explore linguistic features of dogmatism and build a predictive model that analyzes new content. We apply this model to Reddit, where we discover behavioral predictors of dogmatism and topical patterns in the comments of dogmatic users.

Could we use this computational model to help users shed their dogmatic beliefs? Looking forward, our work makes possible new avenues for encouraging pro-social behavior in online communities.

References

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. *arXiv preprint arXiv:1504.00680*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

H Michael Crowson. 2009. Does the dog scale measure dogmatism? another look at construct validity. *The Journal of social psychology*, 149(3):365–383.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. *arXiv preprint arXiv:1306.6078*.

Shayan Doroudi, Ece Kamar, Emma Brunskill, and Eric Horvitz. 2016. Toward a learning science for complex crowdsourcing tasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2623–2634. ACM.

Mohammed El-Nawawy and Shawn Powers. 2010. Al-jazeera english a conciliatory medium in a conflict-driven environment? *Global Media and Communication*, 6(1):61–84.

S Ertel. 1985. Content analysis: An alternative approach to open and closed minds. *The High School Journal*, 68(4):229–240.

Eric Gilbert. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1037–1046. ACM.

Daniel J Gurney, Shelley McKeown, Jamie Churchyard, and Neil Howlett. 2013. Believe it or not: Exploring the relationship between dogmatism and openness within non-religious samples. *Personality and Individual Differences*, 55(8):936–940.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *EMNLP*, pages 751–762.

Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89.

Margaret C Lohman. 2010. An unexamined triumvirate: dogmatism, problem solving, and hrd. *Human Resource Development Review*.

Matthew M Martin, Sydney M Staggers, and Carolyn M Anderson. 2011. The relationships between cognitive flexibility with dogmatism, intellectual flexibility, preference for consistency, and self-compassion. *Communication Research Reports*, 28(3):275–280.

Michael McCluskey and Jay Hmielowski. 2012. Opinion expression during social conflict: Comparing online reader comments and letters to the editor. *Journalism*, 13(3):303–319.

John McKenny. 2005. Content analysis of dogmatism compared with corpus analysis of epistemic stance in student essays. *Information Design Journal & Document Design*, 13(1).

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

English Oxford Dictionary. 2016. Definition of dogmatism.

James W Pennebaker and Martha E Francis. 1996. Cognitive, emotional, and language processes in disclosure. *Cognition & Emotion*, 10(6):601–626.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

James W Pennebaker. 2011. The secret life of pronouns. *New Scientist*, 211(2828):42–45.

Milton Rokeach. 1954. The nature and meaning of dogmatism.

Sachiyo M Shearman and Timothy R Levine. 2006. Dogmatism updated: A scale revision and validation. *Communication Quarterly*, 54(3):275–291.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622.

Nathan Swink. 2011. Dogmatism and moral conviction in individuals: Injustice for all.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of WWW*.

Alain Van Hiel, Emma Onraet, and Sarah De Pauw. 2010. The relationship between social-cultural attitudes and behavioral measures of cognitive style: A meta-analytic integration of studies. *Journal of personality*, 78(6):1765–1800.

Markus Wolf, Jan Sedway, Cynthia M Bulik, and Hans Kordy. 2007. Linguistic analyses of natural written language: Unobtrusive assessment of cognitive style in eating disorders. *International Journal of Eating Disorders*, 40(8):711–717.

Kyung-Hyan Yoo and Ulrike Gretzel. 2009. Comparison of deceptive and truthful travel reviews. *Information and communication technologies in tourism 2009*, pages 37–47.

Muhammad Bilal Zafar, Krishna P Gummadi, and Cristian Danescu-Niculescu-Mizil. 2016. Message impartiality in social media discussions. In *Tenth International AAAI Conference on Web and Social Media*.

Enhanced Personalized Search using Social Data

Dong Zhou[1] **, Séamus Lawless**[2] **, Xuan Wu**[1] **, Wenyu Zhao**[1]**, Jianxun Liu**[1]
1. Key Laboratory of Knowledge Processing and Networked Manufacturing & School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan, 411201, China
2. ADAPT Centre, Knowledge and Date Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland
dongzhou1979@hotmail.com, seamus.lawless@scss.tcd.ie

Abstract

Search personalization that considers the social dimension of the web has attracted a significant volume of research in recent years. A user profile is usually needed to represent a user's interests in order to tailor future searches. Previous research has typically constructed a profile solely from a user's usage information. When the user has only limited activities in the system, the effect of the user profile on search is also constrained. This research addresses the setting where a user has only a limited amount of usage information. We build enhanced user profiles from a set of annotations and resources that users have marked, together with an external knowledge base constructed according to usage histories. We present two probabilistic latent topic models to simultaneously incorporate social annotations, documents and the external knowledge base. Our web search strategy is achieved using personalized social query expansion. We introduce a topical query expansion model to enhance the search by utilizing individual user profiles. The proposed approaches have been intensively evaluated on a large public social annotation dataset. Results show that our models significantly outperformed existing personalized query expansion methods which use user profiles solely built from past usage information in personalized search.

1 Introduction

On today's social web, users can enrich the social context of web pages. The most notable fact is that users can often freely tag web pages with annotations (Gupta et al., 2011). These tags could be high quality descriptors of the web pages' topics and a good indicator of web users' interests. However, the uncontrolled manner of social tagging results in the use of an unrestricted vocabulary. This makes searching through the collection difficult and generally less accurate. Thus the social annotation or bookmarking system demonstrates an extreme example of the vocabulary mismatch problem encountered in personalized web search. To tackle the problem, various personalized query expansion (QE) and results reranking techniques have been proposed and evaluated (Bouadjenek et al., 2016).

There have been some attempts to achieve personalized QE using social data. For example, Researchers have considered selecting the most related tags from a user's profile to expand queries (Bender et al., 2008; Bertier et al., 2009; Bouadjenek et al., 2011). Local analysis and co-occurrence based user profile representation have also been adopted to expand the query (Chirita et al., 2007; Biancalana et al., 2013). Recently, Zhou et al. proposed a query expansion framework based on individual user profiles (Zhou et al., 2012a). In their work, terms in the user profile are modeled according to their associations, which can be defined by co-occurrence statistics or defined by a tag-topic model.

All of the previously mentioned systems are dependent upon historical usage information being available in an individual user profile (Sugiyama et al., 2004; Teevan et al., 2005; Bennett et al., 2012; Zhou et al., 2014; Guha et al., 2015; Zhou et al., 2016). This information is pivotal when tailoring search results to the preferences of specific individuals. However, in some

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 700–710,
Austin, Texas, November 1-5, 2016.

cases a user may have very limited previous interactions with the system. With little usage information to hand, the personalized search experience is poor. Furthermore, using only historical usage information to personalize search may not be enough.

In this paper, we extend personalized search using social data in two directions. First, we exploit external knowledge bases to enhance the user profile built from a user's historical usage information. We build queries from the user tags and annotated web pages. Subsequently, we fetch the relevant documents from an external corpus to be included in the user profile. We then propose to incorporate the user's annotations, web page content information and external documents through two statistical models, which we have named Mixture Enhanced User Profiling (MEUP) Model and Separated Enhanced User Profiling (SEUP) Model. Both models infer latent topics, their probabilities of being relevant and a multinomial distribution of topics of the documents being considered. MEUP mixed the tags, annotated documents and external documents together to infer unified latent topics, while SEUP is an extension of MEUP which learns topics that are shared between the two groups of document-aligned pairs.

Second, we propose a topical query expansion model to personalize web search by utilizing the user profiles. In the topical QE model, profile terms are calculated based on their topical relevance to the query terms to expand the query.

Experimental results show that the Enhanced User Profiling models together with the topic QE can significantly improve retrieval performance over user profiles solely built from a user's historical information. Improvements were observed for users with both a rich amount of usage information and a small amount of information. We also demonstrate that the approach proposed in the paper outperforms existing QE methods proposed for personalized search using social data.

The contribution of this paper can be summarized as follows:

i. *We tackle the challenge of personalized web search using social data in a novel way by enhancing user profiles that are built solely from users' historical usage information.*

ii. *We propose and systematically evaluate two novel generative models to construct enriched user profiles with the help of external corpora in the context of personalized search using social data.*

iii. *We suggest and evaluate a novel query expansion method. Instead of relying on lexical relevance information between query terms and profile terms, we also consider the topical relevance between them to expand the query.*

2 Related Work

2.1 Personalized Search Using Social Media

In personalized search using social media (Jamali and Ester, 2010; Lin et al., 2013), the search process is either performed over "social" data gathered from Web 2.0 applications such as social bookmarking systems, wikis, blogs etc., or it re-adapts the web search results produced by search engines by using social data (Carman et al., 2008; Bouadjenek et al., 2016). For example, the authors in (Vallet et al., 2010) investigated how the ranking of search engine results can be improved with respect to users if the users' social information is taken into consideration. A similar approach was also explored in (Noll and Meinel, 2007) where the system performed re-ranking of Google search results based on social bookmarks and tags harvested from *del.icio.us*[1]. However, the data sparsity problem poses a challenge to this approach as not all Web pages returned by search engines are tagged in the *del.icio.us* dataset.

2.2 Personalized Results Re-Ranking

Because of this problem, researchers started to use social data as a test collection to develop personalized techniques. In this way, personalization usually involves two general approaches. The first approach submits a query into the collection but re-ranks the returned results based on an individual user profile. In (Xu et al., 2008) the authors re-rank the results according to the topical relevance of documents and users' interests. Carmel et al. (Carmel et al., 2009) investigated personalized results re-ranking based on the user's social relations. Wang and Jin (Wang and Jin, 2010) explored gathering data from multiple online social systems for adaptive search personalization. Bouadjenek et al. (Bouadjenek et al., 2013a; Bouadjenek et al., 2013b) propose to use social data and user relationships to enhance document

[1] http://www.delicious.com

Notation	Meaning
$\mathcal{U}$	finite sets of users
$\mathcal{D}$	finite sets of web pages/documents
$\mathcal{T}$	finite sets of tags
$\mathcal{A}$	a ternary relation, elements are tags
$\mathcal{A}^u$	the set of annotations of a user
$\mathcal{T}^u$	the tag vocabulary of a user
$\mathcal{D}^u$	a user's set of documents
t	a tag
d	a document
u	a user
w	a word/term
$docTerm^u$	the vocabulary extracted from the documents that a user has tagged
$exterTerm^u$	the full set of terms extracted from a user's external documents
$\mathcal{D}_{exter}$	an external corpus
$\mathcal{D}^u_{exter}$	a user's set of external documents
q	a source query
$q^{\mathcal{T}^u}$	a query concatenated by tags of a user
$Q^{\mathcal{D}^u}$	queries extracted from a user's set of documents
Q_{exter}	queries to be sent to an external corpus
$s_{x,y}$	retrieval score of a query q_y to retrieve a document d_x
K	number of topics
μ_z	mean of Log-normal distribution of retrieval scores for topic z
σ_z	deviation of Log-normal distribution of retrieval scores for topic z
N_{d_j}	number of `words` in document d_j
$z_{j,i}$	topic associated with the i-th word in the document d_j
$w_{j,i}$	i-th word in document d_j
$n_{j,k}$	the number of times that topic k sampled w.r.t. document d_j
$v_{k,w_{j,i}}$	the number of times $w_{j,i}$ has been generated by topic k
θ	multinomial distribution of topics
φ	multinomial distribution of words
ϕ	multinomial distribution of words (used in SEUP)
α	the parameter of topic Dirichlet prior
β	the parameter of word Dirichlet prior

Table 1. Basic notations used in the paper

representation for re-ranking purposes. Though this group of work is attractive, if relevant items cannot be fetched in the first place, regardless of the complex re-ranking process, the results still tend to be unsatisfactory.

2.3 Personalized Query Expansion

Another group of work modifies or augments a user's original query. This approach is termed query expansion (Zhou et al., 2015). Researchers have considered tag-tag relationships for personalized query expansion, by selecting the most related tags from a user's profile (Bender et al., 2008; Bertier et al., 2009). However, tags cannot be relied upon to consistently provide precise descriptions of resources for use when searching. Local analysis and co-occurrence based user profile representation have also been adopted to expand the query (Chirita et al., 2007; Biancalana and Micarelli, 2009). However, the expansion terms are solely based on lexical matching between the query and the terms which exist in the user profile. Zhou et al. proposed a query expansion framework based on individual user profiles (Zhou et al., 2012a; Zhou et al., 2012b). In their work, terms in the user profile are modeled according to their associations, which can be defined by co-occurrence statistics or defined by a tag-topic model. The method simultaneously incorporates annotations and web documents in a latent graph, regularized by terms extracted from the top-ranked documents.

However, all of the previously mentioned systems consider constructing user profiles solely from past usage information. In contrast, in this paper we extend personalized web search using social data by exploiting an external knowledge base to enhance the user profile.

3 Problem Definition and Solution Overview

In social annotation and bookmarking systems such as *del.icio.us* or *CiteUlike*[2], users can label interesting web resources with primarily short and unstructured *annotations* in natural language called *tags*. These web resources are denoted as a URL in the *del.icio.us* website. Textual content can be crawled by following a URL that refers to a *document* or *web page*. Please refer to Table 1 for the basic notations used in this paper.

Formally, social tagging data can be represented by a tuple $\mathcal{P} := (\mathcal{U}, \mathcal{D}, \mathcal{T}, \mathcal{A})$. $\mathcal{A} \subseteq \mathcal{U} \times \mathcal{D} \times \mathcal{T}$ is a ternary relation, whose elements are called tag assignments or annotations (or bookmarks). The

[2] http://www.citeulike.org

Input: A query q
Tags of a user $\mathcal{T}^u$
Documents of a user $\mathcal{D}^u$
An external corpus $\mathcal{D}_{exter}$
Output: An expanded query q'

/* step one: External documents fetch */
1. $q^{\mathcal{T}^u} \leftarrow \cup(t \in \mathcal{T}^u)$
2. **for all** $d_x \in \mathcal{D}^u$ **do**
3. $Q^{\mathcal{D}^u} \leftarrow EXTRACTTOP(w \in d_x)$
4. $Q_{exter} \leftarrow q^{\mathcal{T}^u} \cup Q^{\mathcal{D}^u}$
5. **for all** $q_y \in Q_{exter}$ **do**
6. $\mathcal{D}^u_{exter} \leftarrow RETRIEVE_{\mathcal{D}_{exter}}(q_y)$
7. Record retrieval score $s_{x,y}$
/* step two: User profile modelling */
8. **for** $k \in [1, K]$ **do**
9. Initialize μ_z and σ_z randomly
10. **for** $d_j \in \mathcal{T}^u \cup \mathcal{D}^u \cup \mathcal{D}^u_{exter}$ **do**
11. **for** w_i indexed by $i = 1, \dots, N_{d_j}$ **do**
12. Draw $z_{j,i}$ from $p(z_{j,i} = k)$
13. Update $n_{j,k}$ and $v_{k,w_{j,i}}$
14. Calculate the posterior estimate of θ and φ
/* step three: Personalized query expansion */
15. $\{w_1, w_2 \dots w_n\} \leftarrow exterTerm^u \cup docTerm^u \cup \mathcal{T}^u$
16. **for all** $w \in \{w_1, w_2 \dots w_n\}$ **do**
17. calculate $P(w|q)$ using topics from step two
18. Output q' consists of top δ terms with the highest $P(w|q)$

Table 2. Procedure for personalized search using social data

set of annotations of a user is defined as: $\mathcal{A}^u \coloneqq \{(t, d)|u, d, t \in \mathcal{A}\}$. The tag vocabulary of a user, is given as $\mathcal{T}^u \coloneqq \{t|(t, d) \in \mathcal{A}^u\}$. A user's set of documents is $\mathcal{D}^u \coloneqq \{d|(t, d) \in \mathcal{A}^u\}$. We define terms extracted from a user's set of documents as $docTerm^u \coloneqq \{w|w \in \mathcal{D}^u\}$, where w denotes a word/term in the annotated documents. Similarly, we define terms extracted from a user's set of external documents as $exterTerm^u \coloneqq \{w|w \in \mathcal{D}^u_{exter}\}$, where $\mathcal{D}^u_{exter}$ denotes a user's set of external documents from an external corpus $\mathcal{D}_{exter}$.

In a typical personalized search scenario, given a source query q and a set of words in the user profile $\{w_1, w_2 \dots w_n\}$ the goal is to return a ranked list of profile terms to be added to the query, for a second round retrieval of results.

Our personalization approach consists of three main steps (see Table 2): Fetching external documents; User profile modelling; and Personalized query expansion. We enhance a user's historical usage information in step one. We firstly concatenate all tags t in $\mathcal{T}^u$ into a query $q^{\mathcal{T}^u}$ (line 1). Then for each document d in $\mathcal{D}^u$, we extract terms with the highest inverted document frequency (*idf*) scores as queries $Q^{\mathcal{D}^u}$ (lines 2-3, with the *EXTRACTTOP* function returns top λ terms). Next we send queries in Q_{exter} ($q^{\mathcal{T}^u} \cup Q^{\mathcal{D}^u}$) to an external corpus $\mathcal{D}_{exter}$ to fetch $\mathcal{D}^u_{exter}$ together with their retrieval scores $s_{x,y}$ (lines 5-7, the number of documents retrieved by each query is controlled by the parameter γ). Step two integrates $\mathcal{T}^u$ (here all tags are concatenated and viewed as a single document), $\mathcal{D}^u$, $\mathcal{D}^u_{exter}$ and their retrieval scores $s_{x,y}$ into a topic model such that a multinomial distribution of topics specific to each document can be inferred (lines 8-14, we eliminate the procedure for the SEUP model because its similarity to the simpler model, see the next section). In the last step, the algorithm uses the output of step two to build a topical query expansion model in order to expand the original query (lines 15-18). Note that step one and step two could be executed off-line so as to improve the efficiency of the algorithm.

4 Enhanced User Profiling Models

In this section we describe how to model user profiles (i.e. step two in Table 2). We present two Enhanced User Profiling (EUP) models for this purpose.

4.1 Mixture Enhanced User Profiling

Topic discovery in the EUP models is influenced not only by term co-occurrences, but also by the retrieval scores of documents. To avoid normalization, we employ a log-normal distribution for retrieval scores to infer latent topics via the documents and their relevance probabilities.

The MEUP model developed here is a generative model of retrieval scores and the words in the documents. The generative process is as follows:

Generative process of the MEUP model

1. **for** each topic $k \in [1, K]$ **do**
sample the mixture of words $\varphi \sim Dirichlet(\beta)$
2. **for** each document $d_j \in \mathcal{T}^u \cup \mathcal{D}^u \cup \mathcal{D}^u_{exter}$ **do**
sample the mixture of topics $\theta_j \sim Dirichlet(\alpha)$
for each word w_i indexed by $i = 1, \dots, N_{d_j}$ **do**
sample the topic index topic $z_{j,i} \sim Mult(\theta_{d_j})$
sample the weight of word $w_{j,i} \sim Mult(\varphi_{z_{j,i}})$
sample the retrieval score $s_{,i} \sim \mathcal{N}(\mu_{z_{j,i}}, \sigma_{z_{j,i}})$

In the above process, the retrieval scores of terms in the same document are the same and calculated by a language model retrieval function (Manning et al., 2008) for retrieved documents in $\mathcal{D}^u_{exter}$. The retrieval scores for $\mathcal{T}^u$ (here all tags are concatenated and viewed as a single document) and documents in $\mathcal{D}^u$ are set to one. We normalize the scores by the max score in the retrieval list. We used a fixed number of latent topics K. The posterior distribution of topics depends on two sets of information, both the terms and retrieval scores of the documents.

In this model, inference is intractable. We use Gibbs Sampling (Heck and Thomas, 2015) to perform approximate inference. We employ a conjugate prior for the multinomial distributions, and integrate out θ and φ. In the sampling procedure, we need to calculate the conditional distribution $p(z_{j,i} = k)$ (line 12 in Table 2). By using Gibbs Sampling, for each word the topic is sampled from:

$$p(z_{j,i} = k) \propto \frac{n_{j,k,\neg i} + \alpha}{n_{j,\cdot,\neg i} + K \cdot \alpha} \times \frac{v_{k,w_{j,i},\neg} + \beta}{v_{k,\cdot,\neg} + V \cdot \beta} \times \frac{1}{s_{j,i}\sigma_{z_{j,i}}\sqrt{2\pi}} \exp\left(-\frac{\left(\ln s_{j,i} - \mu_{z_{j,i}}\right)^2}{2\sigma^2_{z_{j,i}}}\right)$$

where $n_{j,k,\neg i}$ counts the number of times that topic with index k has been sampled from the multinomial distribution specific to document d_j with the current $z_{j,i}$ not counted. Another counter variable $v_{k,w_{j,i},\neg}$ counts the number of times $w_{j,i}$ has been generated by topic k, but not counting the current $w_{j,i}$. A dot denotes summation over all values of the variable whose index that dot takes. $\mu_{z_{j,i}}$ and $\sigma_{z_{j,i}}$ are elements from μ_z and σ_z, respectively. After that we can calculate the posterior estimate of θ and φ (line 14 in Table 2).

4.2 Separated Enhanced User Profiling

In the MEUP model, $\mathcal{T}^u$, $\mathcal{D}^u$ and $\mathcal{D}^u_{exter}$ are mixed together to infer unified latent topics. However, the MEUP model may miss important information when the topics are learned. Our SEUP model extends the MEUP model by learning topics which are shared between document-aligned pairs. In order to do this, we create pseudo-aligned documents between $\mathcal{T}^u$, $\mathcal{D}^u$ and $\mathcal{D}^u_{exter}$. This procedure works as follows. For each external document in $\mathcal{D}^u_{exter}$ retrieved by a query from Q_{exter}, which is formed through step one of our approach, we treat the document (from $\mathcal{D}^u_{exter}$) and the query (from Q_{exter}) as pseudo-aligned documents in two groups. The first group we named source group C, the other group we named target group E. By using the aligned documents, we propose a model to learn the latent topics between the two groups.

Note that in this case, there is a comparable document set aligned at the document-level. Therefore, θ can be viewed as a group independent factor, and shared among comparable aligned documents. Henceforth, the generation process for the SEUP model is slightly different from the MEUP model. The generative process is summarized below:

Generative process of the SEUP model

1. **for** each of topics $k \in [1, K]$ **do**
   sample the mixture of words $\varphi \sim Dirichlet(\beta)$
   sample the mixture of words $\phi \sim Dirichlet(\beta)$
2. **for** each document pair $d_j = \{d^C_j \in \mathcal{T}^u \cup \mathcal{D}^u, d^E_j \in \mathcal{D}^u_{exter}\}$ **do**
   sample the mixture of topics $\theta_j \sim Dirichlet(\alpha)$
   for each word w^C_i indexed by $i = 1, \ldots, N_{d^C_j}$ **do**
   sample the topic index topic $z^C_{j,i} \sim Mult(\theta_{d_j})$
   sample the weight of word $w^C_{j,i} \sim Mult(\varphi_{z^C_{j,i}})$
   sample the retrieval score $s^C_{j,i} \sim \mathcal{N}(\mu^C_{z_{j,i}}, \sigma^C_{z_{j,i}})$
   for each word w^E_i indexed by $i = 1, \ldots, N_{d^E_j}$ **do**
   sample the topic index topic $z^E_{j,i} \sim Mult\left(\theta_{d_j}\right)$
   sample the weight of word $w^E_{j,i} \sim Mult(\phi_{z^E_{j,i}})$
   sample the retrieval score $s^E_{j,i} \sim \mathcal{N}(\mu^E_{z_{j,i}}, \sigma^E_{z_{j,i}})$

Similar to the MEUP model, the updated formulas for Gibbs sampling for the SEUP model are:

$$p(z^C_{j,i} = k) \propto \frac{n^C_{j,k,\neg i} + n^E_{j,k} + \alpha}{n^C_{j,\cdot,\neg i} + n^E_{j,\cdot} + K \cdot \alpha} \times \frac{v^C_{k,w^C_{j,i},\neg} + \beta}{v^C_{k,\cdot,\neg} + V^C \cdot \beta} \times \frac{1}{s^C_{j,i}\sigma^C_{z_{j,i}}\sqrt{2\pi}} \exp\left(-\frac{\left(\ln s^C_{j,i} - \mu^C_{z_{j,i}}\right)^2}{2\sigma^{C\,2}_{z_{j,i}}}\right)$$

$$p(z^E_{j,i} = k) \propto \frac{n^E_{j,k,\neg i} + n^C_{j,k} + \alpha}{n^E_{j,\cdot,\neg i} + n^C_{j,\cdot} + K \cdot \alpha} \times \frac{v^E_{k,w^E_{j,i},\neg} + \beta}{v^E_{k,\cdot,\neg} + V^E \cdot \beta} \times \frac{1}{s^E_{j,i}\sigma^E_{z_{j,i}}\sqrt{2\pi}} \exp\left(-\frac{\left(\ln s^E_{j,i} - \mu^E_{z_{j,i}}\right)^2}{2\sigma^{E\,2}_{z_{j,i}}}\right)$$

The meaning of the symbols used in the SEUP model is the same as in the MEUP model, except this time for two groups *E* and *C*. In the two EUP models, the multinomial distribution of topics is specific to each document and each word can be easily inferred.

5 Topical Query Expansion

In step three of our approach to personalization, we use the output from step two to build a QE model that calculates the weights of the profile terms to be added to the initial query. In this section we detail this process.

Given the query $q = \{w_a^q\}_{a=1}^n$ of n independent query terms, the probability of the query generating a word w is defined as (see also (Lavrenko and Croft, 2001; Ganguly et al., 2012)):

$$P(w|q) = P(w|w_1^q, \dots, w_n^q) \propto \prod_{a=1}^{n} P(w|w_a^q)$$

We further assume that there are a set of relevant documents $\{d_b\}_{b=1}^N$ related to the query and the word being considered, where N is the number of documents. Incorporating this set of documents into the above equation leads to:

$$P(w|w_a^q) = \sum_{b=1}^{N} P(w|d_b)P(d_b|w_a^q) \propto \frac{1}{N}\sum_{b=1}^{N} P(w|d_b)P(w_a^q|d_b)$$

The calculation discards the uniform prior for $P(w_a^q)$, and takes the uniform prior of documents outside the summation.

As we already have outputs from step two, the documents inside the user profile can be used as a set of relevant documents in the above calculation. In addition, because we now have latent topics related to each document and each word, there is no longer a direct dependency of w on d_b and q. In this case, in order to estimate $P(w|d_b)$, we can marginalize the probability over the latent topic variables z_k, then we have:

$$P(w|d_b) = \sum_{k=1}^{K} P(w|z_k)P(z_k|d_b)$$

Similarly, the probability $P(w_a^q|d_b)$ becomes:

$$P(w_a^q|d_b) = \sum_{k=1}^{K} P(w_a^q|z_k)P(z_k|d_b)$$

So that the probability of the query generating a word w can be re-defined as:

$$P(w|w_a^q) \propto \frac{1}{N}\sum_{b=1}^{N}(\sum_{k=1}^{K} P(w|z_k)P(z_k|d_b))\times(\sum_{k=1}^{K} P(w_a^q|z_k)P(z_k|d_b))$$

In the SEUP Model, we use one side of the word-topic distributions from the group that contains tags and annotated documents to calculate the weighting. All the profile terms $\{w_1, w_2 \dots w_n\} = exterTerm^u \cup docTerm^u \cup \mathcal{T}^u$ are ranked by their probability of being generated by the given query $P(w|q)$ (line 16-17 in Table 2), and the top δ terms are chosen to expand the query.

6 Evaluation

In the following section we describe experiments which have been designed to evaluate the proposed method. We start the section by discussing the experimental settings, and then we present and analyze the results.

6.1 Experimental Setup

In order to evaluate the above proposed methods on real-world data, we selected two delicious datasets: *socialbm0311* and *deliciousT140*, which are public, described and analyzed in (Zubiaga et al., 2009; Zubiaga et al., 2013). The *deliciousT140* dataset is made up by 144,574 unique URLs, all of them with their corresponding social tags retrieved from *del.icio.us*. However, this dataset does not contain the actual web pages (i.e. documents). So we used another *socialbm0311* dataset. It contains the complete bookmarking activity for almost 2 million users. After matching the documents in *deliciousT140 wi*th the bookmark activities in *socialbm0311*, we obtained a total of 5,153,720 bookmark activities, 259,511 users, 131,283 web pages and 137,870 tags. We used a public parser[3] to parse the web pages in order to get their textual content.

We constructed two corpora from different external knowledge bases. The first corpus was obtained from the largest encyclopedia – Wikipedia[4]. A Wikipedia snapshot was obtained on the 14/08/2014, which contained a collection of 4,634,369 articles. The second corpus consists of English news documents from the Glasgow Herald 1995, Los Angeles Times 1994 and Los Angeles Times 2002, a collection made available by the CLEF AdHoc-News Test Suites (2004-2008)[5], which we refer to as CLEF. This collection contains 304,630 documents.

To investigate the effects of enhanced user profiles, we selected two groups of users as test users. One group contains 1,000 randomly selected users with no more than 50 bookmarks (refer to as **User-SMALL**) and another group contains 1,000 randomly selected users with more than 200 bookmarks (refer to as **User-LARGE**). These two groups of users represent users with small amount

[3] http://htmlparser.sourceforge.net/
[4] http://www.wikipedia.org
[5] http://catalog.elra.info/

of and rich amount of past usage information respectively. The English terms were processed by down-casing the alphabetic characters, removing the stop words and stemming words using the Por ter stemmer. For each user, 75% of his/her tags with annotated web pages were used to create the user profile and the other 25% were used as a test collection.

The evaluation method used by previous researchers in personalized social search (Xu et al., 2008; Wang and Jin, 2010; Zhou et al., 2012a) is employed. The main assumption is as follows: Any documents tagged by u with t are considered relevant for the personalized query (u, t) (u submits the query t).

The following evaluation metrics were chosen to measure the effectiveness of the various approaches: the normalized discounted cumulative gain (NDCG), mean reciprocal rank (MRR) and mean average precision (MAP) (Voorhees, 1999; Järvelin and Kekäläinen, 2000). The average performance over all users is calculated. Statistically significant differences were determined using a paired t-test at a confidence level of 95%.

6.2 Experimental Runs

The proposed approach is applied to social search personalization through the means of query expansion. We evaluate our proposed models and compare with several state-of-the-art methods as follows.

LM A popular and quite robust language model retrieval method which has previously demonstrated good results (Zhai and Lafferty, 2001). We compute the Kullback-Leibler divergence between the query and document language model as described in (Zhai and Lafferty, 2001).

LMRM A relevance model involves pseudo-relevance feedback in the language model as in (Lavrenko and Croft, 2001). We include this model as a competitive non-personalized query expansion baseline.

LMRM-external This is a modified version of the relevance model as described in (Diaz and Metzler, 2006). Instead of using the top-ranked documents as pseudo-relevance documents, this model uses external corpora to obtain the relevance documents. We include this model as a strong non-personalized baseline as we also used external corpora in our models. In the experiments, this method will acquire external documents from the Wikipedia corpus and CLEF.

Co-occur This method has been used by several researchers. In this method the selection of expansion terms is based on their co-occurrence statistics with the query terms and other terms inside the user model. We used this approach as previously it demonstrated satisfactory performance as in (Chirita et al., 2007).

Co-tag Pure tag-tag relationships are also favored by many researchers. This method is based on the co-tagging activities a user performed (Bender et al., 2008; Bertier et al., 2009; Bouadjenek et al., 2011). In this case, the user profiles contain training tags with their co-tagging statistics computed using the Jaccard coefficient.

Tag-topic-regu Zhou et al. (Zhou et al., 2012a) proposed a query expansion framework based on regularizing the smoothness of word associations over a connected graph using terms extracted from top-ranked documents. The user profiles are built according to a Tag-Topic model in a latent graph. We include the highest performing method from their work for comparison.

MEUP From our proposed methods, the MEUP method using the MEUP model and the topical query expansion method for social web search.

SEUP This is our alternative proposed method, by using the SEUP model and the topical query expansion method to personalize search.

The number of documents retrieved by each query in step one is set to $\gamma = 5$ empirically. Parameter λ used in the *EXTRACTTOP* function is set to 10. For the EUP modeling, α and β were set to $50/K$ and 0.01. In the expansion method, the number of expansion terms δ are set to 5. All the parameters in the other baseline models are set according to their tuning procedures in the original papers

6.3 Results

Firstly we examine the experimental results that describe the performance of the proposed methods in this paper together with three non-personalized baselines on the overall test users, which are shown in Table 3. The statistically significant differences are marked as *l* and *w* with respect to the **LMRM** and **LMRM-external** baselines as these two methods work better than the simpler **LM**

User-SMALL

	Wikipedia			CLEF		
	MAP	**NDCG**	**MRR**	**MAP**	**NDCG**	**MRR**
LM	0.0216	0.0449	0.0226	0.0216	0.0449	0.0226
LMRM	0.0241	0.0547	0.0261	0.0241	0.0547	0.0261
LMRM-external	0.0283	0.0588	0.0307	0.0272	0.0585	0.0290
Co-occur	0.0499 $^{l, w}$	0.0812 $^{l, w}$	0.0600 $^{l, w}$	0.0499 $^{l, w}$	0.0812 $^{l, w}$	0.0600 $^{l, w}$
Co-tag	0.0491 $^{l, w}$	0.0758 $^{l, w}$	0.0538 $^{l, w}$	0.0491 $^{l, w}$	0.0758 $^{l, w}$	0.0538 $^{l, w}$
Tag-topic-regu	0.0597 $^{l, w, o, t,}$	0.0955 $^{l, w, o, t,}$	0.0666 $^{l, w, o, t,}$	0.0597 $^{l, w, o, t,}$	0.0955 $^{l, w, o, t,}$	0.0666 $^{l, w, o, t,}$
MEUP	0.0729 $^{l, w, o, t, r}$	0.1058 $^{l, w, o, t, r}$	0.0844 $^{l, w, o, t, r}$	0.0722 $^{l, w, o, t, r}$	0.0981 $^{l, w, o, t, r}$	0.0770 $^{l, w, o, t, r}$
SEUP	0.0906 $^{l, w, o, t, r}$	0.1316 $^{l, w, o, t, r}$	0.1032 $^{l, w, o, t, r}$	0.0802 $^{l, w, o, t, r}$	0.1221 $^{l, w, o, t, r}$	0.0940 $^{l, w, o, t, r}$

User-LARGE

	Wikipedia			CLEF		
	MAP	**NDCG**	**MRR**	**MAP**	**NDCG**	**MRR**
LM	0.0178	0.0366	0.0194	0.0178	0.0366	0.0194
LMRM	0.0185	0.0400	0.0201	0.0185	0.0400	0.0201
LMRM-external	0.0195	0.0433	0.0263	0.0190	0.0420	0.0245
Co-occur	0.0386 $^{l, w}$	0.0578 $^{l, w}$	0.0409 $^{l, w}$	0.0386 $^{l, w}$	0.0578 $^{l, w}$	0.0409 $^{l, w}$
Co-tag	0.0381 $^{l, w}$	0.0546 $^{l, w}$	0.0399 $^{l, w}$	0.0381 $^{l, w}$	0.0546 $^{l, w}$	0.0399 $^{l, w}$
Tag-topic-regu	0.0470 $^{l, w, o, t,}$	0.0778 $^{l, w, o, t,}$	0.0498 $^{l, w, o, t,}$	0.0470 $^{l, w, o, t,}$	0.0778 $^{l, w, o, t,}$	0.0498 $^{l, w, o, t,}$
MEUP	0.0579 $^{l, w, o, t, r}$	0.0971 $^{l, w, o, t, r}$	0.0629 $^{l, w, o, t, r}$	0.0545 $^{l, w, o, t, r}$	0.0805 $^{l, w, o, t, r}$	0.0581 $^{l, w, o, t, r}$
SEUP	0.0633 $^{l, w, o, t, r}$	0.1049 $^{l, w, o, t, r}$	0.0678 $^{l, w, o, t, r}$	0.0604 $^{l, w, o, t, r}$	0.0978 $^{l, w, o, t, r}$	0.0651 $^{l, w, o, t, r}$

Table 3. Overall results, statistically significant differences between our methods and LMRM, LMRM-External, Co-occur, Co-tag, Tag-topic-regu are indicated by *l, w, o, t, r* respectively.

method. As illustrated by the results, the **LM** model was the lowest performer for all evaluation metrics for two groups of users. This result shows that merely borrowing common lexical-matching techniques from traditional information retrieval will not solve the personalized search problem. With the help of pseudo-relevance feedback, the **LMRM** and **LMRM-external** methods work consistently better than the **LM** baseline. This demonstrates the power of query expansion. Specifically, the technique that explores external corpora to obtain the relevant documents works better than the method which simply uses top-ranked documents. The results are consistent with previous research (Diaz and Metzler, 2006). The improvements are more noticeable when using Wikipedia as the external corpus. However, all the non-personalized baselines are outperformed by the personalized approaches including our proposed methods **MEUP** and **SEUP**, all with statistically significant results. This illustrates that non-personalized query expansion methods can only bring limited improvements while methods with additional terms from the user profiles can greatly improve retrieval effectiveness.

Next we evaluate the performance of the proposed methods compared to several personalized baselines that use only the users' past information for query expansion, i.e. **Co-occur**, **Co-tag** and **Tag-topic-regu** methods.

As seen from Table 3, three conclusions emerge. First, **MEUP** and **SEUP** both outperform all personalization methods previously proposed, in all metrics measured with two external corpora for both groups of users. Moreover, the difference between our proposed methods and the baseline runs is always significant. We believe that the strong performance of our methods is due to the fact that our methods do not only consider a user's past usage information, but also uses an external knowledge base to enhance the user profiling process. Secondly, the **SEUP** method works consistently better than the **MEUP** method. This result confirms that merely mixing the documents from the historical evidence and external knowledge bases will miss some important information. By treating the documents as a pseudo-aligned corpus, we obtain much better performance. The highest improvement over the best performing run reaches 54.95% (in terms of the **SEUP** method with the MRR metric when compared to **Tag-topic-regu** by using Wikipedia as the external corpus in the **User-SMALL** group). Third, further improvements are achieved by us-

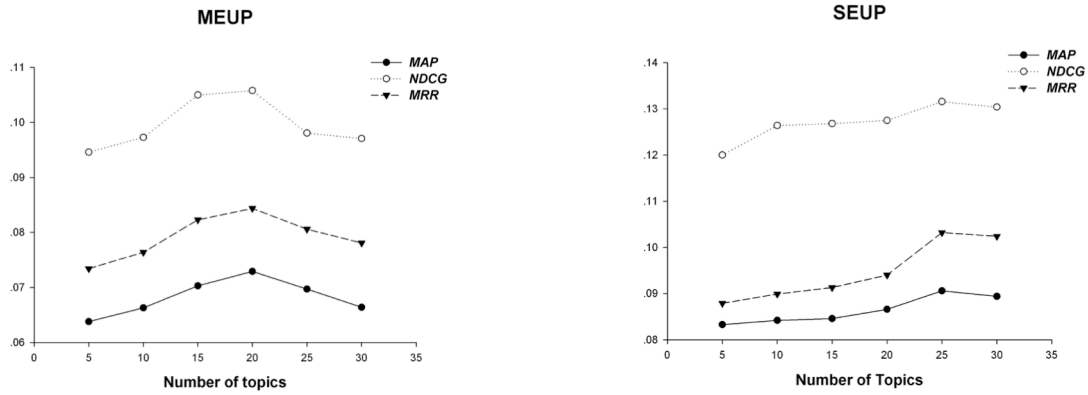


Fig. 1. Performance with different number of topics by using Wikipedia as external corpus in the user-SMALL group

ing Wikipedia as the external corpus rather than using the CLEF collection. The possible reason, as pointed out by Diaz and Metzler (Diaz and Metzler, 2006), is that an external corpus is likely to be a better source of expansion terms if it has better topic coverage over the target corpus. The results also show that the improvements over baseline models in the **User-SMALL** group are more noticeable than in the **User-LARGE** group. However, the differences are small. This result confirms that our methods work well both for users with small amounts, and those with rich amounts of past usage information.

We now examine the effect of the performance of the number of latent topics used in **MEUP** and **SEUP**. We vary the number of topics in both methods from 5 to 30, the results are shown in Figure 1, using Wikipedia as the external corpus in the **user-SMALL** group (we eliminated other results as they gave similar results). As can be seen from the figure, the highest performance is reached when the number of latent topics is 20 in **MEUP** and 25 in **SEUP**. When the number of topics continues to grow, the performance starts to degrade. However, even the lowest scored run still outperformed the strongest baseline. By varying the topic numbers, **SEUP** still outperforms **MEUP**.

7 Conclusion and Future Work

In this paper, we tackle the challenge of personalized web search using social data in a novel way by building enhanced user profiles from the annotations and resources the user has marked, together with an external knowledge base. We present two probabilistic latent models to simultaneously incorporate social annotations, documents and the external knowledge base. In addition, we introduce a topical query expansion model to enhance the search by utilizing individual user profiles. The proposed methods performed well on the social data crawled from the web, delivering statistically significant improvements over non-personalized and personalized representative baseline systems by constructing user profiles from a user's historical usage information only. It is also confirmed that our proposed methods work well for both active and less active users. In future research, we aim to automatically estimate the number of topics to be used in the EUP models. We also plan to explore the use of more external resources and novel latent semantic models to enhance performance.

Acknowledgements

The work described in this paper was supported by the National Natural Science Foundation of China under Project No. 61300129, No. 61572187 and No. 61272063, Scientific Research Fund of Hunan Provincial Education Department of China under Grant No. 16K030, Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, China under grant No. [2013] 1792, Hunan Provincial Innovation Foundation For Postgraduate under grant No. CX2016B575. This work is also supported by the ADAPT Centre for Digital Content Technology, which is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

M. Bender, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, R. Schenkel and G. Weikum (2008). Exploiting social relations for query expansion and result ranking. In *Proceedings of the IEEE 24th International Conference on Data Engineering Workshop, ICDEW 2008,* Chicago, IL, USA, IEEE. p. 501-506.

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk and Xiaoyuan Cui (2012). Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, Portland, Oregon, USA, ACM. p. 185-194.

Marin Bertier, Rachid Guerraoui, Vincent Leroy and Anne-Marie Kermarrec (2009). Toward personalized query expansion. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, Nuremberg, Germany, ACM. p. 7-12.

Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli and Giuseppe Sansonetti (2013). Social semantic query expansion. *ACM Transactions on Intelligent Systems and Technology,* **4**(4): 1-43.

Claudio Biancalana and Alessandro Micarelli (2009). Social Tagging in Query Expansion: A New Way for Personalized Web Search. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, IEEE. p. 1060-1065.

Mohamed Reda Bouadjenek, Hakim Hacid and Mokrane Bouzeghoub (2013a). Sopra: a new social personalized ranking function for improving web search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland, ACM. p. 861-864.

Mohamed Reda Bouadjenek, Hakim Hacid and Mokrane Bouzeghoub (2016). Social networks and information retrieval, how are they converging? A survey, a taxonomy and an analysis of social information retrieval approaches and platforms. *Information Systems* **56**: 1-18.

Mohamed Reda Bouadjenek, Hakim Hacid, Mokrane Bouzeghoub and Johann Daigremont (2011). Personalized social query expansion using social bookmarking systems. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, Beijing, China, ACM. p. 1113-1114.

Mohamed Reda Bouadjenek, Hakim Hacid, Mokrane Bouzeghoub and Athena Vakali (2013b). Using social annotations to enhance document representation for personalized search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland, ACM. p. 1049-1052.

Mark J. Carman, Mark Baillie and Fabio Crestani (2008). Tag data and personalized information retrieval. In *Proceeding of the 2008 ACM workshop on Search in social media*, Napa Valley, California, USA, ACM. p. 27-34.

David Carmel, Naama Zwerdling, Ido Guy, Shila Ofek-Koifman, Nadav Har'el, Inbal Ronen, Erel Uziel, Sivan Yogev and Sergey Chernov (2009). Personalized social search based on the user's social network. In *Proceeding of the 18th ACM conference on Information and knowledge management*, Hong Kong, China, ACM. p. 1227-1236.

Paul - Alexandru Chirita, Claudiu S. Firan and Wolfgang Nejdl (2007). Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, ACM. p. 7-14.

Fernando Diaz and Donald Metzler (2006). Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, USA, ACM. p. 154-161.

Debasis Ganguly, Johannes Leveling and GarethJ F. Jones (2012). Topical Relevance Model. *Information Retrieval Technology*. Yuexian Hou, Jian-Yun Nie, Le Sun, Bo Wang and Peng Zhang, Springer Berlin Heidelberg. **7675:** 326-335.

Ramanathan Guha, Vineet Gupta, Vivek Raghunathan and Ramakrishnan Srikant (2015). User Modeling for a Personal Assistant. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, Shanghai, China, ACM. p. 275-284.

Manish Gupta, Rui Li, Zhijun Yin and Jiawei Han (2011). An Overview of Social Tagging and Applications. *Social Network Data Analytics*. Charu C. Aggarwal, Springer US**:** 447-497.

Ronald H Heck and Scott L Thomas (2015). *An Introduction to Multilevel Modeling Techniques: MLM and SEM Approaches Using Mplus*, Routledge.

Mohsen Jamali and Martin Ester (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, Barcelona, Spain, ACM. p. 135-142.

Kalervo Järvelin and Jaana Kekäläinen (2000). IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, Athens, Greece, ACM. p. 41-48.

Victor Lavrenko and W. Bruce Croft (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, Louisiana, USA, ACM. p. 120-127.

Jovian Lin, Kazunari Sugiyama, Min-Yen Kan and Tat-Seng Chua (2013). Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland, ACM. p. 283-292.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (2008). *Introduction to Information Retrieval*, Cambridge University Press.

MichaelG Noll and Christoph Meinel (2007). Web Search Personalization Via Social Bookmarking and Tagging. *The Semantic Web*. Karl Aberer, Key-Sun Choi, Natasha Noyet al, Springer Berlin Heidelberg. **4825:** 367-380.

Kazunari Sugiyama, Kenji Hatano and Masatoshi Yoshikawa (2004). Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA, ACM. p. 675-684.

Jaime Teevan, Susan T. Dumais and Eric Horvitz (2005). Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, ACM. p. 449-456.

David Vallet, Iván Cantador and Joemon M Jose (2010). Personalizing web search with folksonomy-based user and document profiles. In *Proceedings of the 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK,* Springer. p. 420-431.

E. M. Voorhees (1999). The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*.

Qihua Wang and Hongxia Jin (2010). Exploring online social activities for adaptive search personalization. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, Toronto, ON, Canada, ACM. p.999-1008.

Shengliang Xu, Shenghua Bao, Ben Fei, Zhong Su and Yong Yu (2008). Exploring folksonomy for personalized search. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, Singapore, ACM. p. 155-162.

Chengxiang Zhai and John Lafferty (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, ACM. p. 403-410.

D. Zhou, S. Lawless, J. Liu, S. Zhang and Y. Xu (2015). Query expansion for personalized cross-language information retrieval. In *Proceedings of the 10th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP),* Trento, Italy, IEEE. p. 1-5.

Dong Zhou, Séamus Lawless and Vincent Wade (2012a). Improving search via personalized query expansion using social media. *Information Retrieval,* **15**(3-4): 218-242.

Dong Zhou, Séamus Lawless and Vincent Wade (2012b). Web Search Personalization Using Social Data. In *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries, TPDL 2012, Paphos, Cyprus,* Springer. p. 298-310.

Dong Zhou, Séamus Lawless, Xuan Wu, Wenyu Zhao and Jianxun Liu (2016). A study of user profile representation for personalized cross-language information retrieval. *Aslib Journal of Information Management,* **68**(4): 448-477.

Dong Zhou, Mark Truran, Jianxun Liu, Wei Li and Gareth Jones (2014). Iterative Refinement Methods for Enhanced Information Retrieval. *International Journal of Intelligent Systems,* **29**(4): 341-364.

Arkaitz Zubiaga, Victor Fresno, Ricardo Martinez and Alberto Perez Garcia-Plaza (2013). Harnessing folksonomies to produce a social classification of resources. *IEEE Transactions on Knowledge and Data Engineering,* **25**(8): 1801-1813.

Arkaitz Zubiaga, Alberto Pérez Garcia-Plaza, Víctor Fresno and Ricardo Martinez (2009). Content-based clustering for tag cloud visualization. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining, ASONAM*, IEEE. p. 316-319.

Effective Greedy Inference for Graph-based Non-Projective Dependency Parsing

Ilan Tchernowitz **Liron Yedidsion** **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
{ilantc@campus|lirony@ie|roiri@ie}.technion.ac.il

Abstract

Exact inference in high-order graph-based non-projective dependency parsing is intractable. Hence, sophisticated approximation techniques based on algorithms such as belief propagation and dual decomposition have been employed. In contrast, we propose a simple greedy search approximation for this problem which is very intuitive and easy to implement. We implement the algorithm within the second-order TurboParser and experiment with the datasets of the CoNLL 2006 and 2007 shared task on multilingual dependency parsing. Our algorithm improves the run time of the parser by a factor of 1.43 while losing 1% in UAS on average across languages. Moreover, an ensemble method exploiting the joint power of the parsers, achieves an average UAS 0.27% higher than the TurboParser.

1 Introduction

Dependency parsing is instrumental in NLP applications, with recent examples in information extraction (Wu and Weld, 2010), word embeddings (Levy and Goldberg, 2014), and opinion mining (Almeida et al., 2015). The two main approaches for this task are graph based (McDonald et al., 2005) and transition based (Nivre et al., 2007).

The graph based approach aims to optimize a global objective function. While exact polynomial inference algorithms exist for projective parsing (Eisner, 1996; McDonald et al., 2005; Carreras, 2007; Koo and Collins, 2010, inter alia), high order non-projective parsing is NP-hard (McDonald and Pereira, 2006). The current remedy for this comes in the form of advanced optimization techniques such as dual decomposition (Martins et al., 2013), LP relaxations (Riedel et al., 2012), belief propagation (Smith and Eisner, 2008; Gormley et al., 2015) and sampling (Zhang et al., 2014b; Zhang et al., 2014a).

The transition based approach (Zhang and Nivre, 2011; Bohnet and Nivre, 2012; Honnibal et al., 2013; Choi and McCallum, 2013a, inter alia), and the easy first approach (Goldberg and Elhadad, 2010) which extends it by training non-directional parsers that consider structural information from both sides of their decision points, lack a global objective function. Yet, their sequential greedy solvers are fast and accurate in practice.

We propose a greedy search algorithm for high-order, non-projective graph-based dependency parsing. Our algorithm is a simple iterative graph-based method that does not rely on advanced optimization techniques. Moreover, we factorize the graph-based objective into a sum of terms and show that our basic greedy algorithm relaxes the global objective by sequentially optimizing these terms instead of globally optimizing their sum.

Unlike previous greedy approaches to dependency parsing, transition based and non-directional, our algorithm does not require a specialized feature set or a training method that specializes in local decisions. In contrast, it supports global parameter training based on the comparison between an induced tree and the gold tree. Hence, it can be integrated into any graph-based parser.

We first present a basic greedy algorithm that relaxes the global graph-based objective (Section 3). However, as this simple algorithm does not provide a

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 711–720,
Austin, Texas, November 1-5, 2016.

realistic estimation of the impact of an arc selection on uncompleted high-order structures in the partial parse forest, it is not competitive with state of the art approximations. We hence present an advanced version of our algorithm with an improved arc score formulation and show that this simple algorithm provides high quality solutions to the graph-based inference problem (Section 4).

Particularly, we implement the algorithm within the TurboParser (Martins et al., 2013) and experiment (Sections 8 and 9) with the datasets of the CoNLL 2006-2007 shared tasks on multilingual dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007). On average across languages our parser achieves UAS scores of 87.78% and 89.25% for first and second order parsing respectively, compared to respective UAS of 87.98% and 90.26% achieved by the original TurboParser.

We further implement (Section 6) an ensemble method that integrates information from the output tree of the original TurboParser and the arc weights learned by our variant of the parser into our search algorithm to generate a new tree. This yields an improvement: average UAS of 88.03% and 90.53% for first and second parsing, respectively.

Despite being greedy, the theoretical runtime complexity of our advanced algorithm is not better than the best previously proposed approximations for our problem ($O(n^{k+1})$, for n word sentences and k order parsing, Section 5). In experiments, our algorithms improve the runtime of the TurboParser by a factor of up to 2.41.

The main contribution of this paper is hence in providing a simple, intuitive and easy to implement solution for a long standing problem that has been addressed in past with advanced optimization techniques. Besides the intellectual contribution, we believe this will make high-order graph-based dependency parsing accessible to a much broader research and engineering community as it substantially relaxes the coding and algorithmic proficiency required for the implementation and understanding of parsing algorithms.

2 Problem Formulation

We start with a brief definition of the high order graph-based dependency parsing problem. Given an n word input sentence, an input graph $G = (V, E)$ is defined. The set of vertices is $V = \{0, ..., n\}$, with the $\{1, \ldots, n\}$ vertices representing the words of the sentence, in their order of appearance, and the 0 vertex is a specialized *root* vertex. The set of arcs is $E = \{(u, v) : u \in \{0, ..., n\}, v \in \{1, ..., n\}, u \neq v\}$, that is, the root vertex has no incoming arcs.

We further define a part of order k to be a subset of E of size k, and denote the set of all parts with *parts*. For the special case of $k = 1$ a part is an arc. Different works employed different $parts$ sets (e.g. (Martins et al., 2013; McDonald et al., 2005; Koo and Collins, 2010)). Generally, most *parts* sets consist of arcs connecting vertices either vertically (e.g. $\{(u, v), (v, z)\}$ for $k = 2$) or horizontally (e.g. $\{(u, v), (u, z)\}$, for $k = 2$). In this paper we focus on the parts employed by (Martins et al., 2013), a state-of-the-art parser, but our algorithms are generally applicable for any *parts* set consistent with this general definition.[1]

In graph-based dependency parsing, each part p is given a score $W_p \in \mathcal{R}$. A Dependency Tree (*DT*) T is a subset of arcs for which the following conditions hold: (1) Every vertex, except for the root, has an incoming arc: $\forall v \in V \setminus \{0\} : \exists u \in V s.t. (u, v) \in T$; (2) No vertex has multiple incoming arcs: $\forall (u, u', v) \in V, (u, v) \in T \rightarrow (u', v) \notin T$; and (3) There are no cycles in T. The score of a DT T is finally defined by:

$$score(T) = \sum_{part \subseteq T} W_{part}$$

The inference problem in this model is to find the highest scoring DT in the input weighted graph.

3 Basic Greedy Inference

We start with a basic greedy algorithm (Algorithm 1), analyze the approximation it provides for the graph-based objective and its inherent limitations.

[1] More generally, a part is defined by two arc subsets, A and B, such that a part p belongs to a tree T if $\forall e \in A : e \in T$ and $\forall e \in B : e \notin T$. In this paper we assume $B = \phi$. Hence, we cannot experiment with the third order TurboParser as in all its third order parts $B \neq \phi$. Also, when we integrate our algorithms into the second order TurboParser we omit the nextSibling part for which $B \neq \phi$. For the original TurboParser to which we compare our results, we do not omit this part as it improves the parser's performance.

Algorithm 1 maintains a *partial tree* data structure, T^i, to which it iteratively adds arcs from the input graph G, one in each iteration, until a dependency tree T^n is completed. For this end, in every iteration, i, a value, v_e^i, composed of $loss_e^i$ and $gain_e^i$ terms, is computed for every arc $e \in E$ and the arc with the lowest v_e^i value is added to T^{i-1} to create the extended partial tree T^i.

Due to the aforementioned conditions on the induced dependency tree, every arc that is added to T^{i-1} yields a set of *lostArcs* and *lostParts* that cannot be added to the partial tree in subsequent iterations. The loss value is defined to be:

$$loss_e^i := \sum_{part \in lostParts} W_{part}$$

That is, every part that contains one or more arcs that violate the dependency tree conditions for a tree that extends the partial tree $T^{i-1} \cup \{e\}$ is considered a *lost part* as it will not be included in any tree extending $T^{i-1} \cup \{e\}$. The loss value sums the weights of these parts.

Likewise, the gain value is the sum of the weights of all the parts that are added to T^{i-1} when we add e to it. Denote this set of parts with $P_e := \{part : part \subseteq T^{i-1} \cup \{e\}, part \notin T^{i-1}\}$, then:

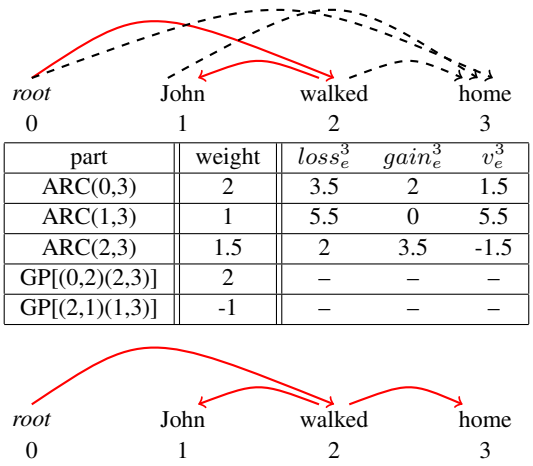
$$gain_e^i := \sum_{part \in P_e} W_{part}$$

Finally, v_e^i is given by:

$$v_e^i = loss_e^i - gain_e^i$$

After the arc with the minimal value v_e^i is added to T^{i-1}, the arcs that violate the structural constraints on dependency trees are removed from G.

An example of an update iteration of the algorithm (lines 3-16) is given in Figure 1. In this example we consider two types of parts: first-order, arc, parts (ARC) and second-order grandparent parts (GP), consisting of arc pairs, $\{(g,u),(u,v)\}$. The upper graph shows the partial tree T^2 (solid arcs) as well as the rest of the graph G (dashed arcs). The parts included in T^2 are ARC(0,2), ARC(2,1) and GP[(0,2),(2,1)]. The table contains the weights of the parts and the values computed for the arcs during the third iteration. The arc that is chosen is (2,3), as it has the minimal v_e^3 value. Thus, in the bottom



part	weight	$loss_e^3$	$gain_e^3$	v_e^3
ARC(0,3)	2	3.5	2	1.5
ARC(1,3)	1	5.5	0	5.5
ARC(2,3)	1.5	2	3.5	-1.5
GP[(0,2)(2,3)]	2	–	–	–
GP[(2,1)(1,3)]	-1	–	–	–

Figure 1: An example of an iteration of Algorithm 1 (lines 3-16)). See description in text.

graph that corresponds to T^3 all other incoming arcs to vertex 3 are removed. (in this instance there are no cycle forming arcs).

Analysis We now turn to an analysis of the relaxation that Algorithm 1 provides for the global graph-based objective. Recall that our objective in iteration i is: $v_{e^i}^i = min\{v_e^i\}$. For the inferred tree T^n it holds that:

$$\sum_{e^i \in T^n} v_{e^i}^i - \sum_{part \subseteq G} W_{part} =$$

$$\sum_{part \nsubseteq T^n} W_{part} - \sum_{part \subseteq T^n} W_{part} - \sum_{part \subseteq G} W_{part} =$$

$$-2 \times \sum_{part \subseteq T^n} W_{part} + \sum_{part \nsubseteq T^n} W_{part} - \sum_{part \nsubseteq T^n} W_{part} =$$

$$-2 \times \sum_{part \subseteq T^n} W_{part}$$

The first equation holds since $\sum_{e^i \in T^n} v_{e^i}^i$ is the sum of all lost parts (parts that are not in T^n) minus all the gained parts (parts in T^n). Each of these parts was counted exactly once: when the part was added to the partial tree or when one of its arcs was removed from G. The second equation splits the term of $\sum_{part \subseteq G} W_{part}$ to two sums, one over parts in T^n and the other over the rest. Since $\sum_{part \subseteq G} W_{part}$ and 2 are constants, we get:

$$\arg\min_{T^n}(-\sum_{part \subseteq T^n} W_{part}) = \arg\min_{T^n} \sum_{e^i \in T^n} v_{e^i}^i$$

From this argument it follows that our inference algorithm performs sequential greedy optimization over the presented factorization of the graph-based

objective instead of optimizing the sum of terms, and hence the objective, globally.

The main limitation of Algorithm 1 is that it does not take into account high order parts contribution until the part is actually added to T. For example, in Figure 1, when the arc $(2,1)$ is added, the part GP[(2,1),(1,3)] is getting closer to completion. Yet, this is not taken into account when considering whether $(2,1)$ should be added to the tree or not. Including this information in the gain and loss values of an arc can improve the accuracy of the algorithm, especially in high-order parsing.

Algorithm 1 Basic Greedy Inference

1: $T^0 = \{\}$
2: **for** $i \in 1..n$ **do**
3: **for** $e = (u, v) \in E$ **do**
4: $P_e := \{part \in parts : part \subseteq T^{i-1} \cup \{e\}, part \nsubseteq T^{i-1}\}$
5: $gain_e^i := \sum_{part \in P_e} W_{part}$
6: $incomingSet := \{(u', v) \in E : u' \neq u\}$
7: $cycleSet := \{(u', v') \in E : T^{i-1} \cup \{e\} \cup (u', v')\ contains\ a\ cycle\}$
8: $lostArcs = (incomingSet \cup cycleSet)$
9: $lostParts = \{part : \exists e \in lostArcs \cap part\}$
10: $loss_e^i := \sum_{part \in lostParts} W_{part}$
11: $v_e^i := loss_e^i - gain_e^i$
12: **end for**
13: $e^i = (u^i, v^i) = \arg\min_{e'}\{v_{e'}^i\}$
14: $T^i = T^{i-1} \cup \{e^i\}$
15: remove from G all incoming arcs to v^i
16: remove from G all cycle forming arcs w.r.t T^i
17: **end for**

4 Greedy Inference with Partial Part Predictions

In order for the algorithm to account for information about partial high order parts, we estimate the probability that such parts would be eventually included in T^n. Our way to do this (Algorithm 2) is by estimating these probabilities for arcs and from these derive parts probabilities.

Particularly, for the set of incoming arcs of a vertex v, $E_v = \{e = (u, v) : e \in E\}$, a probability measure p_e is computed according to:

$$p_{e=(u,v)} = \frac{exp^{\alpha \times W_e}}{\sum_{e'=(u',v)} exp^{\alpha \times W_{e'}}}$$

Where α is a hyper parameter of the model. For $\alpha = 0$ we get a uniform distribution over all possible heads of a vertex v, and for large α values arcs with larger weights get higher probabilities.

Algorithm 2 Greedy Inference with Partial Part Predictions

1: $T^0 = \{\}$
2: **for** $i \in 1..n$ **do**
3: **for** $e = (u, v) \in E$ **do**
4: $P_e := \{part \in parts : e \in part\}$
5: $gain_e^i := \sum_{part \in P_e} W_{part} \times p_{part}|(T^{i-1} \cup \{e\})$
6: $incomingSet := \{(u', v) \in E : u' \neq u\}$
7: $cycleSet := \{(u', v') \in E : T^{i-1} \cup \{e\} \cup (u', v')\ contains\ a\ cycle\}$
8: $lostArcs = (incomingSet \cup cycleSet)$
9: $lostParts = \{part : \exists e \in lostArcs \cap part\}$
10: $loss_e^i := \sum_{part \in lostParts} W_{part} \times p_{part}|T^{i-1}$
11: $v_e^i := \beta \times loss_e^i - (1 - \beta) \times gain_e^i$
12: **end for**
13: $e^i = (u^i, v^i) = \arg\min_{e'}\{v_{e'}^i\}$
14: $T^i = T^{i-1} \cup \{e^i\}$
15: remove from G all incoming arcs to v^i
16: remove from G all cycle forming arcs w.r.t T^i
17: **end for**

The intuition behind this measure is that arcs mostly compete with other arcs that have the same target vertex and hence their weight should be normalized accordingly. Using this measure, we define the arc-factored probability of a part to be:

$$p_{part} = \prod_{e \in part} p_e$$

And the residual probability of a part given an existing partial tree T:

$$p_{part}|T = \frac{p_{part}}{\prod_{e \in (part \bigcap T)} p_e}$$

These probability measures are used in both the gain and the loss computations (lines 5 and 10 in Algorithm 2) as follows:

$$gain_e^i := \sum_{part: e \in part} W_{part} \times p_{part}|(T^{i-1} \cup \{e\})$$

$$loss_e^i := \sum_{part \in lostParts} W_{part} \times p_{part}|T^{i-1}$$

Finally, as adding an arc to the dependency subtree results in an exclusion of several arcs, the number of lost parts is also likely to be much higher than the number of gained parts. In order to compensate for this effect, we introduce a balancing

hyper-parameter, $\beta \in [0,1]$, and change the computation of v_e^i (line 10 in Algorithm 2) to be: $v_e^i := \beta \times loss_e^i - (1-\beta) \times gain_e^i$.

5 Runtime Complexity Analysis

In this section we provide a sketch of the runtime complexity analysis of the algorithm. Full details are in appendix A. In what follows, we denote the maximal indegree of a vertex with n_{in}.

Algorithm 1 Algorithm 1 consists of two nested loops (lines 2-3) and hence lines 4-11 are repeated $O(n \times |E|) = O(n \times n \times n_{in})$ times. At each repetition, loss (lines 6-10) and gain (lines 4-5) values are computed. Afterwards the graph's data structures are updated (lines 13-16). We define data structures (DSs) that keep our computations efficient. With these DSs the total runtime of lines 4-11 is $O(n_{in} + min\{n_{in}^{k-1}, n_{in}^2\})$. The DSs are initialized in $O(|parts| \times k)$ time and their total update time is $O(k \times |parts|) = O(n_{in}^{k+1})$. Thus algorithm 1 runs in $O(|parts| \times k + n^2 \times n_{in} \times (n_{in} + min\{n_{in}^{k-1}, n_{in}^2\}))$ time.

Algorithm 2 Algorithm 2 is similar in structure to Algorithm 1. The enhanced loss and gain computations take $O(n_{in} + min\{n_{in}^{k-1}, n_{in}^2\})$ time. The initialization of the DSs takes $O(|parts| \times k)$ time and their update time is $O(n_{in}^k \times k^2)$. The total runtime of Algorithm 2 is $O(n_{in}^{k+1} \times k + n \times (n \times n_{in} \times (n_{in} + min\{n_{in}^2, n_{in}^{k-1}\}) + n_{in}^k \times k^2))$. For unpruned graphs and $k \geq 2$ this is equivalent to $O(n^{k+1})$, the theoretical runtime of the TurboParser's dual decomposition inference algorithm.

6 Error Propagation

Unlike modern approximation algorithms for our problem, our algorithm is greedy and deterministic. That is, in each iteration it selects an arc to be included in its final dependency tree and this decision cannot be changed in subsequent iterations. Hence, our algorithm is likely to suffer from error-propagation. We propose two solutions to this problem described within Algorithm 2.

Beam search In each iteration (lines 3-16) the algorithm outputs its $|B|$ best solutions to be subsequently considered in the next iteration. That is, lines 4-10 are performed $|B|$ times for each edge $e \in E$, one for each of the $|B|$ partial solutions in the beam, $b^j \in B$. For each such solution, we denote its weight, as calculated by the previous iteration of the algorithm with $beamVal_{b^j}$. When evaluating v_e^i for an arc e with respect to b^j (line 11), we set $v_e^{i,j} = beamVal_{b^j} + \beta \times loss_e^i - (1-\beta) \times gain_e^i$.

Post-search improvements After Algorithm 2 is executed, we perform s iterations of local greedy arc swaps. That is, for every vertex v, s.t. $(u,v) \in T^n$, we try to switch the arc (u,v) with the arc (u',v) as follows. Let T_v^n be the sub tree that is rooted at v, we distinguish between two cases:
(1) If $u' \notin T_v^n$ then $T^n = T^n \setminus \{(u,v)\} \cup \{(u',v)\}$.
(2) If $u' \in T_v^n$ then let w be the first vertex on the path from v to u' (if $(v,u') \in T$ then $w = u'$): $T^n = T^n \setminus \{(u,v),(v,w)\} \cup \{(u',v),(u,w)\}$.

After inspecting all possible substitutions, we choose the one that yields the best increase in the tree score (if such a substitution exists) and perform the substitution.

7 Parser Combination

In our experiments (see below), we implemented our algorithms within the TurboParser so that each of them, in turn, serves as its inference algorithm. In development data experiments with Algorithm 2 we found that for first order parsing, both our algorithm and the TurboParser predict on average over all languages around 1% of the gold arcs that are not included in the output of the other algorithm. For second order parsing, the corresponding numbers are 1.75% (for gold arcs in the output of our algorithm but not of the original TurboParser) and 4.3% (for the other way around). This suggests that an ensemble method may improve upon both parsers.

We hence introduce a variation of Algorithm 2 that accepts a dependency tree T_o as an input, and biases its output towards that tree. As different parsers usually generate weights on different scales, we do not directly integrate part weights. Instead, we change the weight of each part $part \subseteq T_o$ of order j, to be $W_{part} = W_{part} + \gamma_j$, where γ_j is an hyperparameter reflecting our belief in the prediction of the other parser on parts of order j. The change is applied only at test time, thus integrating two pre-trained parsers.

8 Experimental Setup

We implemented our algorithms within the TurboParser (Martins et al., 2013)[2]. That is, every other aspect of the parser – feature set, pruning algorithm, cost-augmented MIRA training (Crammer et al., 2006) etc., is kept fixed but our algorithms replace the inference algorithms: Chu-Liu-Edmonds ((Edmonds, 1967), first order) and dual-decomposition (higher order). We implemented two variants, for algorithm 1 and 2 respectively, and compare their results to those of the original TurboParser.

We experiment with the datasets of the CoNLL 2006 and 2007 shared task on multilingual dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007), for a total of 17 languages. When a language is represented in both sets, we used the 2006 version. We followed the standard train/test split of these datasets and, for the 8 languages with a training set of at least 10000 sentences, we randomly sampled 1000 sentences from the training set to serve as a development set. For these languages, we first trained the parser on the training set and then used the development set for hyperparameter tuning ($|B|$, s, α, β, and $\gamma_1, \ldots, \gamma_k$ for k order parsing).[3][4]

We employ four evaluation measures, where every measure is computed per language, and we report the average across all languages: (1) Unlabeled Attachment Score (UAS); (2) Undirected UAS (U-UAS) - for error analysis purposes; (3) Shared arcs (SARC) - the percentage of arcs shared by the predictions of each of our algorithms and of the original TurboParser; and (4) Tokens per second (TPS) - for ensemble models this measure includes the TurboParser's inference time.[5] We also report a *gold(x,y) = (a,b)* measure: where a is the percentage of gold standard arcs included in trees produced by algorithm x but not by y, and b is the corresponding number for y and x. We consider two setups.

Fully Supervised Training In this setup we only consider the 8 languages with a development set. For each language, the parser is trained on the training set and then the hyperparameters are tuned. First we set the beam size ($|B|$) and number of improvement iterations (s) to 0, and tune the other hyperparameters on the language-specific development set. Then, we tune $|B|$ and s, using the optimal parameters of the first step, on the English dev. set.

Minimally Supervised Training Here we consider all 17 languages. For each language we randomly sampled 20 training sets of 500 sentences from the original training set, trained a parser on each set and tested on the original test set. Results for each language were calculated as the average over these 20 folds. The hyper parameters for all languages were tuned once on the English development set to the values that yielded the best average results across the 20 training samples.

9 Results

Fully Supervised Training Average results for this setup are presented in table 1 (top). Unsurprisingly, UAS for second order parsing with basic greedy inference (Algorithm 1, BGI) is very low, as this model does not take information about partial high order parts into account in its edge scores. We hence do not report more results for this algorithm.

The table further reflects the accuracy/runtime tradeoff provided by Algorithm 2 (basic greedy inference with partial part predictions, BGI-PP): a UAS degradation of 0.34% and 2.58% for first and second order parsing respectively, with a runtime improvement by factors of 1.01 and 2.4, respectively. Employing beam search and post search improvements (BGI-PP+i+b) to compensate for error propagation improves UAS but harms the runtime gain: for example, the UAS gap in second order parsing is 1.01% while the speedup factor is 1.43.

As discussed in footnote 1 and Section 11, our algorithm does not support the third-order parts of the TurboParser. However, the average UAS of the third-order TurboParser is 90.62% (only 0.36% above second order TurboParser) and its TPS is 72.12 (almost 5 times slower).

The accuracy gaps according to UAS and undirected UAS are similar, indicating that the source

[2] https://github.com/andre-martins/TurboParser

[3] $|B| = 3$, $s = 5$, $\alpha \in [0, 2.5]$, $\beta \in [0.2, 0.5]$, $\gamma_1 \in [0.5, 1.5]$, $\gamma_2 \in [0.2, 0.3]$. Our first order part weights are in $[-9, 4]$, and second order part weights in $[-3, 13]$.

[4] The original TurboParser is trained on the training set of each language and tested on its test set, without any further division of the training data to training and development sets.

[5] Run times where computed on an Intel(R) Xeon(R) CPU E5-2697 v3@2.60GHz machine with 20GB RAM memory.

Fully supervised		Individual Models				Ensemble Models			
		UAS	TPS	SARC	U-UAS	UAS	TPS	SARC	U-UAS
TurboParser	order1	87.98	5621.30	–	88.82	–	–	–	–
	order2	90.26	356.63	–	90.98	–	–	–	–
BGI	order1	83.78	5981.91	90.87	90.87	–	–	–	–
	order2	27.54	715.41	27.76	27.77	–	–	–	–
BGI-PP	order1	87.64	5680.60	97.15	88.53	88.03	2876.03	99.59	88.84
	order2	87.68	858.25	92.66	88.73	90.50	249.40	99.54	91.20
BGI-PP + i	order1	87.76	4648.4	98.10	88.64	87.96	2557.00	99.47	88.80
	order2	88.98	639.97	94.40	89.81	90.50	297.10	99.43	91.19
BGI-PP + i + b	order1	87.78	3253.80	98.29	88.73	87.91	2053.00	99.07	88.82
	order2	89.25	511.47	94.79	90.02	90.53	212.40	99.40	91.21

(a) The fully supervised setup.

Minimally supervised		Individual Models				Ensemble Models			
		UAS	TPS	SARC	U-UAS	UAS	TPS	SARC	U-UAS
TurboParser	order1	78.99	13097.00	–	80.38	–	–	–	
	order2	80.52	830.05	–	81.84	–	–	–	
BGI-PP	order1	78.76	13848.00	85.36	80.15	79.14	6499.00	87.36	80.50
	order2	78.80	3089.40	84.59	80.27	80.60	636.30	95.57	81.88
BGI-PP + i	order1	78.87	11673.00	85.54	80.25	79.24	6516.00	87.55	80.59
	order2	79.36	2414.00	84.81	80.76	80.67	621.50	95.41	82.16
BGI-PP + i + b	order1	78.91	4212.50	85.58	80.29	79.29	4349.00	87.61	80.62
	order2	79.45	1372.70	84.89	80.84	80.69	518.10	95.44	81.96

(b) The minimally supervised setup.

Table 1: Results for the fully supervised (top table) and minimally supervised (bottom table) setups. The left column section of each table is for individual models while the right column section is for ensemble models (Section 7). BGI-PP is the basic greedy inference algorithm with partial part predictions, +i indicates post-search improvements and +b indicates beam search (Section 6). The Tokens per Second (TPS) measure for the ensemble models reports the additional inference time over the TurboParser inference. All scores are averaged across individual languages.

of differences between the parsers is not arc directionality. The percentage of arcs shared between the parsers increases with model complexity but is still as low as 94.79% for BGI-PP+i+b in second order parsing. In this setup, gold(BGI-PP+i+b, TurboParser) = (1.6%,2.6%) which supports the development data pattern reported in Section 6 and further justifies an ensemble approach.

The right column section of the table indeed shows consistent improvements of the ensemble models over the TurboParser for second order parsing: the ensemble models achieve UAS of 90.5-90.53% compared to 90.26% of the TurboParser. Naturally, running the TurboParser alone is faster by a factor of 1.67. Like for the individual inference algorithms, the undirected UAS measure indicates that the gain does not come from arc directionality improvements. The ensemble methods share almost all of their arcs with the TurboParser, but in cases of disagreement ensembles tend to be more accurate.

Table 2 complements our results, providing UAS values for each of the 8 languages participating in this setup. The UAS difference between BGI+PP+i+b and the TurboParser are (+0.24)-(-0.71) in first order parsing and (+0.18)-(-2.46) in second order parsing. In the latter case, combining these two models (BGI+PP+i+b+e) yields improvements over the TurboParser in 6 out of 8 languages.

Minimally Supervised Training Results for this setup are in table 1 (bottom). While result patterns are very similar to the fully supervised case, two observations are worth mentioning. First, the percentage of arcs shared by our algorithms and the original parser is much lower than in the fully supervised case. This is true also for shared gold arcs: gold(BGI-PP+b+i,TurboParser) = (4.86%,5.92%) for second order parsing. This suggests that more sophisticated ensemble techniques may be useful in this setup.

Second, ensemble modeling improves UAS over the TurboParser also for first order parsing, leading to a gain of 0.3% in UAS for the BGI+i+b ensemble (79.29% vs. 78.99%). As the percentage of shared arcs between the ensemble models and the TurboParser is particularly low in first order parsing, as well as the shared gold arcs

language	First Order				Second Order			
	TurboParser	BGI-PP	BGI-PP +i+b	BGI-PP +i+b+e	TurboParser	BGI-PP	BGI-PP +i+b	BGI-PP +i+b+e
swedish	**87.12**	86.35	86.93	**87.12**	88.65	86.14	87.85	**89.29**
bulgarian	**90.66**	90.22	90.42	**90.66**	92.43	89.73	91.50	**92.58**
chinese	**84.88**	83.89	84.17	84.17	86.53	81.33	85.18	**86.59**
czech	**83.53**	83.46	83.44	83.44	86.35	84.91	86.26	**87.50**
dutch	88.48	**88.56**	88.43	88.43	91.30	89.64	90.49	**91.34**
japanese	93.03	93.18	**93.27**	**93.27**	93.83	93.78	**94.01**	**94.01**
catalan	**88.94**	88.50	88.67	88.93	**92.25**	89.3	90.46	92.24
english	**87.18**	86.94	86.84	**87.18**	**90.70**	86.52	88.24	90.66

Table 2: Per language UAS for the fully supervised setup. Model names are as in Table 1, 'e' stands for ensemble. Best results for each language and parsing model order are highlighted in bold.

(gold(BGI+i+b,TurboParser) = (4.98%,5.5%)), improving the ensemble techniques is a promising future research direction.

10 Related Work

Our work brings together ideas that have been considered in past, although in different forms.

Greedy Inference Goldberg and Elhadad (2010) introduced an easy-first, greedy, approach to dependency parsing. Their algorithm adds at each iteration the best candidate arc, in contrast to the left to right ordering of standard transition based parsers. This work is extended at (Tratz and Hovy, 2011; Goldberg and Nivre, ; Goldberg and Nivre, 2013).

The easy-first parser consists of a feature set and a specialized variant of the structured perceptron training algorithm, both dedicated to greedy inference. In contrast, we show that a variant of the TurboParser that employs Algorithm 2 for inference and is trained with its standard global training algorithm, performs very similarly to the same parser that employs dual decomposition inference.

Error Propagation in Deterministic Parsing Since deterministic algorithms are standard in transition-based parsing, the error-propagation problem has been dealt with in that context. Various methods were employed, with beam search being a prominent idea (Sagae and Lavie, 2006; Titov and Henderson, 2007; Zhang and Clark, 2008; Huang et al., 2009; Zhang and Nivre, 2011; Bohnet and Nivre, 2012; Choi and McCallum, 2013b, inter alia).

Post Search Improvements Several previous works employed post-search improvements techniques. Like in our case, these techniques improve the tree induced by an initial, possibly more principled, search technique through local, greedy steps.

McDonald and Pereira (2006) proposed to approximate high-order graph-based non-projective parsing, by arc-swap iterations over a previously induced projective tree. Levi et al. (2016) proposed a post-search improvements method, different than ours, to compensate for errors of their graph-based, undirected inference algorithm. Finally, Zhang et al. (2014a) demonstrated that multiple random initialization followed by local improvements with respect to a high-order parsing objective result in excellent parsing performance. Their algorithm, however, shouldbhhb employ hundreds of random initializations in order to provide state-of-the-art results.

Ensemble Approaches Finally, several previous works combined dependency parsers. These include Nivre and McDonald (2008) who used the output of one parser to provide features for another, Zhang and Clark (2008) that proposed a beam-search based parser that combines two parsers into a single system for training and inference, and Martins et al. (2008) that employed stacked learning, in which a second predictor is trained to improve the performance of the first. Our work complements these works by integrating information from a pre-trained TurboParser in our algorithm at test time only.

11 Discussion

We presented a greedy inference approach for graph-based, high-order, non-projective dependency parsing. Our experiments with 17 languages show that our simple and easy to implement algorithm is a decent alternative for dual-decomposition inference.

A major limitation of our algorithm is in-

cluding information from parts that require a given set of arcs *not* to be included in the dependency tree (footnote 1). For example, the $nextSibling((1,2),(1,5))$ part of the TurboParser would fire *iff* the tree includes the arcs $(1,2)$ and $(1,5)$ but not the arcs $(1,3)$ and $(1,4)$.

In order to account for such parts, we should decide how to compute their probabilities and, additionally, at which point they are considered part of the tree. We explored several approaches, but failed to improve our results. Hence, we did not experiment with the third-order TurboParser as all of its third-order parts contain "non-included" arcs. This is left for future work.

A Runtime Complexity Analysis

Here we analyze the complexity of our algorithms, denoting the maximal indegree of a vertex with n_{in}.

Algorithm 1 Algorithm 1 consists of two nested loops (lines 2-k3) and hence lines 4-11 are repeated $O(n \times |E|) = O(n \times n \times n_{in})$ times. At each repetition, loss (lines 6-10) and gain (lines 4-5) values are computed. Afterwards the graph's data structures are updated (lines 13-16).

For every arc that we examine (line 3), there are $O(n_{in})$ *lost* arcs, as there are $O(n_{in})$ incoming arcs (set 1) and $O(n_{in})$ cycles to break (set 2). Since every lost arc translates to a set of *lost parts*, we can avoid repeating computations by storing the partial loss of every arc in a data structure (DS): $e \rightarrow \sum_{part:e \in part} w_{part}$. Now, instead of summing all the lost parts, (every edge participates in $O(n_{in}^{k-1})$ parts,[6] thus there are $O(n_{in}^{k})$ lost parts per added arc), we can sum only $O(n_{in})$ partial loss values. However, since some lost parts may contain an arc from set 1 and an arc from set 2, we need to subtract the values that were summed twice, this can be done in $O(min\{n_{in}^{k-1}, n_{in}^{2}\})$ time by holding a second DS: $e_1 \times e_2 \rightarrow \sum_{part:e_1 \in part \wedge e_2 \in part} w_{part}$.[7]

In order to efficiently compute the gain values, we hold a mapping from arcs to the sum of weights of parts that can be completed in the current iteration by adding the arc to the tree. With this DS, gain values can be computed in constant time. In total, the runtime of lines 4-11 is $O(n_{in} + min\{n_{in}^{k-1}, n_{in}^{2}\})$.

The DSs are initialized in $O(|parts| \times k)$ time. Since every part is deleted at most once, and gets updated (its arcs are added to the tree) at most k times, the total DS update time is $O(k \times |parts|) = O(n_{in}^{k+1})$. Thus algorithm 1 runs in $O(|parts| \times k + n^2 \times n_{in} \times (n_{in} + min\{n_{in}^{k-1}, n_{in}^{2}\}))$ time.

Algorithm 2 Algorithm 2 is similar in structure to Algorithm 1 but the loss and gain computations are more complex. To facilitate efficiency, we hold two DSs: (a) a mapping from arcs to the sum of *lost parts* values, which are now $w_{part} \times P_{part}$ for $part \in parts$; and (b) a mapping from arc pairs to the sum of part values for parts that contain both arcs. The loss and gain values can be computed, as above, in $O(n_{in} + min\{n_{in}^{k-1}, n_{in}^{2}\})$ time.

The initialization of the DSs takes $O(|parts| \times k)$ time. In the i-th iteration we add $e = (u, v)$ to T^i, and remove the $lostArcs$ from E. Every lost arc participates in $O(n_{in}^{k-1})$ parts, and we need to update $O(k)$ entries for each lost part in DS(a) (as the value of the other arcs of that part should no longer account for that part's weight) and $O(k^2)$ entries in DS (b). Thus, the total update time of the DSs is $O(n_{in}^{k} \times k^2)$ and the total runtime of Algorithm 2 is $O(n_{in}^{k+1} \times k + n \times (n \times n_{in} \times (n_{in} + min\{n_{in}^{2}, n_{in}^{k-1}\}) + n_{in}^{k} \times k^2))$. For unpruned graphs and $k \geq 2$ this is equivalent to $O(n^{k+1})$, the theoretical runtime of the TurboParser's dual decomposition inference algorithm.

Acknowledgments

The third author was partly supported by a research grant from the Microsoft/Technion research center for electronic commerce: Context Sensitive Sentence Understanding for Natural Language Processing.

References

Mariana SC Almeida, Cláudia Pinto, Helena Figueira, Pedro Mendes, and André FT Martins. 2015. Aligning opinions: Cross-lingual opinion mining with dependencies. In *ACL*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Pro-*

[6] Assuming that a part is a connected component.

[7] For first order parsing this is not needed; for second order parsing it is done in $O(n_{in})$ time.

ceedings of EMNLP-CoNLL. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *CoNLL*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*.

Jinho D Choi and Andrew McCallum. 2013a. Transition-based dependency parsing with selectional branching. In *ACL*.

Jinho D Choi and Andrew McCallum. 2013b. Transition-based dependency parsing with selectional branching. In *ACL*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *ACL*.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL-HLT*.

Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

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

Matthew Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3:489–501.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *CoNLL*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *EMNLP*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*.

Effi Levi, Roi Reichart, and Ari Rappoport. 2016. Edge-linear first-order dependency parsing with undirected minimum spanning tree inference. In *ACL*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*.

André FT Martins, Dipanjan Das, Noah A Smith, and Eric P Xing. 2008. Stacking dependency parsers. In *EMNLP*.

A. Martins, M. Almeida, and N. A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *ACL*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL-08: HLT*, June.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut – delayed column and row generation for graph based parsers. In *EMNLP-CoNLL*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proc. of the COLING/ACL on Main conference poster sessions*.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*.

Ivan Titov and James Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *EMNLP-CoNLL*.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *EMNLP*.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *ACL*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *EMNLP*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL*.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014a. Greed is good if randomized: New inference for dependency parsing. In *EMNLP*.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014b. Steps to excellence: Simple inference with refined scoring of dependency trees. In *ACL*.

Generating Abbreviations for Chinese Named Entities Using Recurrent Neural Network with Dynamic Dictionary

Qi Zhang, Jin Qian, Ya Guo, Yaqian Zhou, Xuanjing Huang
Shanghai Key Laboratory of Data Science
School of Computer Science, Fudan University
Shanghai, P.R. China
{qz, jqian12, yguo13, zhouyaqian, xjhuang}@fudan.edu.cn

Abstract

Chinese named entities occur frequently in formal and informal environments. Various approaches have been formalized the problem as a sequence labelling task and utilize a character-based methodology, in which character is treated as the basic classification unit. One of the main drawbacks of these methods is that some of the generated abbreviations may not follow the conventional wisdom of Chinese. To address this problem, we propose a novel neural network architecture to perform task. It combines recurrent neural network (RNN) with an architecture determining whether a given sequence of characters can be a word or not. For demonstrating the effectiveness of the proposed method, we evaluate it on Chinese named entity generation and opinion target extraction tasks. Experimental results show that the proposed method can achieve better performance than state-of-the-art methods.

1 Introduction

Abbreviations of Chinese named entities are frequently used on different kinds of environments. Along with the development of social media, this kinds of circumstance occurs more frequently. Unlike western languages such as English, Chinese does not insert spaces between words or word forms that undergo morphological alternations. Hence, most of the Chinese natural language processing methods assume a Chinese word segmenter is used in a pre-processing step to produce word-segmented Chinese sentences as input. However, if the Chinese word segmenter produces erroneous output, the quality of these methods will be degraded as a direct result. Moreover, since the word segmenter may split the targets into two individual words, many methods adopted character-based methodologies, such as methods for named entity recognition (Wu et al., 2005), aspect-based opinion mining (Xu et al., 2014), and so on.

Through character-based methodology, most of the previous abbreviation generation approaches have been formalized as sequence labelling problem. Chinese characters are treated as the basic classification unit and are classified one by one. In these methods, dictionaries play important effect in constructing features and avoiding meaningless outputs. Various previous works have demonstrated the significant positive effectiveness of the external dictionary (Zhang et al., 2010). However, because these external dictionaries are usually static and pre-constructed, one of the main drawbacks of these methods is that the words which are not included in the dictionaries cannot be well processed. This issue has also been mentioned by numerous previous works (Peng et al., 2004; Liu et al., 2012).

Hence, understanding how Chinese words are constructed can benefit a variety of Chinese NLP tasks to avoid meaningless output. For example, to generate the abbreviation for a named entity, we can use a binary classifier to determine whether a character should be removed or retained. Both "国航" and "中国国航" are appropriate abbreviations for "中国国际航空公司(Air China)". However "国航司" is not a Chinese word and cannot be understood by humans.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 721–730,
Austin, Texas, November 1-5, 2016.

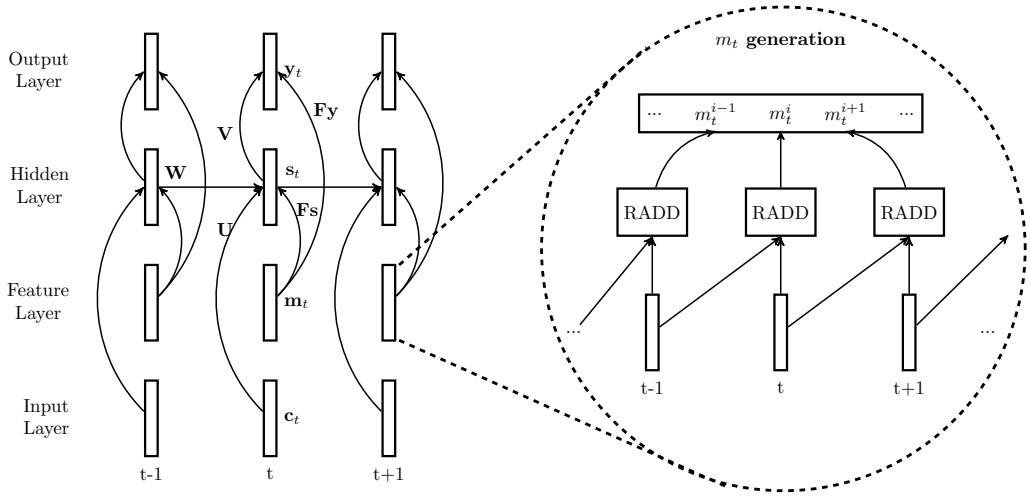


Figure 1: The architecture of RNN with dynamic dictionary.

Thus we are motivated to study the task of "*dynamic dictionary*" and integrating it with sequence labelling model to perform the abbreviation generation task. Dynamic dictionary denotes a binary classification problem which tries to determine whether or not a given sequence of characters is a word. Although human can use implicit knowledge to easily recognize whether an unseen text segment is a word or not at first glance, the task is not as easy as it may seem. First, Chinese has a different morphological system from English. Each Chinese character represents both a syllable and a morpheme (McBride-Chang et al., 2003). Hence, Chinese script is sometimes described as being morphosyllabic. Second, there are many homophones in Chinese. This means that characters that have very different written forms may sound identical. Third, there are a huge number of Chinese words. Without taking the implicit knowledge of morphology into consideration, an arbitrary sequence of characters can be used as a name. In Mandarin, there are approximately 7,000 characters in daily use. Hence, determining whether a given sequence of characters is a word or not is an challenging task.

Since the length of Chinese words is variable, in this paper, we propose a modified recurrent architecture to model the dynamic dictionary construction task. For processing sequence labelling tasks, we also combine the proposed method with RNN. Since the proposed dynamic dictionary model can be pre-trained independently with extensive domain independent dictionaries, the combined model can be easily used in different domains. The proposed model can take advantage of both the sequence-level discrimination ability of RNN and the ability of external dictionary.

The main contributions of this work can be summarized as follows:

- We define the dynamic dictionary problem and construct a large dataset, which consists of more than 20 million words for training and evaluation.
- We integrate RNN with a deep feedforward network based dynamic dictionary learning method for processing Chinese NLP tasks which are formalized as sequence labelling tasks.
- Experimental results demonstrate that the accuracy of the proposed method can achieve better results than current state-of-the-arts methods on two different tasks.

2 Model Architecture

2.1 Dynamic Dictionary

The task of dynamic dictionary is to predict whether a given sequence of characters can be a word or not. The input is a text segment, which contains a variable number of characters. The output is an

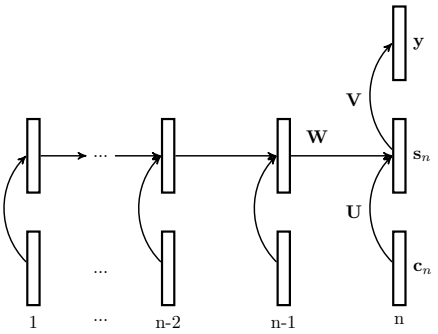


Figure 2: The recurrent architecture used in this work for modelling dynamic dictionary (RADD).

binary value. It is different from the traditional sequence classification tasks, whose the number of outputs are usually same as the input. However, the information of the whole sequence is an important factor and should be incorporated. Hence, in this work, we use a modified recurrent architecture (RADD is used to represent the network in the following literature for short), which is shown in Fig.2.

In Fig.2, n represents the number of characters of the input text segment. c_k represents input character at time k encoded using embeddings of characters through table lookup. The hidden layer s_k maintains the past and current information. The hidden activations of the last step s_n could be considered as the representation of the whole text segment. s_n is used as the input to the classification layer. y produces a probability distribution over the binary labels. Each layer is also represents a set of neurons. Layers are also connected with weights denoted by the matrices $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{V}$. The values in the hidden and output layers are calculated as follows:

$$\begin{aligned} \mathbf{s}_k &= f(\mathbf{U}c_k + \mathbf{W}s_{k-1}) \\ \mathbf{y} &= f(\mathbf{V}s_n) \end{aligned} \tag{1}$$

where $f(\cdot)$ is sigmoid activation function $f(z) = \frac{1}{1+exp^{-z}}$. The architecture can be unfolded as a deep feedforward network.

We define all the parameters for the stage of modelling dynamic dictionary to be trained as $\theta = (W, U, V)$. Given an input text segment, the network with parameter θ outputs the probability, $p(1|x, \theta)$, of the given text segment can be a word or not. Cross entropy criterion is used as the loss function O of the binary classification problem. The network is trained by stochastic gradient descent using *backpropagation through time* (BPTT) (Werbos, 1990). The hidden layer activation of position i at time t, $\mathbf{s}_t^i$, is:

$$\mathbf{s}_t^i = f(a_t^i), \tag{2}$$

$$a_t^i = \sum_j u_{ij} c_t^j + \sum_l w_{il} s_{t-1}^l. \tag{3}$$

The error firstly propagates from output layer to hidden layer of last time step N. The derivatives with respect to the hidden active of position i at the last time step N can be calculated as follows:

$$\delta_N^i = f'(a_N^i) \frac{\partial O}{y} v^i, \tag{4}$$

where v^i represents the weight of hidden-output connection and the activation of the output layer y. The gradients of hidden layer of previous time steps can be recursively computed as:

$$\delta_t^i = f'(a_t^i) \sum_j \delta_{t+1}^j w_{ij}. \tag{5}$$

Given all (suppose the number is T) the training examples (x_i, y_i), we can then define the objective function as follows:

$$J(\theta) = \sum_{i=1}^{T} logp(y^{(i)}|x^{(i)}, \theta). \tag{6}$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ through stochastic gradient decent over shuffled mini-batches with the Adadelta(Zeiler, 2012) update rule.

2.2 RNN-RADD

As mentioned in the previous section, features extracted from external dictionary have been empirically proved to be useful for Chinese NLP various tasks. However, since these external dictionaries are usually pre-constructed, the out-of-vocabulary problem may impact the performance. Hence, we in this work propose to use RNN to determine whether a given sequence of characters is a word or not. Then the proposed RADD is incorporated into RNN (RNN-RADD is used as the abbreviation of the combined model).

2-gram	$c_{t-2}c_{t-1},\ c_{t-2}c_{t-1},\ c_{t-1}c_t$ $c_tc_{t+i},\ c_{t+1}c_{t+2}$
3-gram	$c_{t-2}c_{t-1}c_t,\ c_{t-1}c_tc_{t+1},\ c_tc_{t+1}c_{t+2}$
4-gram	$c_{t-2}c_{t-1}c_tc_{t+1},\ c_{t-1}c_tc_{t+1}c_{t+2}$

Table 1: Illustration of the templates used to generate m_t.

RNN-RADD also follows the character based methodology. Hence, the basic units of RNN-RADD are Chinese characters. The architecture is illustrated in Fig. 1, where c_t denotes the input character at time t encoded using dense distributed representation. The hidden layer s_t also maintains the history of the character sequence. y_t denotes the probability distribution over labels. m_t represents the features generated through RADD. Following previous works, we construct a number of text segments from the contexts of the character based on pre-defined templates. The templates used in this work is shown in the Table. For an input text segment, RADD generates a binary value to indicate whether or not the text segment is a word a not. m_{tj} represents the value of the output corresponding to the jth template for the tth character. Each layer represents a set of neurons. Layers are connected with weights denoted by the matrices $\mathbf{U}$, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{F}_s$, and $\mathbf{F}_y$.

The values in the hidden and output layers in the RNN-RADD can be expressed as follows:

$$\begin{aligned}\mathbf{s}_t &= f(\mathbf{U}c_t + \mathbf{F_s}m_t + \mathbf{W}s_{t-1}), \\ \mathbf{y}_t &= g(\mathbf{V}s_t + \mathbf{F_y}m_t).\end{aligned} \quad (7)$$

Since RAD is trained separately with large scale domain independent dictionaries. In this work, the weight matrices of the RNN-RADD are updated with the similar way as RNN. The error loss function is computed via cross entropy criterion. The parameters are trained by stochastic gradient descent using BPTT. In order to speed up training process, the m_t and character embeddings are keep statistic, during the training procedure.

2.3 Learning Method

Based on the Eq.(3) and Eq.(4), Log-scale objective functions $Q(\theta)$ of RNN-RADD can be calculated as:

$$Q(\theta) = \sum_{t=1}^{T}(\eta a_{y^*_{t-1}y^*_t} + z_t^{y^*_t} - logZ_{R-CRF}).$$

To update the label transition weights, we compute gradients as follows:

$$\begin{aligned}\frac{\partial Q(\theta)}{\partial a_{ji}} &= \eta \sum_t \delta(y_{t-1} = j, y_t = i) \\ &- \eta \sum_t (\frac{\alpha^j_{t-1}\beta^i_t exp(\eta a_{ji} + z^t_i)}{\sum_j \alpha^j_t \beta^j_t}),\end{aligned}$$

where α^i_{t-1} is the sum of partial path scores ending at position $t-1$, with label i, which can be computed as follows:

$$\alpha^i_{t-1} = exp(z^i_{t-1}) \sum_j \alpha^j_{t-2} exp(\eta a_{ji}).$$

β^j_t is the sum of partial path scores starting at position t, with label j and exclusive of observation t, which can be computed as follows:

$$\beta^j_t = \sum_q \beta^q_{t+1} exp(\eta a_{jq} + z^j_{t+1}).$$

The model parameters θ are updated using stochastic gradient ascent (SGA) over the training data multiple passes.

3 Experiments

To demonstrate the effectiveness of the proposed method, we first compared the proposed RNN-based dynamic dictionary construction method against several baseline methods on the task. Then, we evaluated the performance of the proposed method on two Chinese natural language processing tasks: Chinese word segmentation, and opinion target extraction.

3.1 Experimental Settings

To generate the distributed representations for Chinese characters, we use the method similar to Skip-ngram (Mikolov et al., 2013), which has been successfully employed in comparable tasks. However,

in this work, characters were considered the basic units of data, and the toolkit was provided by the authors[1]. We used Sogou news corpus (SogouCA[2]), which consists of news articles belonging to 18 different domains published from June 2012 to July 2012, as the training data to optimize the distributed representations of Chinese characters. After several experiments on development, we decided to set the dimension of the character embedding to 200. Through several evaluations on the validation set, in both RNN-RAD and RAD, the hidden layer size is set to 50.

3.2 Learning Chinese Dynamic Dictionary

For training and testing the proposed dynamic dictionary method, we constructed a dataset by collecting words and names from publicly available resources belonging to different domains, including a Chinese dictionary[3], an English-Chinese bilingual wordlist[4], Baidu Baike[5], the Chinese Domain Dictionary[6], and the Chinese person names list[7]. After removing duplicates, the dataset contains 11,406,995 words in total. Based on the statics of the dictionary we used, about 80.6% of Chinese person names are three characters, and words with two characters comprise the majority of the normal Chinese dictionary. Since some sources contain corporation and organization names, there are also a number of words whose lengths are longer than ten characters. However, in all sources, most of the words are less than five characters.

We randomly selected 50,000 items for use as test data and an additional 50,000 items for use as development data for tuning parameters. In addition to these positive examples, for training and testing, we also needed negative examples, so we extracted bigrams, trigrams, 4-grams, and 5-grams from the SogouCA Then, we randomly extracted a number of n-grams which were not included in the collected word lists described above as negative training data. We treat these n-grams as negative results. For training, testing, and development, we randomly selected 20 million, 50,000, and 50,000 n-grams respectively.

Besides the proposed RADD method, we also evaluated some state-of-the-art supervised methods, including:

Support Vector Machine (SVM) is one of the most common supervised methods and has been successfully used for various tasks (Hearst et al., 1998). Hence, in this work, we also evaluated its performance on the same task. We used the characters as features to construct the vector representation. Since the number of Chinese characters is limited, we used all of the characters existing in the training data. We used LIBSVM to implement (Chang and Lin, 2011).

Conditional Random Fields (CRFs) were proposed by Lafferty et al. (2001) to model sequence labeling tasks. According to the description given in §2.2, an NLP task can be converted into a sequence labeling problem. Hence, we used CRF to model characters as basic features and several combination templates of them. Compared to SVM, CRF takes both richer features and the labeling sequence into consideration. CRF++ 0.58[8] was used to do the experiments.

Dynamic Convolutional Neural Network (DCNN), defined by Kalchbrenner et al. (2014), is used to model sentence semantics. The proposed method can handle input sequences of varying length, so we adopted their method by using the embeddings of characters as input. The toolkit we used in this work is provided by the authors[9].

Recursive Autoencoder (RAE) (Socher et al., 2011), is a machine learning framework for representing variable sized words with a fixed length vector. In this work, we used greedy unsupervised RAE for modeling sequences of Chinese characters. The toolkit was provided by the authors [10]. Then, SVM was used to do the binary classification based on the generated vectors.

Table 3.2 illustrates the results of the different methods on this task. From the results, we see that the proposed method obtains the best perfor-

[1] https://code.google.com/p/word2vec/
[2] http://www.sogou.com/labs/dl/ca.html
[3] http://download.csdn.net/detail/logken/3575376
[4] https://catalog.ldc.upenn.edu/LDC2002L27
[5] http://baike.baidu.com
[6] http://www.datatang.com/data/44250/
[7] http://www.datatang.com/data/13482
[8] http://crfpp.googlecode.com/svn/trunk/doc/index.html
[9] http://nal.co/DCNN
[10] http://www.socher.org/

Methods	P	R	F1
SVM	82.27%	84.74%	83.49%
CRF	80.81%	86.82%	83.71%
DCNN	86.86%	86.55%	86.71%
RAE	84.77%	85.45%	85.11%
RADD	**89.74%**	**91.00%**	**90.39%**

Table 2: Comparison of different methods on the dynamic dictionary construction task.

mance among all of the approaches. DCNN, RAE, and RADD outperform SVM and CRF, which use characters as features. One possible reason is that the character representations are more powerful in capturing morphology than characters only. Another advantage of the deep learning framework is that it can be easily trained and makes feature engineering efforts unnecessary.

We also note that although DCNN can capture word relations of varying size in modelling sentences, RADD achieves better performance on the task of learning the morphology of Chinese. One possible interpretation is that although the relations between words in a given sentence can be well captured by DCNN, relations usually exist between nearby characters hence the recurrent network is more appropriate for the task. Moreover, RADD is much easier to implement and is more efficient than DCNN.

Fig. 3 shows the performance of RADD with different character embedding dimensions and hidden layer sizes. From the figure, we see that RADD achieves the best result when the hidden layer size is larger than 200. We also observe that RNN can achieve the highest performance with many different parameters. This means that we can easily find optimal hyper parameters.

3.3 Experimental Results

3.3.1 Abbreviation Generation

The task of generating entity abbreviations involves producing abbreviated equivalents of the original entities. For example, 北大 is an abbreviation of 北京大学 (Peking University). Previous methods usually formulate the task as a sequence labeling problem and model it using character features (Yang et al., 2009; Xie et al., 2011). Although Chen et al. (2013) proposed to use Markov logic networks (MLN) (Richardson and Domingos, 2006) to combine local and global constraints, the morphology of Chinese was rarely considered.

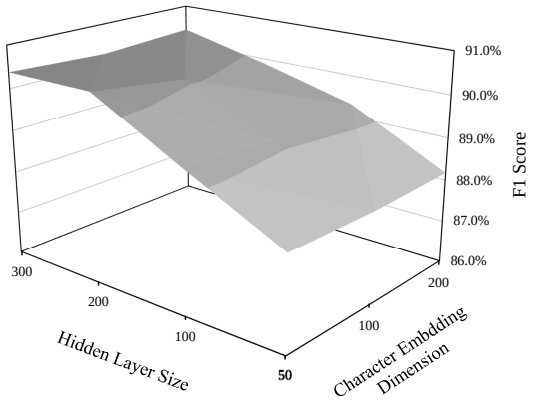


Figure 3: The results of RAD with different character embedding dimension and hidden layer size.

In this work, we report the performance of "RNN-RADD", which takes the dynamic Chinese dictionary into consideration, on the dataset constructed by Chen et al. (2013). The dataset contains 50,232 entity abbreviation pairs. They also reported the performance achieved by their method on the dataset. We follow the strategy used by Chen et al. (2013) to generate training and test data. 75% of randomly selected pairs are used for training data, 5% for development, and the other 20% are used for testing purposes.

For comparison, we also report results achieved by the state-of-the-art methods. Yang et al. (2009) transferred the abbreviation generation method into a sequence labeling problem and proposed to use CRF to model it with several linguistic features. Chen et al. (2013) introduced local and position features and proposed to use MLN to achieve the task. We directly reference and report the results achieved by these methods on the dataset.

Table 3.3.1 shows the relative performances of the different methods. "SVM" and "RNN" denote the results of SVM and RNN on the sequence labeling problem, respectively. From the results, we see that RNN-RADD achieves the best result among all the methods. The relative improvement

Methods	Accuracy
CRFs-Yang (Yang et al., 2009)	39.70%
CRFs-LF+DPF (Chen et al., 2013)	40.60%
MLN (Chen et al., 2013)	56.80%
SVM	40.00%
RNN	60.65%
RNN-RADD	**65.98%**

Table 3: Performance of different methods on abbreviation generation task. CRFs-Yang represents the method and feature sets proposed by Yang et al. (2009). CRF-LF+DPF denotes the local and position features introduced by Chen et al. (2013). MLN represents the method incorporating local and global constraints with MLN.

of it over the previous best result achieved by MLN is about 16.2%. Comparing the performance of RNN-RADD with RNN, we also observe that the dynamic dictionary of Chinese can benefit the abbreviation generation task. The relative improvement is approximately 7.3%.

Fig. 4 shows the values of log-scale objective function of RNN and RNN-RADD during training on the data set. From this figure, we can conclude that the RNN based dynamic dictionary can benefit the task. Although additional feature vector m_i is included, the absolutely value of objective function is lower than its of RNN. It can in some degree demonstrate the effectiveness of the proposed method.

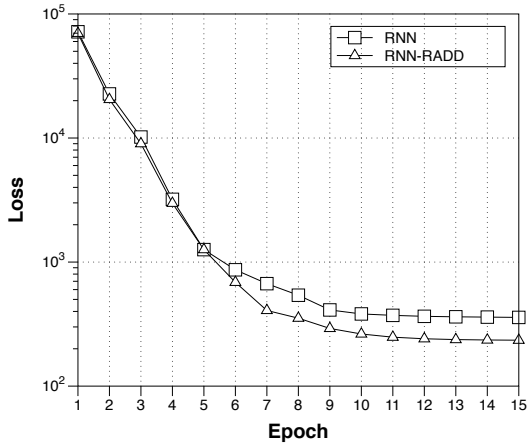


Figure 4: Comparison of RNN and RNN-RADD during training on the abbreviation data set. The vertical axis is the value of log-scale objective functions. Horizontal axis is the number of epochs during training.

	Sentences	Targets
Training	59,786	40,227
Test	11,829	8,034
Development	4,061	2,673

Table 4: Statistics of the dataset used for the opinion target extraction task.

3.3.2 Opinion Target Extraction

Opinion target extraction is a key subtask in the fine-grained opinion mining problem. The goal of it is to identify the items which opinions are expressed on from the given sentences. For example:

The *image quality* of the camera is amazing.

The "*image quality*" is the opinion target of the sentence. Previous methods studied the problem from different perspectives using supervised and unsupervised methods. Syntactic structure constituent is one of the most common assumptions used by previous works (Popescu and Etzioni, 2007; Qiu et al., 2009; Wu et al., 2009; Xu et al., 2014). Since these works usually use character level features, meaningless text segments are one of the major error types. Therefore, we integrate the dynamic Chinese dictionary into this method to detect and discard meaningless text segments.

To evaluate the proposed method, we used a dataset containing more than 6,000 reviews, which contains 75,676 sentences, about vehicles. The opinion target and opinion words were manually labeled. About 80% of the whole dataset is randomly selected for training. 15% and 5% reviews are selected as the test and development datasets respectively. Details of the data are listed in Table 3.3.2.

The task can also be modelled by sequence labelling problem. Hence, besides the proposed RNN-RADD method, we also evaluated some state-of-the-art supervised methods, including: CRF, SVM, and RNN. We used SVM and CRF under the character-based methodology for comparison. RNN is based on the character level embeddings. Table 3.3.2 shows the results of the different methods on the opinion target extraction task. From the results, we can see that, the proposed method RNN-RADD achieve the best performance in F1 score. Comparing the results of RNN with RNN-RADD, we see that the proposed dynamic dictionary

Methods	P	R	F1
CRF	71.1%	77.5%	74.2%
CRF+D	72.5%	74.3%	73.4%
SVM	77.2%	74.9%	76.0%
SVM+D	78.1%	74.3%	76.2%
RNN	79.5%	**81.7%**	80.6%
RNN-RADD	**85.5%**	81.5%	**83.4%**

Table 5: Results of different methods on the opinion target extraction task.

method can benefit the RNN based method. The error reduction achieved by its incorporation is about 11.4%. From the results of CRF and CRF+D, we can observe that dictionary is not always usefulness. We think that the main reason that the dictionary may bring too much conflict. From the results of CRF and RNN, we can see that similar to the Chinese word segmentation task, methods using character dense representations can usually achieve better performance than character based methods.

4 Related Work

Although dictionary can be manually constructed, it is a time-consuming work. Moreover, these manually constructed dictionaries are usually updated only occasionally. It would take months before it could be updated. Hence, automatic dictionary construction methods have also been investigated in recent years. Chang and Su (1997) proposed an unsupervised iterative approach for extracting out-of-vocabulary words from Chinese text corpora. Khoo (Khoo et al., 2002) introduced a method based on stepwise logistic regression to identify two-and three-character words in Chinese text. Jin and Wong (2002) incorporated local statistical information, global statistical information and contextual constraints to identify Chinese words. For collecting Thai unknown words, Haruechaiyasak et al. (2006) proposes a collaborative framework for achieving the task based on Web pages over the Internet.

Except these unsupervised methods, there have been other approaches requiring additional information or selective input. Yarowsky and Wicentowski (2000) proposed to use labeled corpus to train a supervised method for transforming past-tense in English. Rogati et al. (2003) introduced a stemming model based on statistical machine translation for Arabic. They used a parallel corpus to train the model. Luong et al. (2013) studied the problem of word representations for rare and complex words. They proposed to combine recursive neural networks and neural language models to build representations for morphologically complex words from their morphemes. Since English is usually considered limited in terms of morphology, their method can handle unseen words, whose representations could be constructed from vectors of known morphemes.

However, most of the existing Chinese dictionary construction methods focused on find out-of-vocabulary words from corpus. In this paper, we propose to transfer the dictionary construction problem to classification task and use a modified recurrent neutral network to directly model whether a given sequences of characters is a word or not.

5 Conclusion

In this work, we studied the problem of dynamic dictionary which tries to determine whether a sequence of Chinese characters is a word or not. We proposed a deep feed forward network architecture (RADD) to model the problem and integrated it into RNN method. To train the model and evaluate the effectiveness of the proposed method, we constructed a dataset containing more than 11 million words. By applying the proposed combined method to two different Chinese NLP tasks, we can see that it can achieve better performance than state-of-the-art methods. Comparing to the previous methods, the number of hyper parameters of the proposed method RNN-RADD is small and less feature engineering works are needed. In the future, we plan to integrate the dynamic dictionary into the term construction model in information retrieval.

6 Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Jing-Shin Chang and Keh-Yih Su. 1997. An unsupervised iterative method for chinese new lexicon extraction. *Computational Linguistics and Chinese Language Processing*, 2(2):97–148.

Huan Chen, Qi Zhang, Jin Qian, and Xuanjing Huang. 2013. Chinese named entity abbreviation generation using first-order logic. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Choochart Haruechaiyasak, Chatchawal Sangkeettrakarn, Pornpimon Palingoon, Sarawoot Kongyoung, and Chaianun Damrongrat. 2006. A collaborative framework for collecting thai unknown words from the web. In *Proceedings of the COLING/ACL*.

Marti A. Hearst, ST Dumais, E Osman, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28.

Honglan Jin and Kam-Fai Wong. 2002. A chinese dictionary construction algorithm for information retrieval. *ACM Transactions on Asian Language Information Processing (TALIP)*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Christopher SG Khoo, Yubin Dai, and Teck Ee Loh. 2002. Using statistical and contextual information to identify two-and three-character words in chinese text. *Journal of the American Society for Information Science and Technology*, 53(5):365–377.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Xiaohua Liu, Ming Zhou, Furu Wei, Zhongyang Fu, and Xiangyang Zhou. 2012. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of ACL*.

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

Catherine McBride-Chang, Hua Shu, Aibao Zhou, Chun Pong Wat, and Richard K Wagner. 2003. Morphological awareness uniquely predicts young children's chinese character recognition. *Journal of Educational Psychology*, 95(4).

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

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*, volume 9, pages 1199–1204.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Monica Rogati, Scott McCarley, and Yiming Yang. 2003. Unsupervised learning of arabic stemming using a parallel corpus. In *Proceedings of ACL 2003*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the EMNLP '11*.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*.

Youzheng Wu, Jun Zhao, Bo Xu, and Hao Yu. 2005. Chinese named entity recognition based on multiple features. In *Proceedings of HLT/EMNLP*.

Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*.

Li-Xing Xie, Ya-Bin Zheng, Zhi-Yuan Liu, Mao-Song Sun, and Can-Hui Wang. 2011. Extracting chinese abbreviation-definition pairs from anchor texts. In *Machine Learning and Cybernetics (ICMLC)*.

Liheng Xu, Kang Liu, Siwei Lai, and Jun Zhao. 2014. Product feature mining: Semantic clues versus syntactic constituents. In *Proceedings of the 52nd ACL*.

Dong Yang, Yi-cheng Pan, and Sadaoki Furui. 2009. Automatic chinese abbreviation generation using conditional random field. In *Proceedings of NAACL 2009*.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd international conference on computational linguistics: Posters*.

Neural Network for Heterogeneous Annotations

Hongshen Chen[*†§] **Yue Zhang**[‡] **Qun Liu**[♢†]
†Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
§University of Chinese Academy of Sciences
‡Singapore University of Technology and Design
♢ ADAPT centre, School of Computing, Dublin City University
chenhongshen@ict.ac.cn, yue_zhang@sutd.edu.sg

Abstract

Multiple treebanks annotated under heterogeneous standards give rise to the research question of best utilizing multiple resources for improving statistical models. Prior research has focused on discrete models, leveraging stacking and multi-view learning to address the problem. In this paper, we empirically investigate heterogeneous annotations using neural network models, building a neural network counterpart to discrete stacking and multi-view learning, respectively, finding that neural models have their unique advantages thanks to the freedom from manual feature engineering. Neural model achieves not only better accuracy improvements, but also an order of magnitude faster speed compared to its discrete baseline, adding little time cost compared to a neural model trained on a single treebank.

1 Introduction

For many languages, multiple treebanks have been annotated according to different guidelines. For example, several linguistic theories have been used for defining English dependency treebanks, including Yamada and Matsumoto (2003), LTH (Johansson and Nugues, 2007) and Stanford dependencies (De Marneffe et al., 2006). For German, there exist TIGER (Brants et al., 2002) and TüBa-D/Z (Telljohann et al., 2006). For Chinese, treebanks have been made available under various segmentation granularities (Sproat and Emerson, 2003; Emerson, 2005; Xue, 2003). These give rise to the research problem of effectively making use of multiple treebanks under heterogeneous annotations for improving output accuracies (Jiang et al., 2015; Johansson, 2013; Li et al., 2015).

The task has been tackled using two typical approaches. The first is based on *stacking* (Wolpert, 1992; Breiman, 1996; Wu et al., 2003). As shown in Figure 1(a), the main idea is to have a model trained using a source treebank, which is then used to guide a target treebank model by offering source-style features. This method has been used for leveraging two different treebanks for word segmentation (Jiang et al., 2009; Sun and Wan, 2012) and dependency parsing (Nivre and McDonald, 2008; Johansson, 2013).

The second approach is based on *multi-view learning* (Johansson, 2013; Li et al., 2015). The idea is to address both annotation styles simultaneously by sharing common feature representations. In particular, Johansson (2013) trained dependency parsers using the domain adaptation method of Daumé III (2007), keeping a copy of shared features and a separate copy of features for each treebank. Li et al. (2015) trained POS taggers by coupling the labelsets from two different treebanks into a single combined labelset. A summary of such multi-view methods is shown in Figure 1(b), which demonstrates their main differences compared to stacking (Figure 1(a)).

Recently, neural network has gained increasing research attention, with highly competitive results being reported for numerous NLP tasks, including word segmentation (Zheng et al., 2013; Pei et al., 2014; Chen et al., 2015), POS-tagging (Ma et al., 2014; Plank et al., 2016), and parsing (Chen and

[*]Work done when the first author was visiting SUTD.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 731–741,
Austin, Texas, November 1-5, 2016.

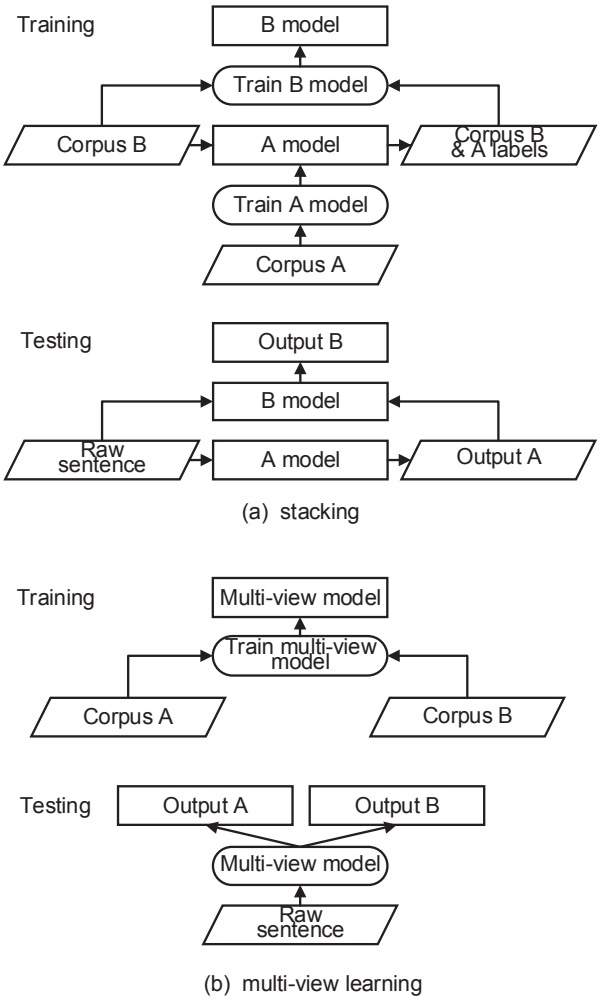


Figure 1: Two main approaches to utilizing heterogeneous annotations.

Manning, 2014; Dyer et al., 2015; Weiss et al., 2015; Zhou et al., 2015). On the other hand, the aforementioned methods on heterogeneous annotations are investigated mainly for discrete models. It remains an interesting research question how effective multiple treebanks can be utilized by neural NLP models, and we aim to investigate this empirically.

We follow Li et al. (2015), taking POS-tagging for case study, using the methods of Jiang et al. (2009) and Li et al. (2015) as the discrete stacking and multi-view training baselines, respectively, and building neural network counterparts to their models for empirical comparison. The base tagger is a neural CRF model (Huang et al., 2015; Lample et al., 2016), which gives competitive accuracies to discrete CRF taggers.

Results show that *neural stacking* allows deeper integration of the source model beyond one-best outputs, and further the fine-tuning of the source model during the target model training. In addition, the advantage of *neural multi-view learning* over its discrete counterpart are many-fold. First, it is free from the necessity of manual cross-labelset interactive feature engineering, which is far from trivial for representing annotation correspondence (Li et al., 2015). Second, compared to discrete model, parameter sharing in deep neural network eliminates the issue of exponential growth of search space, and allows separated training of each label type, in the same way as multi-task learning (Collobert et al., 2011).

Our neural multi-view learning model achieves not only better accuracy improvements, but also an order of magnitude faster speed compared to its discrete baseline, adding little time cost compared to a neural model trained on a single treebank. The C++ implementations of our neural network stacking and multi-view learning models are available under GPL, at `https://github.com/chenhongshen/NNHetSeq`.

2 Baseline Neural Network Tagger

We adopt a neural CRF with a Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber, 1997) feature layer for baseline POS tagger. As shown in Figure 2, the model consists of three main neural layers: the **input layer** calculates dense representation of input words using attention model on character embeddings; the **feature layer** employs a bi-directional LSTM model to extract non-local features from input vectors; the **output layer** uses a CRF structure to infer the most likely label for each input word.

2.1 Input Layer

Given a sentence $w^{(1:n)}$, the input layer builds a vector representation $\vec{r}_w^i$ for each word w^i based on both word and character embeddings. In particular, an embedding lookup table is used to convert vocabulary words into their embedding forms directly. To obtain a character based embedding of w^i, we denote the character sequence of w^i with $c^{(1:n)}$, where c^j is the jth character in w^i.

A character lookup table is used to map each char-

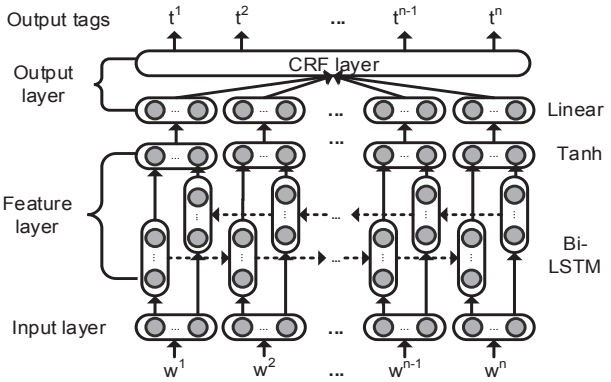


Figure 2: Baseline neural network tagger.

acter c^j into a character embedding $\vec{e}_c^j$. The character embeddings $\vec{e}_c^1, \vec{e}_c^2, ..., \vec{e}_c^m$ are combined using an attention model(Bahdanau et al., 2015): $\vec{w}_c^i = \sum_{j=1}^m a_c^j \odot \vec{e}_c^j$, where a_c^j is the weight for $\vec{e}_c^j$, $\odot$ is the Hadamard product function, and $\sum_{j=1}^m a_c^j = 1$.

Each a_c^j is computed according to both the word embedding vector and 5-character embedding window with the current character $\vec{e}_c^j$ in the middle:

$$
\begin{aligned}
a_c^j =& \frac{t_c^j}{\sum_1^m t_c^j} \\
t_c^j =& \exp(\mathbf{W}^t \vec{h}_c^j + \mathbf{U}^t \vec{e}_w^i + \vec{b}^t) \\
\vec{h}_c^i =& \tanh\big(\mathbf{W}^c(\vec{e}_c^{j-2} \oplus \vec{e}_c^{j-1} \oplus \vec{e}_c^j \oplus \vec{e}_c^{j+1} \\
& \oplus \vec{e}_c^{j+2}) + \vec{b}^c\big)
\end{aligned}
$$

Here $\oplus$ denotes vector concatenation and $\vec{e}_w^i$ is the embedding of current word w^i. $\mathbf{W}^t$, $\mathbf{U}^t$, $\mathbf{W}^c$ and $\vec{b}^t$, $\vec{b}^c$ are model parameters. Finally, $\vec{w}_c^i$ is concatenated with word embedding to form final word representation $\vec{r}_w^i$: $\vec{r}_w^i = \vec{e}_w^i \oplus \vec{w}_c^i$

2.2 Feature Layer

Recently, bi-directional LSTM has been successfully applied in various NLP tasks (Liu et al., 2015; Zhou and Xu, 2015; Klerke et al., 2016; Plank et al., 2016). The feature layer uses a bi-directional LSTM to extract a feature vector $\vec{h}^i$ for each word w^i, respectively. An input vector $\vec{x}^i = (\vec{r}_w^{i-2} \oplus \vec{r}_w^{i-1} \oplus \vec{r}_w^i \oplus \vec{r}_w^{i+1} \oplus \vec{r}_w^{i+2})$ is used to represent each word w^i.

We use a LSTM variation with peephole connections (Graves and Schmidhuber, 2005) to extract features based on $\vec{x}^{(1:n)}$. The model computes a hidden vector $\vec{h}^i$ for each input $\vec{x}^i$, passing information from $\vec{h}^1, ..., \vec{h}^{i-1}$ to $\vec{h}^n$ via a sequence of cell states $\vec{c}^1, \vec{c}^2, ..., \vec{c}^n$. Information flow is controlled using an input gate $\vec{g}^i$, a forget gate $\vec{f}^i$, and an output gate $\vec{o}^i$:

$$
\begin{aligned}
\vec{g}^i =& \sigma(\mathbf{W}^{(g)}\vec{x}^i + \mathbf{U}^{(g)}\vec{h}^{i-1} + \mathbf{V}^{(g)}\vec{c}^{i-1} + \vec{b}^{(g)}) \\
\vec{f}^i =& \sigma(\mathbf{W}^{(f)}\vec{x}^i + \mathbf{U}^{(f)}\vec{h}^{i-1} + \mathbf{V}^{(f)}\vec{c}^{i-1} + \vec{b}^{(f)}) \\
\vec{c}^i =& \vec{f}^i \odot \vec{c}^{i-1} + \\
& \vec{g}^i \odot \tanh(\mathbf{W}^{(u)}\vec{x}^i + \mathbf{U}^{(u)}\vec{h}^{i-1} + \vec{b}^{(u)}) \\
\vec{o}^i =& \sigma(\mathbf{W}^{(o)}\vec{x}^i + \mathbf{U}^{(o)}\vec{h}^{i-1} + \mathbf{V}^{(o)}\vec{c}^i + \vec{b}^{(o)}) \\
\vec{h}^i =& \vec{o}^i \odot \tanh(\vec{c}^i),
\end{aligned}
$$

where σ denotes the component-wise sigmoid function. $\mathbf{W}^{(g)}$, $\mathbf{W}^{(f)}$, $\mathbf{W}^{(u)}$, $\mathbf{W}^{(o)}$, $\mathbf{U}^{(g)}$, $\mathbf{U}^{(f)}$, $\mathbf{U}^{(u)}$, $\mathbf{U}^{(o)}$, $\mathbf{V}^{(g)}$, $\mathbf{V}^{(f)}$, $\mathbf{V}^{(o)}$, $\vec{b}^{(g)}$, $\vec{b}^{(f)}$, $\vec{b}^{(u)}$, $\vec{b}^{(o)}$ are model parameters.

Bi-directional extension of the above LSTM structure is applied in both the left-to-right direction and the right-to-left direction, resulting in two hidden vector sequences $h_l^{(1:n)}, h_r^{(1:n)}$, respectively. Each $\vec{h}_l^i$ is combined with its corresponding $\vec{h}_r^i$ for final feature vector $\vec{h}_f^i$:

$$
\vec{h}_f^i = \tanh(\mathbf{W}^l \vec{h}_l^i + \mathbf{W}^r \vec{h}_r^i + \vec{b}),
$$

where $\mathbf{W}^l$, $\mathbf{W}^r$ and $\vec{b}$ are model parameters.

2.3 Output Layer

The output layer employs a conditional random field (CRF) to infer the POS t^i of each word w^i based on the feature layer outputs. The conditional probability of a tag sequence given an input sentence is:

$$
p(\vec{y}|\vec{x}) = \frac{\prod_{i=1}^n \Psi(\vec{x}, \vec{y}^i) \prod_{i=1}^n \Phi(\vec{x}, \vec{y}^i, \vec{y}^{i-1})}{Z(\vec{x})},
$$

where $Z(\vec{x})$ is the partition function:

$$
Z(\vec{x}) = \sum_{\vec{y}} \prod_{i=1}^n \Psi(\vec{x}, \vec{y}^i) \prod_{i=1}^n \Phi(\vec{x}, \vec{y}^i, \vec{y}^{i-1})
$$

In particular, the output clique potential $\Psi(\vec{x}, \vec{y}^i)$ shows the correlation between inputs and output labels: $\Psi(\vec{x}, \vec{y}^i) = \exp(\vec{s}^i)$, with the emission vector $\vec{s}^i$ being defined as:

$$
\vec{s}^i = \vec{\theta}_0 \vec{h}_f^i, \tag{1}
$$

where $\vec{\theta}_0$ is the model parameter.

The edge clique potential shows the correlation between consecutive output labels using a single transition weight $\tau(\vec{y}^i, \vec{y}^{i-1})$.

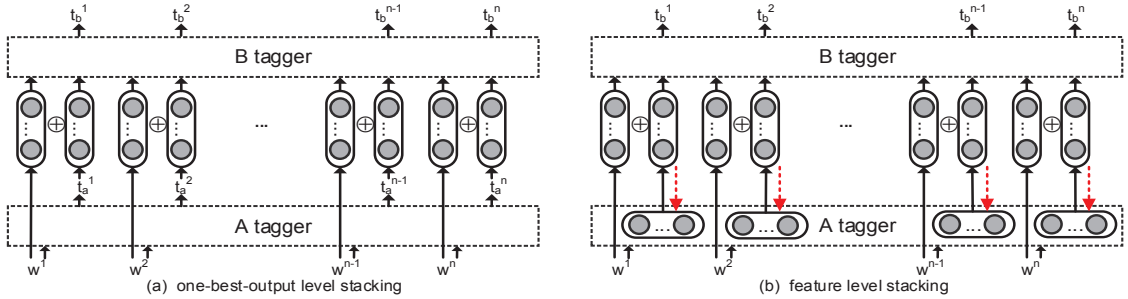


Figure 3: Neural stacking.

3 Stacking

3.1 Discrete Stacking

Stacking integrates corpora A and B by first training a tagger on corpus A, and then using the A tagger to provide additional features to a corpus B model. Figure 1(a) shows the training and testing of discrete stacking models, where the B tagger extracts features from both the raw sentence and A tagger output. This method achieves feature combination at the one-best-output level.

3.2 Neural Stacking

Figure 3(a) and (b) shows the two neural stacking methods of this paper, respectively.

Shallow Integration. Figure 3(a) is a variation of discrete stacking, with the output tags from tagger A being converted to a low-dimensional dense embedding features, and concatenated to the word embedding inputs to tagger B. Formally, for each word w^i, denote the tagger A output as t_a^i, we concatenate the embedding form of t_a^i, denoted as $\vec{e}_a^i$, to the word representation $\vec{r}_w^i$.

$$\vec{r}_w^{i\prime} = \vec{r}_w^i \oplus \vec{e}_a^i = \vec{e}_w^i \oplus \vec{w}_c^i \oplus \vec{e}_a^i \quad (2)$$

Deep Integration. Figure 3(b) offers deeper integration between the A and B models, which is feasible only with neural network features. We call this method feature-level stacking. For feature-level integration, the emission vector $\vec{s}^i$ in Eq.(1) is taken for input to tagger B via a projection:

$$\vec{w}_a^i = \tanh(\mathbf{W}^s \vec{s}^i)$$
$$\vec{r}_w^i = \vec{e}_w^i \oplus \vec{w}_c^i \oplus \vec{w}_a^i,$$

where $\mathbf{W}^s$ is a model parameter.

Fine-tuning. Feature-level stacking further allows tagger A to be fine-tuned during the training of tagger B, with the loss function being back propagated to tagger A via the $\vec{w}_a^i$ layer (shown in the red dotted lines in Figure 3(b)). This is a further benefit of neural stacking compared with discrete stacking.

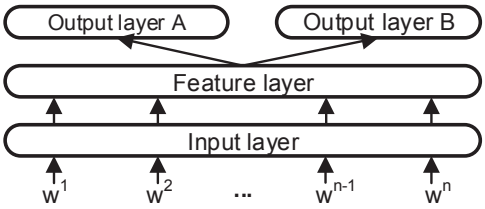


Figure 4: Neural multi-view model.

4 Multi-view Learning

4.1 Discrete Label Coupling

As shown in Figure 1(b), multi-view learning (Li et al., 2015) utilizes corpus A and corpus B simultaneously for training. The coupled tagger directly learns the logistic correspondences between both corpora, therefore can lead a more comprehensive usage of corpus A compared with stacking. In order to better capture such correlation, specifically designed feature templates between two tag sets are essential.

For each training instances, both A and B labels are needed. However, one type of tag is missing. Li et al. (2015) used a mapping function to supplement the missing annotations with the help of the annotated tag. The result is a set of sentence with bundled tags in both annotations, but with ambiguities on one side, due to one-to-many mappings. Li et al. (2015) showed that speed can be significantly improved by manually restricting possible mappings between the labelsets, but a full mapping without restriction yields the highest accuracies.

4.2 Neural Multi-task Learning

Neural multi-task learning is free from manual feature engineering, and avoids manual mapping func-

tions between tag sets by establishing two separate output layers, one for each type of label, with shared low-level parameters. The general structure of a neural multi-view model is shown in Figure 4, which can be regarded as a variation of the parameter sharing model of Caruana (1993) and Collobert et al. (2011). Leveraging heterogeneous annotations for the same task, compared to parameter sharing between different NLP tasks (Collobert et al., 2011), can benefit from tighter integration of information, and hence allows deeper parameter sharing. These are verified empirically in the experiments.

In training and testing, sentences from both corpora go through the same input layer and feature layer. The outputs of each type of tag is then computed separately according to the shared parameters. The conditional probability of a tag sequence given an input sentence and its corpus type is:

$$p(\vec{y}|\vec{x},T) = \frac{\prod_{i=1}^{n} \Psi_T(\vec{x},\vec{y}^{\,i}) \prod_{i=1}^{n} \Phi_T(\vec{x},\vec{y}^{\,i},\vec{y}^{\,i-1})}{Z_T(\vec{x})},$$

where T is the corpus type, $T \in \{A, B\}$. $\Psi_T(\vec{x},\vec{y}^{\,i})$ and $\Phi_T(\vec{x},\vec{y}^{\,i},\vec{y}^{\,i-1})$ are the corresponding output clique potential and edge clique potential, respectively. $Z_T(\vec{x})$ is the partition function:

$$Z_T(\vec{x}) = \sum_{\vec{y}} \prod_{i=1}^{n} \Psi_T(\vec{x},\vec{y}^{\,i}) \prod_{i=1}^{n} \Phi_T(\vec{x},\vec{y}^{\,i},\vec{y}^{\,i-1})$$

This indicates that each time only one output layer is activated according to the corpus type of input sentences.

5 Training

A max-margin objective is used to train the full set of model parameters Θ:

$$L(\Theta) = \frac{1}{D} \sum_{d=1}^{D} l(\vec{x_d},\vec{y_d},\Theta) + \frac{\lambda}{2} \|\Theta\|^2,$$

where $\vec{x}_d, \vec{y}_d|_{d=1}^{D}$ are the training examples, λ is a regularization parameter, and $l(\vec{x}_d,\vec{y}_d,\Theta)$ is the max-margin loss function towards one example $(\vec{x}_d, \vec{y_d})$.

The max-margin loss function is defined as:

$$l(\vec{x}_d,\vec{y_d},\Theta) = \max_{y} \left(s(\vec{y}|\vec{x}_d,\Theta) + \delta(\vec{y},\vec{y}_d)\right) - s(\vec{y}_d|\vec{x}_d,\Theta),$$

Algorithm 1 Neural multi-view training

Input: Two training datasets: $D^{(1)} = (x_n^{(1)}, y_n^{(1)})|_{n=1}^{N}$, $D^{(2)} = (x_m^{(2)}, y_m^{(2)})|_{m=2}^{M}$; Parameters: E, A, R

Output: Θ

```
for i = 1 to E do
    Sample A instances from D^(1) and A * R instances from D^(2) to form a new dataset D_i
    Shuffle D_i.
    for each batch D_i^b in D_i do
        Forward: compute the cost
        Backward: compute the loss of each parameter
        Update the parameters
    end for
end for
```

		sentences	tokens
CTB	train	16091	437991
	dev	803	20454
	test	1910	50319
PD	train	100749	5194829
	dev	18875	958026

Table 1: Data statistics.

where $\vec{y}$ is the model output, $s(\vec{y}|\vec{x}) = logP(\vec{y}|\vec{x})$ is the log probability of $\vec{y}$ and $\delta(\vec{y},\vec{y}_d)$ is the Hamming distance between $\vec{y}$ and $\vec{y}_d$.

We adopt online learning, updating parameters using AdaGrad (Duchi et al., 2011). To train the neural stacking model, we first train a base tagger using corpus A. Then, we train the stacked tagger with corpus B, where the parameters of the A tagger has been pretrained from corpus A and the B tagger is randomly initialized.

For neural multi-view model, we follow Li et al. (2015) and take a the corpus-weighting strategy to sample a number of training instances from both corpora for each training iteration, as shown in Algorithm 1. At each epoch, we randomly sample from the two datasets according to a *corpus weights ratio*, namely the ratio between the number of sentences in each dataset used for training, to form a training set for the epoch.

6 Experiments

6.1 Experimental Settings

We adopt the Penn Chinese Treebank version 5.0 (*CTB5*) (Xue et al., 2005) as our main corpus, with the standard data split following previous work (Zhang and Clark, 2008; Li et al., 2015). People's Daily (*PD*) is used as second corpus with a different scheme. We filter out *PD* sentences longer than 200 words. Details of the datasets are listed in Table 1. The standard token-wise POS tagging accuracy is used as the evaluation metric. The systems are implemented with LibN3L (Zhang et al., 2016).

For all the neural models, we set the hidden layer size to 100, the initial learning rate for Adagrad to 0.01 and the regularization parameter λ to 10^{-8}. *word2vec*[1] is used to pretrain word embeddings. The Chinese Giga-word corpus version 5 (Graff and Chen, 2005), segmented by *zpar*[2] (Zhang and Clark, 2011), is used for the training corpus for word embeddings. The size of word embedding is 50.

6.2 Development Experiments

We use the development dataset for two main purposes. First, under each setting, we tune the model parameters, such as the number of training epochs. Second, we study the influence of several important hyper-parameters using the development dataset. For example, for the NN multi-view learning model, the *corpus weights ratio* (section 5) plays an important role for the performance. We determine the parameters of the model by studying the accuracy along with the increasing epochs.

Effect of batch size and dropout. The batch size affects the speed of training convergence and the final accuracies of the neural models, and the dropout rate has been shown to significantly influence the performance (Chen et al., 2015). We investigate the effects of these two hyper-parameters by adopting a corpus weight ratio of $1:1$ (All the *CTB* training data is used, while the same amount of *PD* is sampled randomly), drawing the accuracies of the neural multi-view learning model against the number of training epochs with various combinations of the dropout rate d and batch size b. The results are shown for the multi-view learning model. For the stacking model, we use b=100 for the *PD* sub model.

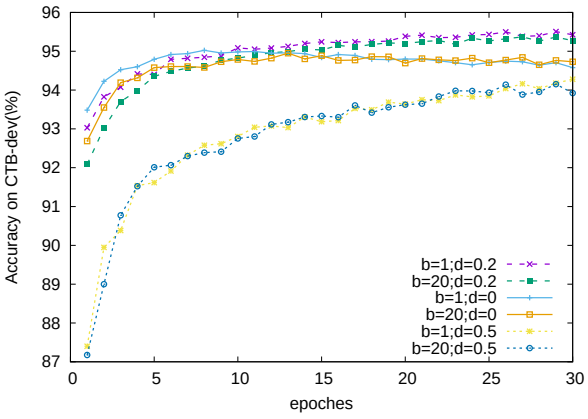


Figure 5: Accuracy on *CTB-dev* with different batch sizes and dropout rates for a multi-view learning model. b represents batch size, d denotes dropout rate.

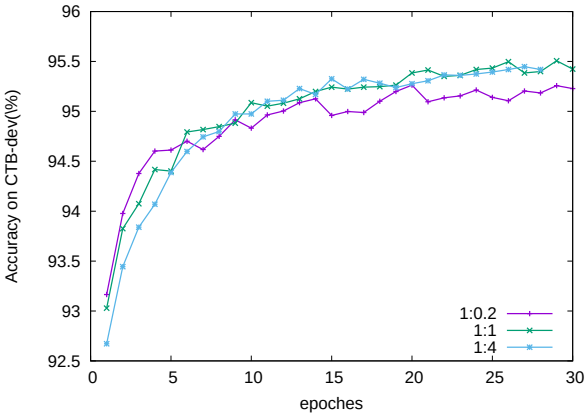


Figure 6: Accuracy on *CTB-dev* with different corpus weights ratio.

The results are shown in Figure 5, where the two dashed lines on the top at epoch 30 represent the dropout rate of 20%, the two solid lines in the middle represent zero dropout rate, and the two dotted lines in the bottom represent a dropout rate 50%. Without using dropout, the performance increases in the beginning, but then decreases as the number of training epochs increases beyond 10. This indicates that the NN models can overfit the training data without dropout. However, when a 50% dropout rate is used, the initial performances are significantly worse, which implies that the 50% dropout rate can be too large and leads to underfitting. As a result, we choose a dropout rate of 20% for the remaining experiments, which strikes the balance between over-

[1]https://code.google.com/p/word2vec
[2]https://github.com/SUTDNLP/ZPar

System	Accuracy
CRF Baseline (Li et al., 2015)	94.10
CRF Stacking (Li et al., 2015)	94.81
CRF Multi-view (Li et al., 2015)	95.00
NN Baseline	94.24
NN Stacking	94.74
NN Feature Stacking	95.01
NN Feature Stacking & Fine-tuning	95.32
NN Multi-view	95.40
Integrated NN Multi-view & Stacking	**95.53**

Table 2: Accuracies on *CTB-test*.

fitting and underfitting.

Figure 5 also shows that the batch size has a relative small influence on the accuracies, which varies according to the dropout rate. We simply choose a batch size of 1 for the remaining experiments according to the performance at the dropout rate 20%.

Effect of corpus weights ratio. Figure 6 shows the effects of different corpus weights ratios. In particular, a corpus weights ratio of 1:0.2 yields relative low accuracies. This is likely because it makes use of the least amount of *PD* data. The ratios of 1:1 and 1:4 give comparable performances. We choose the former for our final tests because it is a much faster choice.

6.3 Final Results

Table 2 shows the final results on the *CTB* test data. We lists the results of stacking method of Jiang et al. (2009) re-implemented by Li et al. (2015), and CRF multi-view method reported by Li et al. (2015). We adopt pair-wise significance test (Collins et al., 2005) when comparing the results between two different models.

Stacking. For baseline tagging using only *CTB*, NN model achieves a result of 94.24, slightly higher than CRF baseline (94.10). NN stacking model integrating *PD* data achieves comparable performance (94.74) compared with CRF stacking model (94.81). Compared with NN baseline, NN stacking model boosts the performance from 94.24 to 94.74, which is significant at the confidence level $p < 10^{-5}$. This demonstrates that neural network model can utilize one-best prediction of the *PD* model for the *CTB* task as effectively as the discrete stacking method of Jiang et al. (2009).

One advantage of NN stacking as compared with discrete stacking method is that it can directly leverage features of *PD* model for *CTB* tagging. Comparison between feature-level stacking and one-best-output level stacking of the NN stacking model shows that the former gives significantly higher results, namely 95.01 vs 94.74 at the confidence level $p < 10^{-3}$.

One further advantage of NN stacking is that it allows the *PD* model to be fine-tuned as an integral sub-model during *CTB* training. This is not possible for the discrete stacking model, because the output of the *PD* model are used as atomic feature in the stacking *CTB* model rather than a gradient admissive neural layer. By fine-tuning the *PD* sub-model, the performance is further improved from 95.01 to 95.32 at the confidence level $p < 10^{-3}$. The final NN stacking model improves over the NN baseline model from 94.24 to 95.32. The improvement is significantly higher compared to that by using discrete stacking which improves over the discrete baseline from 94.01 to 94.74. The final accuracy of the NN stacking model is higher than that of the CRF stacking model, namely 94.81 vs 95.32 at the confidence level $p < 10^{-3}$. This shows that neural stacking is a preferred choice for stacking.

Multi-view training. With respect of the multi-view training method, the NN model improves over the NN baseline from 94.24 to 95.40, by a margin of +1.16, which is higher than that of 0.90 brought by discrete method of Li et al. (2015) over its baseline, from 94.10 to 95.00. NN multi-view training method gives relatively higher improvements compared with NN stacking method. This is consistent with the observation of Li et al. (2015), who showed that discrete label coupling training gives slightly better improvement compared with discrete stacking. The final accuracies of NN multi-view training is also higher than that of its CRF counterpart, namely 95.00 vs 95.40 at the confidence level $p < 10^{-3}$. The difference between the final NN multi-view training result of 95.40 and the final NN stacking results is not significant.[3]

Integration. The flexibility of the NN models further allows both stacking (on the input) and multi-viewing (on the output) to be integrated. When

[3]Note, however, NN stacking method with one-best *PD* output gives significantly lower accuracies (94.74). It is the fine-tuning strategy that allows stacking to give comparable results compared to multi-view training for the NN models.

System	Time Cost(s)
CRF Baseline	176.925
CRF Multi-view (Li et al., 2015)	3992.27
NN Baseline	416.338
NN Multi-view	418.484

Table 3: Time for testing *CTB* training data.

NN multi-view training is combined with a fine-tuned NN feature stacking model, the performance further increases from 95.40 to 95.53. This is the best results we are aware of on this dataset. The improvement is significant at the confidence level $p < 10^{-2}$ compared with fine-tuned NN stacking model (95.32). This indicates that NN multi-view training and stacking model each provide unique advantages for heterogeneous annotations.

6.4 Speed Test

We compare the efficiencies of neural and discrete multi-view training by running our models and the model of Li et al. (2015)[4] with default configurations on the *CTB5* training data. The CRF baseline is adapted from Li et al. (2015). All the systems are implemented in C++ running on an Intel E5-1620 CPU. The results are shown in Table 3.

The NN baseline model is slower than the CRF baseline model. This is due to the higher computation cost of a deep neural network on a CPU. Compared with the CRF baseline, the CRF multi-view model is significantly slower because of its large feature set and the multi-label search space. However, the NN multi-view model achieves almost the same time cost with the NN baseline, and is much more efficient than the CRF counterpart. This shows the efficiency advantage of the NN multi-view model by parameter sharing and output splitting.

7 Related Work

Early research on heterogeneous annotations focuses on annotation conversion. For example, Gao et al. (2004) proposed a transformation-based method to convert the annotation style of a word segmentation corpus to that of another. Manually designed transformation templates are used, which makes it difficult to generalize the method to other tasks and treebanks.

Jiang et al. (2009) described a stacking-based model for heterogeneous annotations, using a pipeline to integrate the knowledge from one corpus to another. Sun and Wan (2012) proposed a structure-based stacking model, which makes use of structured features such as sub-words for model combination. These feature integration is stronger compared to those of Jiang et al. (2009). Johansson (2013) introduced path-based feature templates in using one parser to guide another. In contrast to the above discrete methods, our neural stacking method offers further feature integration by directly connecting the feature layer of the source tagger with the input layer of the target tagger. It also allows the fine-tuning of the source tagger. As one of the reviewers mentioned, two extensions of CRFs, dynamic CRFs (Sutton et al., 2004) and hidden-state CRFs (Quattoni et al., 2004), can also perform similar deep integration and fine-tuning.

For multi-view training, Johansson (2013) used a shared feature representation along with separate individual feature representation for each treebank. Qiu et al. (2013) proposed a multi-task learning model to jointly predict two labelsets given an input sentences. The joint model uses the union of baseline features for each labelset, without considering additional features to capture the interaction between the two labelsets. Li et al. (2015) improves upon this method by using a tighter integration between the two labelsets, treating the Cartesian product of the base labels as a single combined labelset, and exploiting joint features from two labelsets. Though capturing label interaction, their method suffers speed penalty from the sharply increased search space. In contrast to their methods, our neural approach enables parameter sharing in the hidden layers, thereby modeling label interaction without directly combining the two output labelsets. This leads to a lean model with almost the same time efficiency as a single-label baseline.

Recently, Zhang and Weiss (2016) proposed a stack-propagation model for learning a stacked pipeline of POS tagging and dependency parsing. Their method is similar to our neural stacking in fine-tuning the stacked module which yields features for the target model. While their multi-task learning is on heterogenous tasks, our multi-task learning is

[4]http://hlt.suda.edu.cn/zhli/resources/zhenghua-acl2015-resources.zip

defined on heterogenous treebanks.

8 Conclusion

We investigated two methods for utilizing heterogeneous annotations for neural network models, showing that they have respective advantages compared to their discrete counterparts. In particular, neural stacking allows tighter feature integration compared to discrete stacking, and neural multi-view training is free from the feature and efficiency constraints of discrete one. On a standard *CTB* test, the neural method gives the best integration effect, with a multi-view training model enjoying the same speed as its single treebank baseline.

Acknowledgments

The corresponding author is Yue Zhang. We thank Zhenghua Li and Meishan Zhang for providing data and the anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by Singapore Ministry of Education Tier 2 Grant T2MOE201301 and Natural Science Foundation of China (61379086).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Leo Breiman. 1996. Stacked regressions. *Machine learning*, 24(1):49–64.

Richard A Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias1. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Citeseer.

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

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

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

Hal Daumé III. 2007. Frustratingly easy domain adaptation. pages 256–263.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependeny parsing with stack long short-term memory. In *Proceedings of the 53^{rd} Annual Meeting of the Association of Computational Linguistics and the 7^{th} International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*. ACL.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133.

Jianfeng Gao, Andi Wu, Mu Li, Chang-Ning Huang, Hongqiao Li, Xinsong Xia, and Haowei Qin. 2004. Adaptive chinese word segmentation. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 462. Association for Computational Linguistics.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN*, 1:58563–58230.

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

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

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

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging: a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International*

Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 522–530. Association for Computational Linguistics.

Wenbin Jiang, Yajuan , Liang Huang, and Qun Liu. 2015. Automatic adaptation of annotations. *Computational Linguistics*, 41(1):1–29.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *HLT-NAACL*, pages 127–137.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled sequence labeling on heterogeneous annotations: pos tagging as a case study. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1783–1792.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Tagging the web: Building a robust web tagger with neural network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–154, Baltimore, Maryland, June. Association for Computational Linguistics.

Joakim Nivre and Ryan T McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL*, pages 950–958.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL*, pages 293–303.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Xipeng Qiu, Jiayi Zhao, and Xuanjing Huang. 2013. Joint chinese word segmentation and pos tagging on heterogeneous annotated corpora with multiple task learning. In *EMNLP*, pages 658–668.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2004. Conditional random fields for object recognition. *Advances in Neural Information Processing Systems*, pages 1097–1104.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 133–143. Association for Computational Linguistics.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for chinese lexical processing with heterogeneous annotations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 232–241. Association for Computational Linguistics.

Charles Sutton, Andrew Mccallum, and Khashayar Rohanimanesh. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *ICML*, pages 693–723.

Heike Telljohann, Erhard W Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2006. Stylebook for the tübingen treebank of written german (tüba-d/z). In *Seminar fur Sprachwissenschaft, Universitat Tubingen, Tubingen, Germany*.

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

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Dekai Wu, Grace Ngai, and Marine Carpuat. 2003. A stacked, voted, stacked model for named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 200–203. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based

and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.

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

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566. Association for Computational Linguistics.

Meishan Zhang, Jie Yang, Zhiyang Teng, and Yue Zhang. 2016. Libn3l:a lightweight package for neural nlp. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July. Association for Computational Linguistics.

LAMB: A Good Shepherd of Morphologically Rich Languages

Sebastian Ebert and **Thomas Müller** and **Hinrich Schütze**
Center for Information and Language Processing
LMU Munich, Germany
ebert@cis.lmu.de

Abstract

This paper introduces STEM and LAMB, embeddings trained for stems and lemmata instead of for surface forms. For morphologically rich languages, they perform significantly better than standard embeddings on word similarity and polarity evaluations. On a new WordNet-based evaluation, STEM and LAMB are up to 50% better than standard embeddings. We show that both embeddings have high quality even for small dimensionality and training corpora.

1 Introduction

Despite their power and prevalence, embeddings, i.e., (low-dimensional) word representations in vector space, have serious practical problems. First, large text corpora are necessary to train high-quality embeddings. Such corpora are not available for underresourced languages. Second, morphologically rich languages (MRLs) are a challenge for standard embedding models because many inflectional forms are rare or absent even in a large corpus. For example, a Spanish verb has more than 50 forms, many of which are rarely used. This leads to missing or low quality embeddings for such inflectional forms, even for otherwise frequent verbs, i.e., sparsity is a problem. Therefore, we propose to compute normalized embeddings instead of embeddings for surface/inflectional forms (referred to as *forms* throughout the rest of the paper): STem EMbeddings (STEM) for word stems and LemmA eMBeddings (LAMB) for lemmata.

Stemming is a heuristic approach to reducing form-related sparsity issues. Based on simple rules, forms are converted into their stem.[1] However, often the forms of one word are converted into several different stems. For example, present indicative forms of the German verb "brechen" (to break) are mapped to four different stems ("brech", "brich", "bricht", "brecht"). A more principled solution is lemmatization. Lemmatization unites many individual forms, many of which are rare, in one equivalence class, represented by a single lemma. Stems and equivalence classes are more frequent than each individual form. As we will show, this successfully addresses the sparsity issue.

Both methods can learn high-quality semantic representations for rare forms and thus are most beneficial for MRLs as we show below. Moreover, less training data is required to train lemma embeddings of the same quality as form embeddings. Alternatively, we can train lemma embeddings that have the same quality but fewer dimensions than form embeddings, resulting in more efficient applications.

If an application such as parsing requires inflectional information, then stem and lemma embeddings may not be a good choice since they do not contain such information. However, NLP applications such as similarity benchmarks (e.g., MEN (Bruni et al., 2014)) and (as we show below) polarity classification are semantic and are largely

[1] In this paper, we use the term "stem" not in its linguistic meaning, but to refer to the character string that is produced when a stemming algorithm like SNOWBALL is applied to a word form. The stem is usually a prefix of the word form, but some orthographic normalization (e.g., "possibly" → "possible" or "possibli") is often also performed.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 742–752,
Austin, Texas, November 1-5, 2016.

independent of inflectional morphology.

Our contributions are the following. (i) We introduce the normalized embeddings STEM and LAMB and show their usefulness on different tasks for five languages. This paper is the first study that comprehensively compares stem/lemma-based with form-based embeddings for MRLs. (ii) We show the advantage of normalization on word similarity benchmarks. Normalized embeddings yield better performance for MRL languages on most datasets (6 out of 7 datasets for German and 2 out of 2 datasets for Spanish). (iii) We propose a new intrinsic relatedness evaluation based on WordNet graphs and publish datasets for five languages. On this new evaluation, LAMB outperforms form-based baselines by a big margin. (iv) STEM and LAMB outperform baselines on polarity classification for Czech and English. (v) We show that LAMB embeddings are efficient in that they are high-quality for small training corpora and small dimensionalities.

2 Related Work

There have been a large number of studies on English, a morphologically simple language, that show that the effect of normalization, in particular stemming, is different for different applications. For instance, Karlgren and Sahlgren (2001) analyze the impact of morphological analysis on creating word representations for synonymy detection. They compare several stemming methods. Bullinaria and Levy (2012) use stemming and lemmatization before training word representations. The improvement of morphological normalization in both studies is moderate in the best case. Melamud et al. (2014) compute lemma embeddings to predict related words given a query word. They do not compare form and lemma representations.

A finding about English morphology does not provide insight into what happens with the morphology of an MRL. In this paper we use English to provide a data point for morphologically poor languages. Although we show that normalization for embeddings increases performance significantly on some applications – a novel finding to the best of our knowledge – morphologically simple languages (for which normalization is expected to be less important) are not the main focus of the paper. Instead, MRLs are the main focus. For these, we show large improvements on several tasks.

Recently, Köper et al. (2015) compared form and lemma embeddings on English and German focusing on morpho-syntactic and semantic relation tasks. Generally, they found that lemmatization has limited impact. We extensively study MRLs and find a strong improvement on MRLs when using normalization, on intrinsic as well as extrinsic evaluations.

Synonymy detection is a well studied problem in the NLP community (Turney, 2001; Turney et al., 2003; Baroni and Bisi, 2004; Ruiz-Casado et al., 2005; Grigonytė et al., 2010). Rei and Briscoe (2014) classify hyponomy relationships through embedding similarity. Our premise is that semantic similarity comprises all of these relations and more. Our ranking-based word relation evaluation addresses this issue. Similar to Melamud et al. (2014), our motivation is that, in contrast to standard word similarity benchmarks, large resources can be automatically generated for any language with a WordNet. This is also exploited by Tsvetkov et al. (2015). Their intrinsic evaluation method requires an annotated corpus, e.g., annotated with WordNet supersenses. Our approach requires only the WordNet.

An alternative strategy of dealing with data sparsity is presented by Soricut and Och (2015). They compute morphological features in an unsupervised fashion in order to construct a form embedding by the combination of the word's morphemes. We address scenarios (such as polarity classification) in which morphological information is less important, thus morpheme embeddings are not needed.

3 Stem/Lemma Creation

The main hypothesis of this work is that normalization addresses sparsity issues, especially for MRLs, because although a particular word form might not have been seen in the text, its stem or lemma is more likely to be known. For all stemming experiments we use SNOWBALL,[2] a widely used stemmer. It normalizes a form based on deterministic rules, such as *replace the suffix 'tional' by 'tion'* for English.

For lemmatization we use the pipeline version of the freely available, high-quality lemmatizer LEM-

[2] snowball.tartarus.org

MING (Müller et al., 2015). Since it is a language-independent token-based lemmatizer it is especially suited for our multi-lingual experiments. Moreover, it reaches state-of-the-art performance for the five languages that we study. We train the pipeline using the Penn Treebank (Marcus et al., 1993) for English, SPMRL 2013 shared task data (Seddah et al., 2013) for German and Hungarian, and CoNLL 2009 (Hajič et al., 2009) datasets for Spanish and Czech. We additionally use a unigram list extracted from Wikipedia datasets and the ASPELL dictionary of each language.[3]

4 Experiments

4.1 Word Similarity

Our first experiment evaluates how well STEM/LAMB embeddings predict human word similarity judgments. Given a pair of words (m, n) with a human-generated similarity value and a set of embeddings E we compute their similarity as cosine similarity. For form embeddings E^F, we directly use the embeddings of the word pairs' forms (E^F_m and E^F_n) and compute their similarity. For STEM we use $E^S_{stem(w)}$, where $stem(w)$ is the stem of w. For LAMB we use $E^L_{lemma(w)}$, where $lemma(w)$ is the lemma of w; we randomly select one of w's lemmata if there are several. We conduct experiments on English (en), German (de) and Spanish (es). Table 1 gives dataset statistics.

For good performance, high-quality embeddings trained on large corpora are required. Hence, the training corpora for German and Spanish are web corpora taken from COW14 (Schäfer, 2015). Preprocessing includes removal of XML, conversion of HTML characters, lowercasing, stemming using SNOWBALL and lemmatization using LEMMING. We use the entire Spanish corpus (3.7 billion tokens), but cut the German corpus to approximately 8 billion tokens to be comparable to Köper et al. (2015). We train CBOW models (Mikolov et al., 2013) for forms, stems and lemmata using WORD2VEC[4] with the following settings: 400 dimensions, symmetric context of size 2 (no dynamic window), 1 training iteration, negative sampling with 15 samples, a learning rate of 0.025, minimum count of words of 50, and a sampling parameter of 10^{-5}. CBOW is chosen, because it trains much faster than skip-gram, which is beneficial on these large corpora.

Since the morphology of English is rather simple we do not expect STEM and LAMB to reach or even surpass highly optimized systems on any word similarity dataset (e.g., Bruni et al. (2014)). Therefore, for practical reasons we use a smaller training corpus, namely the preprocessed and tokenized Wikipedia dataset of Müller and Schütze (2015).[5] Embeddings are trained with the same settings (using 5 iterations instead of only 1, due to the smaller size of the corpus: 1.8 billion tokens).

Table 1 shows results. We also report the Spearman correlation on the vocabulary intersection, i.e., only those word pairs that are covered by the vocabularies of all models.

Results. Although English has a simple morphology, LAMB improves over form performance on MEN and SL. A tie is achieved on RW. These are the three largest English datasets, giving a more reliable result. Both models perform comparably on WS. Here, STEM is ahead by 1 point. Forms are better on the small datasets MC and RG, where a single word pair can have a large influence on the result. Additionally, these are datasets with high frequency forms, where form embeddings can be well trained. Because of the simple morphology of English, STEM/LAMB do not outperform forms or only by a small margin and thus they cannot compete with highly optimized state-of-the-art systems.[6]

On German, both STEM and LAMB perform better on all datasets except WS. We set the new state-of-the-art of 0.79 on Gur350 (compared to 0.77 (Szarvas et al., 2011)) and 0.39 on ZG (compared to 0.25 (Botha and Blunsom, 2014)); 0.83 on Gur65 (compared to 0.79 (Köper et al., 2015)) is the best performance of a system that does not need additional knowledge bases (cf. Navigli and Ponzetto (2012), Szarvas et al. (2011)).

LAMB's results on Spanish are equally good. 0.82 on MC and 0.58 on WS are again the best per-

[3] ftp://ftp.gnu.org/gnu/aspell/dict

[4] code.google.com/p/word2vec/

[5] cistern.cis.lmu.de/marmot/naacl2015

[6] Baroni et al. (2014)'s numbers are higher on some of the datasets for the *best* of 48 different parameter configurations. In contrast, we do not tune parameters.

formances of a system not requiring an additional knowledge base (cf. Navigli and Ponzetto (2012)). The best performance before was 0.64 for MC and 0.50 for WS (both Hassan and Mihalcea (2009)). STEM cannot improve over form embeddings, showing the difficulty of Spanish morphology.

4.2 Word Relations

Word similarity benchmarks are not available for many languages and are expensive to create. To remedy this situation, we create word similarity benchmarks that leverage WordNets, which are available for a great number of languages.

Generally, a representation is deemed good if words related by a lexical relation in WordNet – synonymy, hyponymy etc. – have high cosine similarity with this representation. Since the gold standard necessary for measuring this property of a representation can be automatically derived from a WordNet, we can create very large similarity benchmarks with up to 50k lemmata for the five languages we investigate: Czech, English, German, Hungarian and Spanish.

We view each WordNet as a graph whose edges are the lexical relations encoded by the WordNet, e.g., synonymy, antonymy and hyponymy. We then define $\mathcal{L}$ as the set of lemmata in a WordNet and the distance $d(l, l')$ between two lemmata l and l' as the length of the *shortest path* connecting them in the graph. The k-neighborhood $N^k(l)$ of l is the set of lemmata l' that have distance k or less, excluding l: $N^k(l) := \{l' | d(l, l') \leq k, l \neq l'\}$. The rank of l for an embedding set E is defined as:

$$\text{rank}_E^k(l) := \underset{i}{\text{argmin}}\, l_i \in N^k(l) \quad (1)$$

where l_i is the lemma at position i in the list of all lemmata in the WordNet, ordered according to cosine similarity to l in descending order. We restrict $i \in [1, 10]$ and set $k = 2$ for all experiments in this paper. We omit the indexes k and E when they are clear from context.

To measure the quality of a set of embeddings we compute the mean reciprocal rank (MRR) on the rank results of all lemmata:

$$\text{MRR}_E = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \frac{1}{\text{rank}_E(l)} \quad (2)$$

We create large similarity datasets for five languages: Czech (cz), English (en), German (de), Hungarian (hu) and Spanish (es) by extracting all lemmata from the WordNet version of the respective language. For English and Spanish we use the preprocessed WordNets from the Open Multilingual WordNet (Bond and Paik, 2012). We use the Czech and Hungarian WordNets (PALA and SMRZ, 2004; Miháltz et al., 2008) and GermaNet (Hamp and Feldweg, 1997) for German. We keep all lemmata that have a known form in the form embeddings and that exist in the lemma embeddings. Moreover, we filter out all synsets that contain only one lemma and discard all multiword phrases. The split into development and test sets is done in a way that the distribution of synset sizes (i.e., the number of lemmata per synset) is nearly equal in both sets.

The number of lemmata in our evaluation sets can be found in Table 2. For more insight, we report results on all parts-of-speech (POS), as well as separately for nouns (n), verbs (v) and adjectives (a).[7] The data is provided as supplementary material.[8]

We propose the following models for the embedding evaluation. For form embeddings we compare three different strategies, a realistic one, an optimistic one and a lemma approximation strategy. In the realistic strategy (*form real*), given a query lemma we randomly sample a form, for which we then compute the k-neighborhood. If the neighbors contain multiple forms of the same equivalence class, we exclude the repetitions and use the next neighbors instead. For instance, if *house* is already a neighbor, then *houses* will be skipped. The optimistic strategy (*form opt*) works similarly, but uses the embedding of the most frequent surface form of a lemma. This is the most likely form to perform best in the form model. This strategy presupposes the availability of information about lemma and surface form counts. As a baseline lemma approximation strategy, we sum up all surface form embeddings that belong to one equivalence class (*form sum*). For STEM we repeat the same experiments as described for forms, leading to *stem real*, *stem opt* and *stem sum*.

For embedding training, Wikipedia comes as a

[7] The all-POS setting includes all POS, not just n, v, a.

[8] All supplementary material is available at `www.cis.uni-muenchen.de/ebert/`

				full vocabulary				vocabulary intersection			
lang	dataset	reference	pairs	form	STEM	LAMB	cov.	form	STEM	LAMB	cov.
de	Gur30	Gurevych (2005)	29	0.76	**0.83**	0.80	29, 29, 29	0.76	**0.83**	0.80	29
	Gur350	Gurevych (2005)	350	0.74	**0.79**	**0.79**	336, 340, 339	0.74	**0.79**	**0.79**	336
	Gur65	Gurevych (2005)	65	0.80	**0.83**	0.82	65, 65, 65	0.80	**0.83**	0.82	65
	MSL	Leviant and Reichart (2015)	999	0.44	0.44	**0.47**	994, 995, 995	0.44	0.44	**0.47**	994
	MWS	Leviant and Reichart (2015)	350	0.60	0.61	**0.62**	348, 350, 350	0.60	**0.61**	**0.61**	348
	WS	Köper et al. (2015)	280	**0.72**	**0.72**	0.71	279, 280, 280	**0.72**	0.71	0.71	279
	ZG	Zesch and Gurevych (2006)	222	0.36	0.38	**0.39**	200, 207, 208	0.36	0.40	**0.41**	200
en	MC	Miller and Charles (1991)	30	**0.82**	0.77	0.80	30, 30, 30	**0.82**	0.77	0.80	30
	MEN	Bruni et al. (2014)	1000	0.72	0.73	**0.74**	1000, 1000, 1000	0.72	0.73	**0.74**	1000
	RG	Rubenstein et al. (1965)	65	**0.82**	0.79	0.79	65, 65, 65	**0.82**	0.79	0.79	65
	RW	Luong et al. (2013)	2034	**0.47**	**0.47**	**0.47**	1613, 1947, 1819	0.47	0.47	**0.48**	1613
	SL	Hill et al. (2014)	999	0.42	0.38	**0.43**	998, 999, 999	0.42	0.38	**0.43**	998
	WS	Finkelstein et al. (2002)	353	0.63	**0.64**	0.63	353, 353, 353	0.63	**0.64**	0.63	353
es	MC	Hassan and Mihalcea (2009)	30	0.70	0.69	**0.82**	30, 30, 30	0.70	0.69	**0.82**	30
	WS	Hassan and Mihalcea (2009)	352	0.54	0.54	**0.58**	350, 352, 352	0.54	0.54	**0.58**	350

Table 1: Word similarity results. The left part shows dataset information. The right part shows Spearman correlation (ρ) for the models with their *full vocabulary* and for the intersection of vocabularies. Coverage is shown for all models in order of appearance. Bold is best per vocabulary and row.

lang	set	all	a	n	v
cz	dev	9694	852	6436	2315
	test	9763	869	6381	2433
de	dev	51682	6347	40674	5018
	test	51827	6491	40623	5085
en	dev	44448	9713	30825	5661
	test	44545	9665	30736	5793
es	dev	12384	1711	8634	1989
	test	12476	1727	8773	1971
hu	dev	19387	1953	15268	2057
	test	19486	1928	15436	2011

Table 2: Number of lemmata in WordNet datasets

natural choice as corpus, because it is available for many languages. Therefore, we use the preprocessed and tokenized Wikipedia datasets of Müller and Schütze (2015). We train 50-dimensional skip-gram embeddings (Mikolov et al., 2013) with WORD2VEC on the original, the stemmed and the lemmatized corpus, respectively. Embeddings are trained for all tokens, because we need high coverage; the context size is set to 5, all remaining parameters are left at their default value.[9]

[9]We train smaller embeddings than before, because we have more models to train and training corpora are smaller.

Results. The MRR results in the left half of Table 3 ("unfiltered") show that for all languages and for all POS, *form real* has the worst performance among the form models. This comes at no surprise since this model does barely know anything about word forms and lemmata. The *form opt* model improves these results based on the additional information it has access to (the mapping from lemma to its most frequent form). *form sum* performs similar to *form opt*. For Czech, Hungarian and Spanish it is slightly better (or equally good), whereas for English and German there is no clear trend. There is a large difference between these two models on German nouns, with *form sum* performing considerably worse. We attribute this to the fact that many German noun forms are rare compounds and therefore lead to badly trained form embeddings, which summed up do not lead to high quality embeddings either.

Among the stemming models, *stem real* also is the worst performing model. We can further see that for all languages and almost all POS, *stem sum* performs worse than *stem opt*. That indicates that stemming leads to many low-frequency stems or many words sharing the same stem. This is especially apparent in Spanish verbs. There, the stemming models are clearly inferior to form models.

Overall, LAMB performs best for all languages and POS types. Most improvements of LAMB are

significant. The improvement over the best form-model reaches up to 6 points (e.g., Czech nouns). In contrast to *form sum*, LAMB improves over *form opt* on German nouns. This indicates that the sparsity issue is successfully addressed by LAMB.

In general, morphological normalization in terms of stemming or lemmatization improves the result on all languages, leading to an especially substantial improvement on MRLs. For the morphologically very rich languages Czech and Hungarian, the relative improvement of STEM or LAMB to form-based models is especially high, e.g., Hungarian all: 50%. Moreover, we find that MRLs yield lower absolute performance. This confirms the findings of Köper et al. (2015). Surprisingly, LAMB yields better performance on English despite its simple morphology.

The low absolute results – especially for Hungarian – show that we address a challenging task and that our new evaluation methodology is a good evaluation for new types of word representations.

For further insight, we restrict the nearest neighbor search space to those lemmata that have the same POS as the query lemma. The general findings in the right half of Table 3 ("filtered") are similar to the unrestricted experiment: Normalization leads to superior results. The *form real* and *stem real* models yield the lowest performance. *Form opt* improves the performance and *form sum* is better on average than *form opt*. *Stem sum* can rarely improve on *stem opt*. The best stemming model most often is better than the best form model. LAMB can benefit more from the POS type restriction than the form models. The distance to the best form model generally increases, especially on German adjectives and Spanish verbs. In all cases except on English adjectives, LAMB yields the best performance. Again, in almost all cases LAMB's improvement over the form-models is significant.

4.3 Polarity Classification

Our first two evaluations were intrinsic. We now show the benefit of normalization on an extrinsic task. The task is classification of Czech movie reviews (CSFD, Habernal et al. (2013)) into positive, negative or neutral (Table 4). We reimplement lingCNN (Ebert et al., 2015), a Convolutional Neural Network that uses linguistic information to improve polarity classification. This model reaches close to state-of-the-art performance on data of the SemEval 2015 Task 10B (message level polarity). LingCNN takes several features as input: (i) embedding features, (ii) linguistic features at word level and (iii) linguistic features at review level.

We reuse the 50-dimensional Wikipedia embeddings from Section 4.2 and compare three experimental conditions: using forms, STEM and LAMB.

Linguistic word level features are: (i) SubLex 1.0 sentiment lexicon (Veselovská and Bojar, 2013) (two binary indicators that word is marked positive/negative); (ii) SentiStrength[10] (three binary indicators that word is an emoticon marked as positive/negative/neutral); (iii) prefix "ne" (binary negation indicator in Czech).[11]

All word level features are concatenated to form a single word representation of the review's input words. The concatenation of these representations is the input to a convolution layer, which has several filters spanning the whole representation height and several representations (i.e., several words) in width. The output of the convolution layer is input to a k-max pooling layer (Kalchbrenner et al., 2014). The max values are concatenated with the following linguistic review level features: (i) the count of elongated words, such as "cooool"; (ii) three count features for the number of positive/negative/neutral emoticons using the SentiStrength list; (iii) a count feature for punctuation sequences, such as "!!!"; (iv) and a feature that counts the number of negated words. (v) A final feature type comprises one count feature each for the number of sentiment words in a review, the sum of sentiment values of these words as provided by the sentiment lexicon, the maximum sentiment value and the sentiment value of the last word (Mohammad et al., 2013). The concatenation of max values and review level features is then forwarded into a fully-connected three-class (positive, negative, neutral) softmax layer. We train lingCNN with AdaGrad (Duchi et al., 2011) and early stopping, batch size = 100, 200 filters per width of 3-6; k-max pooling with $k = 5$; learning rate 0.01; and ℓ_2 regularization ($\lambda = 5 \cdot 10^{-5}$).

We also perform this experiment for English on

[10] sentistrength.wlv.ac.uk/

[11] We disregard words with the prefix "nej", because they indicate superlatives. Exceptions are common negated words with this prefix, such as "nejsi" (engl. "you are not").

		unfiltered							filtered						
		form			STEM				form			STEM			
lang	POS	real	opt	sum	real	opt	sum	LAMB	real	opt	sum	real	opt	sum	LAMB
cz	a	0.03	0.04	0.05	0.02	0.05	0.05	**0.06**	0.03‡	0.05†	0.07	0.04†	0.08	0.08	**0.09**
	n	0.15‡	0.21‡	0.24‡	0.18‡	0.27‡	0.26‡	**0.30**	0.17‡	0.23‡	0.26‡	0.20‡	0.29‡	0.28‡	**0.32**
	v	0.07‡	0.13‡	0.16†	0.08‡	0.14‡	0.16‡	**0.18**	0.09‡	0.15‡	0.17‡	0.09‡	0.17†	0.18	**0.20**
	all	0.12‡	0.18‡	0.20‡	0.14‡	0.22‡	0.21‡	**0.25**	-	-	-	-	-	-	-
de	a	0.14‡	0.22‡	0.25†	0.17‡	0.26	0.21‡	**0.27**	0.17‡	0.25‡	0.27‡	0.23‡	**0.33**	**0.33**	**0.33**
	n	0.23‡	0.35‡	0.30‡	0.28‡	0.35†	0.33‡	**0.36**	0.24‡	0.36‡	0.31‡	0.28‡	0.36	0.35‡	**0.37**
	v	0.11‡	0.19‡	0.18‡	0.11‡	0.22	0.18‡	**0.23**	0.13‡	0.20‡	0.21‡	0.13‡	0.24‡	0.23‡	**0.26**
	all	0.21‡	0.32‡	0.28‡	0.24‡	0.33†	0.30‡	**0.34**	-	-	-	-	-	-	-
en	a	0.22‡	0.25‡	0.24‡	0.16‡	0.26‡	0.25‡	**0.28**	0.25‡	0.28‡	0.28‡	0.18‡	0.29‡	**0.32**	0.31
	n	0.24‡	0.27‡	0.28‡	0.22‡	**0.30**	0.28‡	**0.30**	0.25‡	0.28‡	0.29‡	0.23‡	0.31†	0.31‡	**0.32**
	v	0.29‡	0.35‡	**0.37**	0.17‡	0.35	0.24‡	**0.37**	0.33‡	0.39‡	0.42‡	0.21‡	0.42†	0.39‡	**0.44**
	all	0.23‡	0.26‡	0.27‡	0.20‡	0.28‡	0.25‡	**0.29**	-	-	-	-	-	-	-
es	a	0.20‡	0.23‡	0.23‡	0.08‡	0.21‡	0.18‡	**0.27**	0.21‡	0.25‡	0.26‡	0.10‡	0.26‡	0.26‡	**0.30**
	n	0.21‡	0.25‡	0.25‡	0.16‡	0.25‡	0.23‡	**0.29**	0.22‡	0.26‡	0.27‡	0.17‡	0.27‡	0.26‡	**0.30**
	v	0.19‡	0.35†	0.36	0.11‡	0.29‡	0.19‡	**0.38**	0.22‡	0.36‡	0.36‡	0.16‡	0.36‡	0.33‡	**0.42**
	all	0.20‡	0.26‡	0.26‡	0.14‡	0.24‡	0.21‡	**0.30**	-	-	-	-	-	-	-
hu	a	0.02‡	0.06‡	0.06‡	0.05‡	0.08	0.08	**0.09**	0.04‡	0.08‡	0.08‡	0.06‡	**0.12**	0.11	**0.12**
	n	0.01‡	0.04‡	0.05‡	0.03‡	**0.07**	0.06‡	**0.07**	0.01‡	0.04‡	0.05‡	0.04‡	**0.07**†	0.06‡	**0.07**
	v	0.04‡	0.11‡	0.13‡	0.07‡	0.14‡	0.15	**0.17**	0.05‡	0.13‡	0.14‡	0.07‡	0.15‡	0.16†	**0.19**
	all	0.02‡	0.05‡	0.06‡	0.04‡	0.08‡	0.07‡	**0.09**	-	-	-	-	-	-	-

Table 3: Word relation results. MRR per language and POS type for all models. *unfiltered* is the unfiltered nearest neighbor search space; *filtered* is the nearest neighbor search space that contains only one POS. ‡ (resp. †): significantly worse than LAMB (sign test, $p < .01$, resp. $p < .05$). Best unfiltered/filtered result per row is in bold.

the SemEval 2015 Task 10B dataset (cf. Table 4). We reimplement Ebert et al. (2015)'s lexicon features. They exploit the fact that there are many more sentiment lexicons available in English. Other word level features are the same as above. Sentiment count features at review level are computed separately for the entire tweet, for all hashtag words and for each POS type (Ebert et al., 2015).

Considering the much smaller dataset size and shorter sentences of the SemEval data we chose the following hyperparameters: 100k most frequent word types, 100 filters per filter width of 2-5; and k-max pooling with $k = 1$.

Results. Table 5 lists the 10-fold cross-validation results (accuracy and macro F_1) on the CSFD dataset. LAMB/STEM results are consistently better than form results.

In our analysis, we found the following example for the benefit of normalization: "popis a název zajmavý a film je taková filmařská prasárna" (engl. "description and title are interesting, but it is bad film-making"). This example is correctly classified as negative by the LAMB model because it has an embedding for "prasárna" (bad, smut) whereas the form model does not.

The out-of-vocabulary counts for form and LAMB on the first fold of the CSFD experiment are 26.3k and 25.5k, respectively. The similarity of these two numbers suggests that the quality of word embeddings (form vs. LAMB) are responsible for the performance gain.

On the SemEval data, LAMB improves the results over form and stem (cf. Table 5).[12] Hence, LAMB can still pick up additional information despite the simple morphology of English. This is probably due to better embeddings for rare words. The SemEval 2015 winner (Hagen et al., 2015) is a highly domain-dependent and specialized system that we do not outperform.

In the introduction, we discussed that normalization removes inflectional information that is necessary for NLP tasks like parsing. For polarity classification, comparatives and superlatives can be important. Further analysis is necessary to deter-

[12] To be comparable with published results we report the macro F_1 of positive and negative classes.

dataset	total	pos	neg	neu
CSFD	91379	30896	29716	30767
SemEval train	9845	3636	1535	4674
SemEval dev	3813	1572	601	1640
SemEval test	2390	1038	365	987

Table 4: Polarity classification datasets

lang	features	acc	F_1
cz	Brychcin et al. (2013)	-	**81.53**
	form	80.86	80.75
	STEM	**81.51**	81.39
	LAMB	81.21	81.09
en	Hagen et al. (2015)	-	**64.84**
	form	66.78	62.21
	STEM	66.95	62.06
	LAMB	**67.49**	63.01

Table 5: Polarity classification results. Bold is best per language and column.

mine whether their normalization hurts in our experiments. However, note that we evaluate on polarity only, not on valence.

5 Analysis

Normalized embeddings deal better with sparsity than form embeddings. In this section, we demonstrate two additional benefits of LAMB based on its robustness against sparsity.

Embedding Size. We now show that LAMB can train embeddings with fewer dimensions on the same amount of data and still reach the same performance as larger form embeddings. We repeat the word relation experiments of Section 4.2 (all POS) and train all models with embedding sizes 10, 20, 30 and 40 for Spanish. We choose Spanish because it has richer morphology than English and more training data than Czech and Hungarian.

Figure 1 depicts the MRR results of all models with respect to embedding size. The relative ranking of form models is *real* $<$ *opt* $<$ *sum*. That comes from the additional information the more complex models have access to. All stemming models reach lower performance than their form counterparts (similar to results in Table 3). That suggests that stemming is not a proper alternative to correctly dealing with Spanish morphology. LAMB reaches higher performance than *form real* with already 20 dimensions. The 30 dimensional LAMB model is better than all other models. Thus, we can create lower-dimensional lemma embeddings that are as good as higher-dimensional form embeddings; this has the benefits of reducing the number of parameters in models using these embeddings and of reducing training times and memory consumption.

Corpus Size. Our second hypothesis is that less training data is necessary to train good embeddings. We create 10 training corpora consisting of the first k percent, $k \in \{10, 20, \ldots, 100\}$, of the randomized Spanish Wikipedia corpus. With these 10 subcorpora we repeat the word relation experiments of Section 4.2 (all POS). As query lemmata, we use the lemmata from before that exist in all subcorpora.

Figure 2 shows that the relative ranking among the models is the same as before. This time however, *form sum* yields slightly better performance than *form opt*, especially when little training data is available. The stemming models again are inferior to their form counterparts. Only *stem opt* is able to reach performance similar to *form opt*. LAMB always reaches higher performance than *form real*, even when only 10% of the training corpus is used. With 30% of the training corpus, LAMB surpasses the performance of the other models.[13] Again, by requiring less than 30% of the training data, embedding training becomes much more efficient. Furthermore, in low-resource languages that lack the availability of a large homogeneous corpus, LAMB can still be trained successfully.

6 Conclusion

We have presented STEM and LAMB, embeddings based on stems and lemmata. In three experiments we have shown the superiority compared to commonly used form embeddings. Especially (but not only) on MRLs, where data sparsity is a problem, both normalized embeddings perform better than form embeddings by a large margin. In a new challenging WordNet-based experiment we have shown four methods of adding morphological information

[13] Recall that *form opt* is similar to an approach that is used in most systems that have embeddings, which just use the available surface forms.

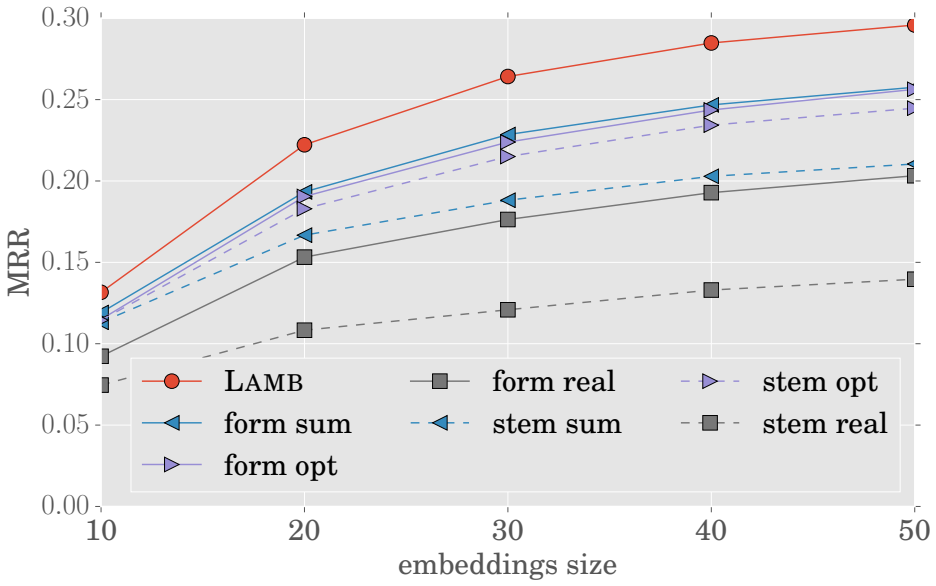


Figure 1: Embedding size analysis

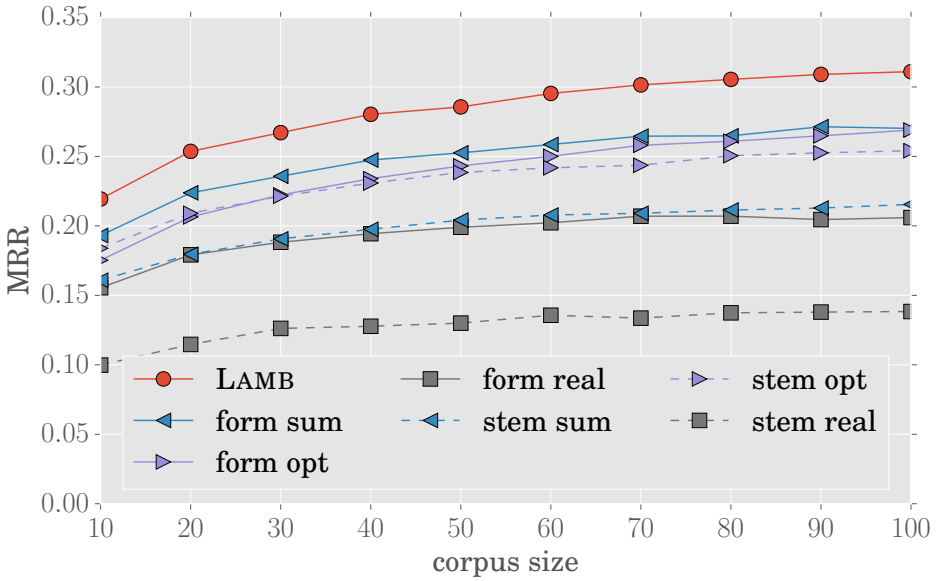


Figure 2: Corpus size analysis

(*opt*, *sum*, STEM, LAMB). Here, LAMB is the best of the proposed ways of using morphological information, consistently reaching higher performance, often by a large margin. STEM methods are not consistently better, indicating that the more principled way of normalization as done by LAMB is to be preferred. The datasets are available as supplementary material at `www.cis.uni-muenchen.de/ebert/`.

Our analysis shows that LAMB needs fewer embedding dimensions and less embedding training data to reach the same performance as form embeddings, making LAMB appealing for underresourced languages.

As morphological analyzers are becoming more widely available, our method – which is easy to implement, only requiring running the analyzer – should become applicable to more and more languages.

Acknowledgments This work was supported by DFG (grant SCHU 2246/10).

References

Marco Baroni and Sabrina Bisi. 2004. Using cooccurrence statistics and the web to discover synonyms in a technical language. In *Proceedings of LREC*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*.

Francis Bond and Kyonghee Paik. 2012. A Survey of Wordnets and their Licenses. In *Proceedings of the 6th Global WordNet Conference*.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of ICML*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, 49.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior Research Methods*, 44(3):890–907.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12.

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze. 2015. A Linguistically Informed Convolutional Neural Network. In *Proceedings of WASSA*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1).

Gintarė Grigonytė, Joao Cordeiro, Gaël Dias, Rumen Moraliyski, and Pavel Brazdil. 2010. Paraphrase alignment for synonym evidence discovery. In *COLING*.

Iryna Gurevych. 2005. Using the Structure of a Conceptual Network in Computing Semantic Relatedness. In *Proceedings of IJCNLP*.

Ivan Habernal, Tomáš Ptáček, and Josef Steinberger. 2013. Sentiment Analysis in Czech Social Media Using Supervised Machine Learning. In *Proceedings of WASSA*.

Matthias Hagen, Martin Potthast, Michel Büchner, and Benno Stein. 2015. Webis: An Ensemble for Twitter Sentiment Detection. In *Proceedings of SemEval*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a Lexical-Semantic Net for German. In *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual Semantic Relatedness Using Encyclopedic Knowledge. In *Proceedings of EMNLP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *CoRR*, abs/1408.3456.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of ACL*.

Jussi Karlgren and Magnus Sahlgren. 2001. From Words to Understanding. In *Foundations of Real World Intelligence*. CSLI Publications.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual Reliability and "Semantic" Structure of Continuous Word Spaces. In *Proceedings of IWCS*.

Ira Leviant and Roi Reichart. 2015. Judgment Language Matters: Multilingual Vector Space Models for Judgment Language Aware Lexical Semantics. *CoRR*, abs/1508.00106.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of CoNLL*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*.

Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. 2014. Probabilistic Modeling of Joint-context in Distributional Similarity. In *Proceedings of CoNLL*.

Márton Miháltz, Csaba Hatvani, Judit Kuti, György Szarvas, János Csirik, Gábor Prószéky, and Tamás Váradi. 2008. Methods and Results of the Hungarian WordNet Project. In *Proceedings of the 4th Global WordNet Conference*.

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

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1).

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *Proceedings of SemEval*.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of NAACL*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with Lemming. In *Proceedings of EMNLP*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelRelate! A Joint Multilingual Approach to Computing Semantic Relatedness July 22-26, 2012, Toronto, Ontario, Canada. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Karel PALA and Pavel SMRZ. 2004. Building Czech Wordnet. *Romanian Journal of Information Science and Technology*, 7(1-2).

Marek Rei and Ted Briscoe. 2014. Looking for Hyponyms in Vector Space Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014. In *Proceedings of CoNLL*.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of RANLP*.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of CMLC*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, et al. 2013. Overview of the SPMRL 2013 shared task: Cross-Framework evaluation of parsing morphologically rich languages. In *Proceddings of SPMRL*.

Radu Soricut and Franz Josef Och. 2015. Unsupervised Morphology Induction Using Word Embeddings. In *Proceedings of NAACL-HLT*.

György Szarvas, Torsten Zesch, and Iryna Gurevych. 2011. Combining Heterogeneous Knowledge Resources for Improved Distributional Semantic Models. In *Proceedings of CICLing*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of Word Vector Representations by Subspace Alignment. In *Proceedings of EMNLP*.

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. *ACM Transactions on Information Systems*.

Peter D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML*.

Kateřina Veselovská and Ondřej Bojar. 2013. Czech SubLex 1.0.

Torsten Zesch and Iryna Gurevych. 2006. Automatically Creating Datasets for Measures of Semantic Relatedness. In *Proceedings of the Workshop on Linguistic Distances*.

Fast Coupled Sequence Labeling on Heterogeneous Annotations via Context-aware Pruning

Zhenghua Li, Jiayuan Chao, Min Zhang*, Jiwen Yang
Soochow University, Suzhou, China
{zhli13,minzhang,jwyang}@suda.edu.cn, chaojiayuan.china@gmail.com

Abstract

The recently proposed coupled sequence labeling is shown to be able to effectively exploit multiple labeled data with heterogeneous annotations but suffer from severe inefficiency problem due to the large bundled tag space (Li et al., 2015). In their case study of part-of-speech (POS) tagging, Li et al. (2015) manually design context-free tag-to-tag mapping rules with a lot of effort to reduce the tag space.

This paper proposes a context-aware pruning approach that performs token-wise constraints on the tag space based on contextual evidences, making the coupled approach efficient enough to be applied to the more complex task of joint word segmentation (WS) and POS tagging for the first time. Experiments show that using the large-scale People Daily as auxiliary heterogeneous data, the coupled approach can improve F-score by $95.55 - 94.88 = 0.67\%$ on WS, and by $90.58 - 89.49 = 1.09\%$ on joint WS&POS on Penn Chinese Treebank. All codes are released at http://hlt.suda.edu.cn/~zhli.

1 Introduction

In statistical natural language processing, manually labeled data is inevitable for model supervision, but is also very expensive to build. However, due to the long-debated differences in underlying linguistic theories or emphasis of application, there often exist multiple labeled corpora for the same or similar tasks following different annotation guidelines (Jiang et al., 2009). For instance, in Chinese language processing, Penn Chinese Treebank version 5 (CTB5) is a widely used benchmark data and contains about 20 thousand sentences annotated with word boundaries, part-of-speech (POS) tags, and syntactic structures (Xue et al., 2005; Xia, 2000), whereas People's Daily corpus (PD)[1] is a large-scale corpus annotated with words and POS tags, containing about 300 thousand sentences from the first half of 1998 of People's Daily newspaper (Yu et al., 2003). Table 1 gives an example with both CTB and PD annotations. We can see that CTB and PD differ in both word boundary standards and POS tag sets.

	Especially		our	nation	economy	declines	.
CTB	特别是/AD		我/PN	国/NN	经济/NN	下滑/VV	。/PU
PD	特别/d	是/v	我国/n		经济/n	下滑/v	。/w

Table 1: An example of heterogeneous annotations.

Previous work on exploiting heterogeneous data mainly focuses on indirect guide-feature methods. The basic idea is to use one resource to generate extra guide features on another resource (Jiang et al., 2009; Sun and Wan, 2012), which is similar to stacked learning (Nivre and McDonald, 2008). Li et al. (2015) propose a coupled sequence labeling approach that can directly learn and predict two heterogeneous annotations simultaneously. The basic idea is to transform a single-side tag into a set of bundled tags for weak supervision based on the idea of ambiguous labeling. Due to the huge size of the bundled tag space, their coupled model is extremely inefficient. They then carefully design tag-to-tag

*Correspondence author

[1] http://icl.pku.edu.cn/icl_groups/corpustagging.asp

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 753–762,
Austin, Texas, November 1-5, 2016.

mapping rules to constrain the search space. Their case study on POS tagging shows that the coupled model outperforms the guide-feature method. However, the requirement of manually designed mapping rules makes their approach less attractive, since such mapping rules may be very difficult to construct for more complex tasks such as joint word segmentation (WS) and POS tagging.

This paper proposes a context-aware pruning approach that can effectively solve the inefficiency problem of the coupled model, making coupled sequence labeling more generally applicable. Specifically, this work makes the following contributions:

(1) We propose and systematically compare two ways for realizing context-aware pruning, i.e., online and offline pruning. Experiments on POS tagging show that both online and offline pruning can greatly improve the model efficiency with little accuracy loss.

(2) We for the first time apply coupled sequence labeling to the more complex task of joint WS&POS tagging. Experiments show that online pruning works badly due to the much larger tag set while offline pruning works well. Further analysis gives a clear explanation and leads to more insights in learning from ambiguous labeling.

(3) Experiments on joint WS&POS tagging show that our coupled approach with offline pruning improves F-score by $95.55 - 94.88 = 0.67\%$ on WS, and by $90.58 - 89.49 = 1.09\%$ on joint WS&POS on CTB5-test over the baseline, and is also consistently better than the guide-feature method.

2 Coupled Sequence Labeling

Given an input sequence of n tokens, denoted by $\mathbf{x} = w_1...w_n$, coupled sequence tagging aims to simultaneously predict two tag sequences $\mathbf{t}^a = t_1^a...t_n^a$ and $\mathbf{t}^b = t_1^b...t_n^b$, where $t_i^a \in \mathcal{T}^a$ and $t_i^b \in \mathcal{T}^b$ $(1 \leq i \leq n)$, and $\mathcal{T}^a$ and $\mathcal{T}^b$ are two different predefined tag sets. Alternatively, we can view the two tag sequences as one bundled tag sequence $\mathbf{t} = [\mathbf{t}^a, \mathbf{t}^b] = [t_1^a, t_1^b]...[t_n^a, t_n^b]$, where $[t_i^a, t_i^b] \in \mathcal{T}^a \times \mathcal{T}^b$ is called a *bundled tag*.

In this work, we treat CTB as the first-side annotation and PD as the second-side. For POS tagging, $\mathcal{T}^a$ is the set of POS tags in CTB, and $\mathcal{T}^b$ is the set of POS tags in PD, and we ignore the word boundary differences in the two datasets, following Li et al. (2015). We have $|\mathcal{T}^a| = 33$ and $|\mathcal{T}^b| = 38$.

For joint WS&POS tagging, we employ the standard four-tag label set to mark word boundaries, among which `B`, `I`, `E` respectively represent that the concerned character situates at the *begining, inside, end* position of a word, and *S* represents a single-character word. Then, we concatenate word boundary labels with POS tags. For instance, the first three characters in Table 1 correspond to "特/`B@AD` 别/`I@AD` 是/`E@AD`" in CTB, and to "特/`B@d` 别/`E@d` 是/`S@v`" in PD. We have $|\mathcal{T}^a| = 99$ and $|\mathcal{T}^b| = 128$.

2.1 Coupled Conditional Random Field (CRF)

Following Li et al. (2015), we build the coupled sequence labeling model based on a bigram linear-chain CRF (Lafferty et al., 2001). The conditional probability of a bundled tag sequence $\mathbf{t}$ is:

$$p(\mathbf{t}|\mathbf{x}, \tilde{\mathcal{S}}; \theta) = \frac{e^{Score(\mathbf{x},\mathbf{t};\theta)}}{Z(\mathbf{x}, \tilde{\mathcal{S}}; \theta)}$$
$$Z(\mathbf{x}, \tilde{\mathcal{S}}; \theta) = \sum\nolimits_{\mathbf{t} \in \tilde{\mathcal{S}}} e^{Score(\mathbf{x},\mathbf{t};\theta)} \quad (1)$$

where θ is the feature weights; $Z(\mathbf{x}, \tilde{\mathcal{S}}; \theta)$ is the normalization factor; $\tilde{\mathcal{S}}$ is the search space including all legal tag sequences for $\mathbf{x}$. We use $\tilde{\mathcal{T}}_i \subseteq \mathcal{T}^a \times \mathcal{T}^b$ to denote the set of all legal tags for token w_i, so $\tilde{\mathcal{S}} = \tilde{\mathcal{T}}_1 \times \cdots \times \tilde{\mathcal{T}}_n$.

According to the linear-chain Markovian assumption, the score of a bundled tag sequence is:

$$Score(\mathbf{x}, \mathbf{t}; \theta) = \theta \cdot \mathbf{f}(\mathbf{x}, [\mathbf{t}^a, \mathbf{t}^b])$$
$$\sum_{i=1}^{n+1} \theta \cdot \begin{bmatrix} \mathbf{f}_{joint}(\mathbf{x}, i, [t_{i-1}^a, t_{i-1}^b], [t_i^a, t_i^b]) \\ \mathbf{f}_{sep\_a}(\mathbf{x}, i, t_{i-1}^a, t_i^a) \\ \mathbf{f}_{sep\_b}(\mathbf{x}, i, t_{i-1}^b, t_i^b) \end{bmatrix} \quad (2)$$

where $\mathbf{f}(\mathbf{x}, [\mathbf{t}^a, \mathbf{t}^b])$ is the accumulated sparse feature vector; $\mathbf{f}_{joint/sep\_a/sep\_b}(\mathbf{x}, i, t', t)$ share the same list of feature templates, and return local feature vectors for tagging w_{i-1} as t' and w_i as t.

Traditional single-side tagging models can only exploit a single set of separate features $\mathbf{f}_{sep\_a}(.)$ or $\mathbf{f}_{sep\_b}(.)$. In contrast, the coupled model makes

use of all three sets of features. Li et al. (2015) demonstrate that the joint features $\mathbf{f}_{joint}(.)$ capture the implicit mappings between heterogeneous annotations, and the separate features function as back-off features for alleviating the data sparseness problem of the joint features.

For the feature templates, we follow Li et al. (2015) and adopt those described in Zhang and Clark (2008) for POS tagging, and use those described in Zhang et al. (2014b) for joint WS&POS tagging.

2.2 Learn from Incomplete Data

The key challenge for coupled sequence labeling is that both CTB and PD are non-overlapping and each contains only one-side annotations. Based on the idea of ambiguous labeling, Li et al. (2015) first concatenate a single-side tag with many possible second-side tags, and then use the set of bundled tags as possibly-correct references during training.

Suppose $\mathbf{x} = w_1...w_n$ is a training sentence from CTB, and $\mathbf{t}^a = \check{t}_1^a...\check{t}_n^a$ is the manually labeled tag sequence. Then we define $\mathcal{T}_i = \{\check{t}_i^a\} \times \mathcal{T}^b$ as the set of possibly-correct bundled tags, and $\mathcal{S} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_n$ as a exponential-size set of possibly-correct bundled tag sequences used for model supervision.

Given $\mathbf{x}$ and the whole legal search space $\tilde{\mathcal{S}}$, the probability of the possibly-correct space $\mathcal{S} \subseteq \tilde{\mathcal{S}}$ is:

$$p(\mathcal{S}|\mathbf{x}, \tilde{\mathcal{S}}; \theta) = \sum_{\mathbf{t} \in \mathcal{V}} p(\mathbf{t}|\mathbf{x}, \tilde{\mathcal{S}}; \theta) = \frac{Z(\mathbf{x}, \mathcal{S}; \theta)}{Z(\mathbf{x}, \tilde{\mathcal{S}}; \theta)} \quad (3)$$

where $Z(\mathbf{x}, \mathcal{S}; \theta)$ is analogous to $Z(\mathbf{x}, \tilde{\mathcal{S}}; \theta)$ in Eq. (3) but only sums over $\mathcal{S}$.

Given $\mathcal{D} = \{(\mathbf{x}_j, \mathcal{S}_j, \tilde{\mathcal{S}}_j)\}_{j=1}^N$, the gradient of the log likelihood is:

$$\begin{aligned}
&\frac{\partial \mathcal{LL}(\mathcal{D}; \theta)}{\partial \theta} = \frac{\partial \log \sum_j p(\mathcal{S}_j|\mathbf{x}_j, \tilde{\mathcal{S}}_j; \theta)}{\partial \theta} \\
=&\sum\nolimits_j \left(\frac{\partial \log Z(\mathbf{x}_j, \mathcal{S}_j; \theta)}{\partial \theta} - \frac{\partial \log Z(\mathbf{x}_j, \tilde{\mathcal{S}}_j; \theta)}{\partial \theta} \right) \\
=&\sum\nolimits_j \left(E_{\mathbf{t}|\mathbf{x}_j, \mathcal{S}_j; \theta}[\mathbf{f}(\mathbf{x}_j, \mathbf{t})] - E_{\mathbf{t}|\mathbf{x}_j, \tilde{\mathcal{S}}_j; \theta}[\mathbf{f}(\mathbf{x}_j, \mathbf{t})] \right)
\end{aligned} \quad (4)$$

where the two terms are the feature expectations under $\mathcal{S}_j$ and $\tilde{\mathcal{S}}_j$ respectively. And the detailed derivations are as follows:

$$\begin{aligned}
&\frac{\partial \log Z(\mathbf{x}, \mathcal{S}; \theta)}{\partial \theta} \\
=&\frac{1}{Z(\mathbf{x}, \mathcal{S}; \theta)} \times \frac{\partial \sum_{\mathbf{t} \in \mathcal{S}} e^{Score(\mathbf{x}, \mathbf{t}; \theta)}}{\partial \theta} \\
=&\sum_{\mathbf{t} \in \mathcal{S}} \left(\frac{e^{Score(\mathbf{x}, \mathbf{t}; \theta)}}{Z(\mathbf{x}, \mathcal{S}; \theta)} \times \frac{\partial Score(\mathbf{x}, \mathbf{t}; \theta)}{\partial \theta} \right) \\
=&\sum_{\mathbf{t} \in \mathcal{S}} p(\mathbf{t}|\mathbf{x}, \mathcal{S}; \theta) \times \mathbf{f}(\mathbf{x}, \mathbf{t}) \\
=&E_{\mathbf{t}|\mathbf{x}, \mathcal{S}; \theta}[\mathbf{f}(\mathbf{x}, \mathbf{t})]
\end{aligned} \quad (5)$$

Please notice that $\mathbf{t} = [\mathbf{t}^a, \mathbf{t}^b]$ denotes a bundled tag sequence in this context of coupled sequence labeling.

2.3 Efficiency Issue

Under *complete mapping*, each one-side tag is mapped to all the-other-side tags for constructing bundled tags, producing a very huge set of legal bundled tags $\tilde{\mathcal{T}}_i = \mathcal{T}^a \times \mathcal{T}^b$. Using the classic Forward-Backward algorithm, we still need $O(n \times |\mathcal{T}^a|^2 \times |\mathcal{T}^b|^2)$ time complexity to compute $E_{\mathbf{t}|\mathbf{x}, \tilde{\mathcal{S}}; \theta}[\mathbf{f}(\mathbf{x}, \mathbf{t})]$, which is prohibitively expensive.[2]

In order to improve efficiency, Li et al. (2015) propose to use a set of context-free tag-to-tag mapping rules for reducing the search space. For example, we may specify that the CTB POS tag "NN" can only be concatenated with a set of PD tags like "{n, vn, ns}".[3] With much effort, they propose a set of *relaxed mapping* rules that greatly reduces the number of bundled tags from $|\mathcal{T}^a| \times |\mathcal{T}^b| = 33 \times 38 = 1,254$ to 179 for POS tagging.

3 Context-aware Pruning

Using manually designed context-free tag-to-tag mapping rules to constrain the search space has two major drawbacks. On the one hand, for more complex problems such as joint WS&POS tagging, it becomes very difficult to design proper mapping rules due to the much larger tag set. On the other hand, the experimental results in Li et al. (2015)

[2] In contrast, computing $E_{\mathbf{t}|\mathbf{x}, \mathcal{S}; \theta}[\mathbf{f}(\mathbf{x}, \mathbf{t})]$ is not the bottleneck, since $|\mathcal{T}_i| = |\mathcal{T}^b|$ for CTB or $|\mathcal{T}_i| = |\mathcal{T}^a|$ for PD.

[3] Please refer to http://hlt.suda.edu.cn/~zhli/resources/pos-mapping-CTB-PD.html for their detailed mapping rules.

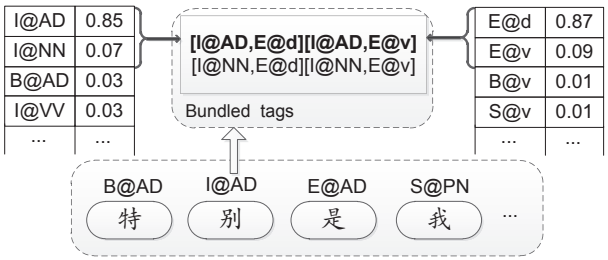


Figure 1: Illustration of context-aware pruning with $r = 2$ on a CTB training sentence.

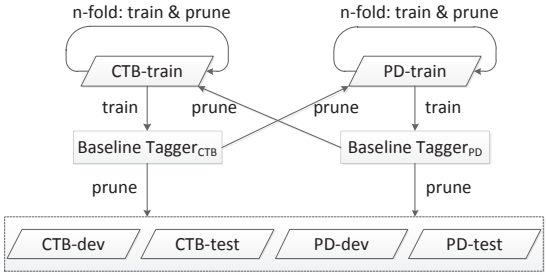


Figure 2: Workflow of offline pruning.

suggest that the coupled model can best learn the implicit context-sensitive mapping relationships between annotations under complete mapping, and imposing strict tag-to-tag mapping constraints usually hurts tagging accuracy.

In this work, our intuition is that the mapping relationships between heterogeneous annotations are highly context-sensitive. Therefore, we propose a context-aware pruning approach to more accurately capture such mappings, thus solving the efficiency issue. The basic idea is to consider only a small set of most likely bundled tags, instead of the whole bundled tag space $\mathcal{T}^a \times \mathcal{T}^b$, based on evidences of surrounding contexts. Specifically, for each token w_i, we only keep r one-side tags according to separate features $\mathbf{f}_{sep\_a/b}(.)$ for each side, and then use the remaining single-side tags to construct $\tilde{\mathcal{T}}_i$ and $\mathcal{T}_i$.

We use the second character "别/I@AD" in Fig. 1 as an example. We list the single-side tags in the descending order of their marginal probabilities according to $\mathbf{f}_{sep\_a/b}(.)$. Then we only keep $r = 2$ single-side tags, used as $\mathcal{T}_i^a$ and $\mathcal{T}_i^b$. Then $\tilde{\mathcal{T}}_i = \mathcal{T}^a \times \mathcal{T}^b$ contains the four bundled tags shown in the upper box, known as the whole possible tag set for searching. And $\mathcal{T}_i = \{\check{t}^a\} \times \mathcal{T}^b$ contains two bundled tags, as marked in bold, knowns as the possibly-correct tag set, since $\check{t}^a$ is the manually labeled tag. The case when the word has the second-side manually-labeled tag $\{\check{t}^b\}$ can be similarly handled.

Beside r, we use another hyper-parameter λ to further reduce the number of one-side tag candidates. The intuition is that in many cases, we may only need to use a smaller number $r' < r$ of possible candidates, since the remaining tags are very unlikely ones according to the marginal probabilities. Therefore, for each item w_i, we define r' as the smallest number of most likely candidate tags whose accumulative probability is larger than λ. Then, we only keep the $\min(r', r)$ most likely candidate tags.

We have $|\tilde{\mathcal{T}}_i| = r^2$ without considering the accumulated probability threshold λ. Thus, it requires $O(nr^4)$ time complexity to compute $E_{\mathbf{t}|\mathbf{x},\tilde{\mathcal{S}};\theta}[\mathbf{f}(\mathbf{x}, \mathbf{t})]$ using the Forward-Backward algorithm.

In the following, we propose two ways for realizing context-aware pruning, i.e., online and offline pruning. Their comparison and analysis are given in the experiment parts.

3.1 Online Pruning

The online pruning approach directly uses the coupled model to perform pruning. Given a sentence, we first use a subset of features $\mathbf{f}_{sep\_a}(.)$ and corresponding feature weights trained so far to compute marginal probabilities of first-side tags, and then analogously process the second-side tags based on $\mathbf{f}_{sep\_b}(.)$. This requires roughly the same time complexity as two baseline models. Then the marginal probabilities are used for pruning.

3.2 Offline Pruning

The offline pruning approach is a little bit more complex, and uses many additional single-side tagging models for pruning. Fig. 2 shows the workflow. Particularly, n-fold jack-knifing is adopted to perform pruning on the same-side training data. Finally, all training/dev/test datasets of CTB and PD are preprocessed in an offline way, so that each word in a sentence has a set of most likely CTB tags ($\mathcal{T}_i^a$) and another set of most likely PD tags ($\mathcal{T}_i^b$).

4 Experiment Settings

Data. Following Li et al. (2015), we use CTB5 and PD for the heterogeneous data. Under the standard

data split of CTB5, the training/dev/test datasets contain $16,091/803/1,910$ sentences respectively. For PD, we use the $46,815$ sentences in January 1998 as the training data, the first $2,000$ sentences in February as the development data, and the first $5,000$ sentences in June as the test data.

Evaluation Metrics. We use the standard token-wise tagging accuracy for POS tagging. For joint WS&POS tagging, besides character-wise tagging accuracy, we also use the standard precision (P), recall (R), and F-score of only words (WS) or POS-tagged words (WS&POS).

Parameter settings. Stochastic gradient descent (SGD) is adopted to train the baseline single-side tagging models, the guide-feature models, and the coupled models.[4]

For the coupled models, we directly follow the simple corpus-weighting strategy proposed in Li et al. (2015) to balance the contribution of the two datasets. We randomly sample $5,000$ CTB-train sentences and $5,000$ PD-train sentences, which are then merged and shuffled for one-iteration training. After each iteration, the coupled model is evaluated on both CTB-dev and PD-dev, providing us two single-side tag accuracies, one on CTB-side tags, and the other on PD-dev tags. Another advantage of using a subset of training data in one iteration is to monitor the training progress in smaller steps. For fair comparison, when building the baseline and guide-feature models, we also randomly sample $5,000$ training sentences from the whole training data for one-iteration training, and then report an tagging accuracy on development data. For all models, the training terminates if peak accuracies stop improving within 30 consecutive iterations, and we use the model that performs the best on development data for final evaluation on test data.

5 Experiments on POS Tagging

5.1 Parameter Tuning

For both online and offline pruning, we need to decide the maximum number of single-side tag candidates r and the accumulative probability threshold λ for further truncating the candidates. Table 2 shows the tagging accuracies and the averaged numbers of single-side tags for each token after pruning.

r	λ	Accuracy (%)		#Tags (pruned)	
		CTB5-dev	PD-dev	CTB-side	PD-side
Online Pruning					
2	0.98	94.25	95.03	2.0	2.0
4	0.98	95.06	95.66	3.9	4.0
8	**0.98**	**95.14**	**95.83**	6.3	7.4
16	0.98	95.12	95.81	7.8	14.1
8	0.90	**95.15**	95.79	3.7	6.3
8	0.95	95.13	**95.82**	5.1	7.1
8	0.99	**95.15**	95.74	7.4	7.9
8	1.00	**95.15**	95.76	8.0	8.0
Offline Pruning					
8	0.9999	94.95	96.05	4.1	5.1
16	**0.9999**	**95.15**	**96.09**	5.2	7.6
32	0.9999	95.13	96.09	5.5	9.3
16	0.99	94.42	95.77	1.6	2.2
16	0.999	95.02	**96.10**	2.6	4.0
16	0.99999	**95.10**	96.09	6.8	8.9

Table 2: POS tagging performance of online and offline pruning with different r and λ on CTB5 and PD.

The first major row tunes the two hyper-parameters for online pruning. We first fix $\lambda = 0.98$ and increase r from 2 to 8, leading to consistently improved accuracies on both CTB5-dev and PD-dev. No further improvement is gained with $r = 16$, indicating that tags below the top-8 are mostly very unlikely ones and thus insignificant for computing feature expectations. Then we fix $r = 8$ and try different λ. We find that λ has little effect on tagging accuracies but influences the numbers of remaining single-side tags. We choose $r = 8$ and $\lambda = 0.98$ for final evaluation.

The second major row tunes r and λ for offline pruning. Different from online pruning, λ has much greater effect on the number of remaining single-side tags. Under $\lambda = 0.9999$, increasing r from 8 to 16 leads to 0.20% accuracy improvement on CTB5-dev, but using $r = 32$ has no further gain. Then we fix $r = 16$ and vary λ from 0.99 to 0.99999. We choose $r = 16$ and $\lambda = 0.9999$ for offline pruning for final evaluation, which leaves each word with about 5.2 CTB-tags and 7.6 PD-tags on average.

[4] We use the implementation of SGD in CRFsuite (`http://www.chokkan.org/software/crfsuite/`), and set $b = 30$ as the batch-size and $C = 0.1$ as the regularization factor.

	Accuracy (%)		Speed
	CTB5-test	PD-test	Toks/Sec
Coupled (Offline)	**94.83**	95.90	246
Coupled (Online)	94.74	**95.95**	365
Coupled (No Prune)	94.58	95.79	3
Coupled (Relaxed)	94.63	95.87	127
Guide-feature	94.35	95.63	584
Baseline	94.07	95.82	1573
Li et al. (2012b)	94.60	—	—

Table 3: POS tagging performance of difference approaches on CTB5 and PD.

5.2 Main Results

Table 3 summarizes the accuracies on the test data and the tagging speed during the test phase. "Coupled (No Prune)" refers to the coupled model with complete mapping in Li et al. (2015), which maps each one-side tag to all the-other-side tags. "Coupled (Relaxed)" refers the coupled model with relaxed mapping in Li et al. (2015), which maps a one-side tag to a manually-designed small set of the-other-side tags. Li et al. (2012b) report the state-of-the-art accuracy on this CTB data, with a joint model of Chinese POS tagging and dependency parsing.

It is clear that both online and offline pruning greatly improve the efficiency of the coupled model by about two magnitudes, without the need of a carefully predefined set of tag-to-tag mapping rules.[5] Moreover, the coupled model with offline pruning achieves 0.76% accuracy improvement on CTB5-test over the baseline model, and 0.48% over our reimplemented guide-feature approach of Jiang et al. (2009). The gains on PD-test are marginal, possibly due to the large size of PD-train, similar to the results in Li et al. (2015).

6 Experiments on Joint WS&POS Tagging

6.1 Parameter Tuning

Table 4 shows results for tuning r and λ. From the results in the first major row, we can see that in the online pruning method, λ seems useless and r becomes the only threshold for pruning unlikely single-side tags. The accuracies are much inferior to those from the offline pruning approach. We believe that the accuracies can be further improved with larger r, which would nevertheless lead to severe inefficiency issue. Based on the results, we choose $r = 16$ and $\lambda = 1.00$ for final evaluation.

r	λ	Accuracy (%)		#Tags (pruned)	
		CTB5-dev	PD-dev	CTB-side	PD-side
Online Pruning					
8	1.00	90.41	89.91	8.0	8.0
16	0.95	90.65	90.22	15.9	16.0
16	0.99	90.77	**90.49**	16.0	16.0
16	**1.00**	**90.79**	**90.49**	16.0	16.0
Offline Pruning					
8	0.995	91.22	91.62	2.6	3.1
16	**0.995**	91.66	91.85	3.2	4.3
32	0.995	**91.67**	**91.87**	3.5	5.6
16	0.95	90.69	91.30	1.6	2.1
16	0.99	**91.64**	**91.92**	2.5	3.5
16	0.999	91.62	91.75	5.1	6.4

Table 4: WS&POS tagging performance of online and offline pruning with different r and λ on CTB5 and PD.

The second major row tries to decide r and λ for the offline pruning approach. Under $\lambda = 0.995$, increasing r from 8 to 16 improves accuracies both on CTB5-dev and PD-dev, but further using $r = 32$ leads to little gain. Then we fix $r = 16$ and vary λ from 0.95 to 0.999. Using $\lambda = 0.95$ leaves only 1.6 CTB tags and 2.1 PD tags for each character, but has a large accuracy drop. We choose $r = 16$ and $\lambda = 0.995$ for offline pruning for final evaluation, which leaves each character with 3.2 CTB-tags and 4.3 PD-tags on average.

6.2 Main Results

Table 5 summarizes the accuracies on the test data and the tagging speed (characters per second) during the test phase. "Coupled (No Prune)" is not tried due to the prohibitive tag set size in joint WS&POS tagging, and "Coupled (Relaxed)" is also skipped since it seems impossible to manually design reasonable tag-to-tag mapping rules in this case.

In terms of efficiency, the coupled model with offline pruning is on par with the baseline single-side tagging model.[6]

[5] Due to the model complexity of "Coupled (No Prune)", we discard all low-frequency (< 3) features in the training data to speed up training. This explains why "Coupled (No Prune)" has slightly lower accuracies than "Coupled (Relaxed)".

[6] The time estimation does not include the two separate processes of pruning single-side tags, which is approximately

	P/R/F (%) on CTB5-test		P/R/F (%) on PD-test		Speed
	Only WS	Joint WS&POS	Only WS	Joint WS&POS	Char/Sec
Coupled (Offline)	**95.65/95.46/95.55**	**90.68/90.49/90.58**	96.39/95.86/96.12	92.70/92.19/92.44	115
Coupled (Online)	95.17/94.71/94.94	89.80/89.37/89.58	95.76/95.45/95.60	91.71/91.41/91.56	26
Guide-feature	95.26/94.89/95.07	89.96/89.61/89.79	95.99/95.33/95.66	91.92/91.30/91.61	27
Baseline	95.00/94.77/94.88	89.60/89.38/89.49	**96.56/96.00/96.28**	**92.74/92.20/92.47**	119

Table 5: WS&POS tagging performance of difference approaches on CTB5 and PD.

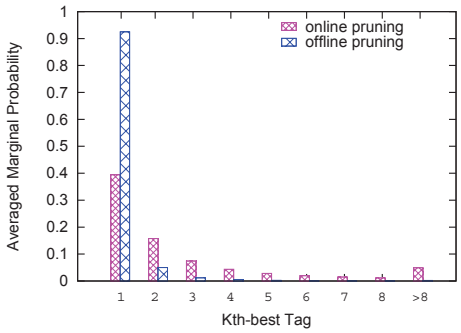


Figure 3: Probability distribution with online/offline pruning for the task of joint WS&POS.

In terms of F-score, the coupled model with offline pruning achieves 0.67% (WS) and 1.09% (WS&POS) gains on CTB5-test over the baseline model, and 0.48% (WS) and 0.79% (WS&POS) over our reimplemented guide-feature approach of Jiang et al. (2009). Similar to the case of POS tagging, the baseline model is very competitive on PD-test due to the large scale of PD-train.

6.3 Analysis

Online vs. offline pruning. The averaged numbers of single-side tags after pruning in Table 4 and 2), suggest that the online pruning approach works badly in assigning proper marginal probabilities to different tags. Our first guess is that in online pruning, the weights of separate features are optimized as a part of the coupled model, and thus producing somewhat flawed probabilities. However, our further analysis gives a more convincing explanation.

Fig. 3 compares the distribution of averaged probabilities of k^{th}-best CTB-side tags after online and offline pruning. The statistics are gathered on CTB5-test. Under online pruning, the averaged probability of the best tag is only about 0.4, which is surprisingly low and cannot be explained with the aforementioned improper optimization issue. Please note that both the online and offline models uses the best choices of r and λ based on Table 4, and are trained until convergence.

After a few trials of reducing the size of PD-train for training the coupled model, we realize that the underlying reason is that ambiguous labeling makes the probability mass more uniformly distributed, since for a PD-train sentence, the characters only have the gold-standard PD-side tags, and the model basically uses all CTB-side tags as gold-standard answers. Thanks to the CTB-train sentences, the model may be able to choose the correct tag, but inevitably becomes more indecisive at the same time due to the PD-train sentences.

In contrast, the offline pruning approach directly uses two baseline models for pruning, which is a job perfectly suitable for the baseline models. The entropy of the probability distribution for online pruning is about 1.524 while that for offline pruning is only 0.355.

Error distributions. To better understand the gains from the coupled approach, we show the F-score of specific POS tags for both the baseline and coupled models in Fig. 4, in the descending order of absolute F-score improvements. The largest improvement is from words tagged as "LB" (mostly for the word "被", marking a certain type of passive construction), and the F-score increases by $65.22 - 54.55 = 10.67\%$. Nearly all POS tags have more or less F-score improvement. Due to the space limit, we only show the tags with more than 2.0% improvement. The most noticeable exception is that F-score drops by $84.80 - 86.49 = -1.69\%$ for words tagged as "OD" (ordinal numbers, as opposed to cardinal numbers).

In terms of words, we find the largest gain is from "卢森博格/NR" (Luxemburgo, place name), which appears 11 times in CTB5-test, with an absolute

equal to the time of two baseline models.

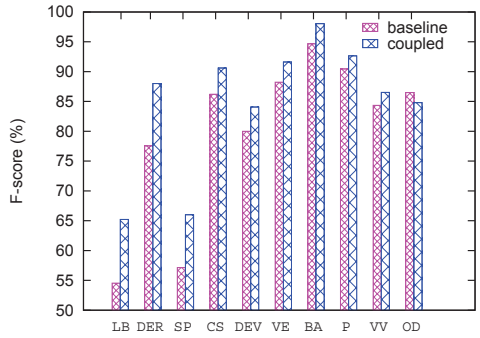


Figure 4: F-score comparison between the baseline and coupled WS&POS tagging models on different CTB POS tags.

	F (%) on CTB5X-test	
	Only WS	Joint WS&POS
Coupled (Offline)	**98.01**	**94.39**
Guide-feature	97.96	94.06
Baseline	97.37	93.23
Sun and Wan (2012)	—	94.36
Jiang et al. (2009)	98.23	94.03

Table 6: WS&POS tagging performance of difference approaches on CTB5X and PD.

improvement of $90.00 - 16.67 = 73.33\%$ in recall ratio. The reason is that PD-train contains a lot of related words such as "卢森堡" (Luxembourg, place name) and "克拉泽博格" (Krayzelburg, person name) while CTB5-train has none.

6.4 Comparison with Previous Work

In order to compare with previous work, we also run our models on CTB5X and PD, where CTB5X adopts a different data split of CTB5 and is widely used in previous research on joint WS&POS tagging (Jiang et al., 2009; Sun and Wan, 2012). CTB5X-dev/test only contain $352/348$ sentences respectively. Table 6 presents the F scores on CTB5X-test. We can see that the coupled model with offline pruning achieves 0.64% (WS) and 1.16% (WS&POS) F-score improvements over the baseline model, and 0.05% (WS) and 0.33% (WS&POS) over the guide-feature approach.

The original guide-feature method in Jiang et al. (2009) achieves 98.23% and 94.03% F-score, which is very close to the results of our reimplemented model. The sub-word stacking approach of Sun and Wan (2012) can be understood as a more complex variant of the basic guide-feature method.[7]

The results on both the larger CTB5-test (in Table 5) and CTB5X-test suggest that the coupled approach is more consistent and robust than the guide-feature method. The reason may be two-fold. First, in the coupled approach, the model is able to actively learn the implicit mappings between two sets of annotations, whereas the guide-feature model can only passively learn when to trust the automatically produced tags. Second, the coupled approach can directly learn from both heterogeneous training datasets, thus covering more phenomena of language usage.

7 Related Work

A lot of research has been devoted to design an effective way to exploit non-overlapping heterogeneous labeled data, especially in Chinese language processing, where such heterogeneous resources are ubiquitous due to historical reasons. Jiang et al. (2009) first propose the guide-feature approach, which is similar to stacked learning (Nivre and McDonald, 2008), for joint WS&POS tagging on CTB and PD. Sun and Wan (2012) further extend the guide-feature method and propose a more complex sub-word stacking approach. Qiu et al. (2013) propose a linear coupled model similar to that of Li et al. (2015). The key difference is that the model of Qiu et al. (2013) only uses separate features, while Li et al. (2015) and this work explore joint features as well.

Li et al. (2012a) apply the guide-feature idea to dependency parsing on CTB and PD. Zhang et al. (2014a) extend a shift-reduce dependency parsing model in order to simultaneously learn and produce two heterogeneous parse trees, which however assumes the existence of training data with both-side annotations.

Our context-aware pruning approach is similar to coarse-to-fine pruning in parsing community (Koo and Collins, 2010; Rush and Petrov, 2012), which is a useful technique that allows us to use very complex parsing models without too much efficiency cost. The idea is first to use a simple and basic off-shelf model to prune the search space and only keep highly likely dependency links, and then let the complex

[7] Sun and Wan (2012) achieve 94.68% F-score on CTB5X-test by further employing a re-training strategy.

model infer in the remaining search space. Weiss and Taskar (2010) propose structured prediction cascades: a sequence of increasingly complex models that progressively filter the space of possible outputs, and provide theoretical generalization bounds on a novel convex loss function that balances pruning error with pruning efficiency.

This work is also closely related with multi-task learning, which aims to jointly learn multiple related tasks with the benefit of using interactive features under a share representation (Ben-David and Schuller, 2003; Ando and Zhang, 2005; Parameswaran and Weinberger, 2010). However, as far as we know, multi-task learning usually assumes the existence of data with labels for multiple tasks at the same time, which is unavailable in our scenario, making our problem more particularly difficult.

Our coupled CRF model is similar to a factorial CRF (Sutton et al., 2004), in the sense that the bundled tags can be factorized into two connected latent variables. Initially, factorial CRFs are designed to jointly model two related (and typically hierarchical) sequential labeling tasks, such as POS tagging and chunking. In this work, our coupled CRF model jointly handles two same tasks with different annotation schemes. Moreover, this work provides a natural way to learn from incomplete annotations where one sentence only contains one-side labels.

Learning with ambiguous labeling is previously explored for classification (Jin and Ghahramani, 2002), sequence labeling (Dredze et al., 2009), parsing (Riezler et al., 2002; Täckström et al., 2013). Recently, researchers propose to derive natural annotations from web data to supervise Chinese word segmentation models in the form of ambiguous labeling (Jiang et al., 2013; Liu et al., 2014; Yang and Vozila, 2014).

8 Conclusion

This paper proposes a context-aware pruning approach for the coupled sequence labeling model of Li et al. (2015). The basic idea is to more accurately constrain the bundled tag space of a token according to its contexts in the sentence, instead of using heuristic context-free tag-to-tag mapping rules in the original work. We propose and compare two different ways of realizing pruning, i.e., online and offline pruning. In summary, extensive experiments leads to the following findings.

(1) Offline pruning works well on both POS tagging and joint WS&POS tagging, whereas online pruning only works well on POS tagging but fails on joint WS&POS tagging due to the much larger tag set. Further analysis shows that the reason is that under online pruning, ambiguous labeling during training makes the probabilities of single-side tags more evenly distributed.

(2) In terms of tagging accuracy and F-score, the coupled approach with offline pruning outperforms the baseline single-side tagging model by large margin, and is also consistently better than the mainstream guide-feature method on both POS tagging and joint WS&POS tagging.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We are very grateful to Meishan Zhang for inspiring us to use online pruning to improve the efficiency of the coupled approach. We also thank Wenliang Chen for the helpful discussions. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61502325, 61432013).

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learn Research*, 6:1817–1853.

Shai Ben-David and Reba Schuller. 2003. Exploiting task relatedness for multiple task learning. In *COLT*.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *ECML/PKDD Workshop on Learning from Multi-Label Data*.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging – a case study. In *Proceedings of ACL*, pages 522–530.

Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of ACL*, pages 761–769.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Proceedings of NIPS*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*, pages 1–11.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

Zhenghua Li, Wanxiang Che, and Ting Liu. 2012a. Exploiting multiple treebanks for parsing with quasi-synchronous grammar. In *ACL*, pages 675–684.

Zhenghua Li, Min Zhang, Wanxiang Che, and Ting Liu. 2012b. A separately passive-aggressive training algorithm for joint POS tagging and dependency parsing. In *COLING*, pages 1681–1698.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled sequence labeling on heterogeneous annotations: POS tagging as a case study. In *Proceedings of ACL*, pages 1783–1792.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of EMNLP*, pages 864–874.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

S. Parameswaran and K.Q. Weinberger. 2010. Large margin multi-task metric learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1867–1875.

Xipeng Qiu, Jiayi Zhao, and Xuanjing Huang. 2013. Joint Chinese word segmentation and POS tagging on heterogeneous annotated corpora with multiple task learning. In *Proceedings of EMNLP*, pages 658–668.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, and Mark Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of ACL*, pages 271–278.

Alexander Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL-2012*, pages 498–507.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for Chinese lexical processing with heterogeneous annotations. In *Proceedings of ACL*, pages 232–241.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *International Conference on Machine Learning (ICML)*.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL*, pages 1061–1071.

David Weiss and Ben Taskar. 2010. Structured prediction cascades. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Fei Xia. 2000. The part-of-speech tagging guidelines for the penn Chinese treebank 3.0. In *Technical Report, Linguistic Data Consortium*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

Fan Yang and Paul Vozila. 2014. Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of EMNLP*, pages 90–98.

Shiwen Yu, Huiming Duan, Xuefeng Zhu, Bin Swen, and Baobao Chang. 2003. Specification for corpus processing at Peking University: Word segmentation, POS tagging and phonetic notation (In Chinese). *Journal of Chinese Language and Computing*, 13(2):121–158.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896.

Meishan Zhang, Wanxiang Che, Yanqiu Shao, and Ting Liu. 2014a. Jointly or separately: Which is better for parsing heterogeneous dependencies? In *Proceedings of COLING*, pages 530–540.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014b. Character-level Chinese dependency parsing. In *Proceedings of ACL*, pages 1326–1336.

Unsupervised Neural Dependency Parsing*

Yong Jiang, Wenjuan Han and **Kewei Tu**
{jiangyong,hanwj,tukw}@shanghaitech.edu.cn
School of Information Science and Technology
ShanghaiTech University, Shanghai, China

Abstract

Unsupervised dependency parsing aims to learn a dependency grammar from text annotated with only POS tags. Various features and inductive biases are often used to incorporate prior knowledge into learning. One useful type of prior information is that there exist correlations between the parameters of grammar rules involving different POS tags. Previous work employed manually designed features or special prior distributions to encode such information. In this paper, we propose a novel approach to unsupervised dependency parsing that uses a neural model to predict grammar rule probabilities based on distributed representation of POS tags. The distributed representation is automatically learned from data and captures the correlations between POS tags. Our experiments show that our approach outperforms previous approaches utilizing POS correlations and is competitive with recent state-of-the-art approaches on nine different languages.

1 Introduction

Unsupervised structured prediction from data is an important problem in natural language processing, with applications in grammar induction, POS tag induction, word alignment and so on. Because the training data is unannotated in unsupervised structured prediction, learning is very hard. In this paper, we focus on unsupervised dependency parsing, which aims to identify the dependency trees of sentences in an unsupervised manner.

*This work was supported by the National Natural Science Foundation of China (61503248).

Previous work on unsupervised dependency parsing is mainly based on the dependency model with valence (DMV) (Klein and Manning, 2004) and its extension (Headden III et al., 2009; Gillenwater et al., 2010). To effectively learn the DMV model for better parsing accuracy, a variety of inductive biases and handcrafted features have been proposed to incorporate prior information into learning. One useful type of prior information is that there exist correlations between the parameters of grammar rules involving different POS tags. Cohen and Smith (2009; 2010) employed special prior distributions to encourage learning of correlations between POS tags. Berg-Kirkpatrick et al. (2010) encoded the relations between POS tags using manually designed features.

In this work, we propose a neural based approach to unsupervised dependency parsing. We incorporate a neural model into the DMV model to predict grammar rule probabilities based on distributed representation of POS tags. We learn the neural network parameters as well as the distributed representations from data using the expectation-maximization algorithm. The correlations between POS tags are automatically captured in the learned POS embeddings and contribute to the improvement of parsing accuracy. In particular, probabilities of grammar rules involving correlated POS tags are automatically smoothed in our approach without the need for manual features or additional smoothing procedures.

Our experiments show that on the Wall Street Journal corpus our approach outperforms the previous approaches that also utilize POS tag correla-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 763–771,
Austin, Texas, November 1-5, 2016.

tions, and achieves a comparable result with recent state-of-the-art grammar induction systems. On the datasets of eight additional languages, our approach is able to achieve better performance than the baseline methods without any parameter tuning.

2 Related work

2.1 Dependency Model with Valence

The dependency model with valence (DMV) (Klein and Manning, 2004) is the first model to outperform the left-branching baseline in unsupervised dependency parsing of English. The DMV model is a generative model of a sentence and its parse tree. It generates a dependency parse from the root in a recursive top-down manner. At each step, a decision is first made as to whether a new child POS tag shall be generated from the current head tag; if the decision is yes, then a new child POS tag is sampled; otherwise, the existing child tags are recursively visited. There are three types of grammar rules in the model: `CHILD, DECISION` and `ROOT`, each with a set of multinomial parameters $\mathrm{P}_{CHILD}(c|h, dir, val)$, $\mathrm{P}_{DECISION}(dec|h, dir, val)$ and $\mathrm{P}_{ROOT}(c|root)$, where dir is a binary variable indicating the generation direction (left or right), val is a boolean variable indicating whether the current head POS tag already has a child in the current direction or not, c indicates the child POS tag, h indicates the head POS tag, and dec indicates the decision of either STOP or CONTINUE. A `CHILD` rule indicates the probability of generating child c given head h on direction dir and valence val. A `DECISION` rule indicates the probability of STOP or CONTINUE given the head, direction and valence. A `ROOT` rule is the probability of a child c generated by the root. The probability of a dependency tree is the product of probabilities of all the grammar rules used in generating the dependency tree. The probability of a sentence is the sum of probabilities of all the dependency trees consistent with the sentence.

The basic DMV model has the limitation of being oversimplified and unable to capture certain linguistic structures. Headden et al. (2009) incorporated more types of valence and lexicalized information in the DMV model to increase its representation power and achieved better parsing accuracy than the basic DMV model.

2.2 DMV-based Learning Algorithms for Unsupervised Dependency Parsing

To learn a DMV model from text, the Expectation Maximization (EM) algorithm (Klein and Manning, 2004) can be used. In the E step, the model calculates the expected number of times each grammar rule is used in parsing the training text by using the inside-outside algorithm. In the M-step, these expected counts are normalized to become the probabilities of the grammar rules.

There have been many more advanced learning algorithms of the DMV model beyond the basic EM algorithm. In the work of Cohen and Smith (2008), a logistic normal prior was used in the DMV model to capture the similarity between POS tags. In the work of Berg-Kirkpatrick et al. (2010), features that group various morphological variants of nouns and verbs are used to predict the `DECISION` and `CHILD` parameters. These two approaches both utilize the correlations between POS tags to obtain better probability estimation of grammar rules involving such correlated POS tags. In the work of Tu and Honavar (2012), unambiguity of parse trees is incorporated into the training objective function of DMV to obtain a better performance.

2.3 Other Approaches to Unsupervised Dependency Parsing

There are many other approaches to unsupervised dependency parsing that are not based on DMV. Daumé III (2009) proposed a stochastic search based method to do unsupervised Shift-Reduce transition parsing. Rasooli and Faili (2012) proposed a transition based unsupervised dependency parser together with "baby-step" training (Spitkovsky et al., 2010) to improve parsing accuracy. Le and Zuidema (2015) proposed a complicated reranking based unsupervised dependency parsing system and achieved the state-of-the-art performance on the Penn Treebank dataset.

2.4 Neural based Supervised Dependency Parser

There exist several previous approaches on using neural networks for supervised dependency parsing. Garg and Henderson (2011) proposed a Temporal Restricted Boltzmann Machine to do transition

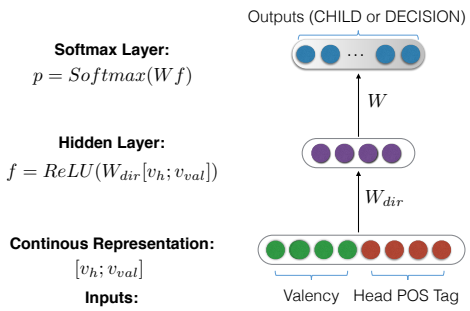


Figure 1: Structure of the neural network. Both CHILD and DECISION use the same architecture for the calculation of distributions.

based dependency parsing. Stenetorp (2013) applied recursive neural networks to transitional based dependency parsing. Chen and Manning (2014) built a neural network based parser with dense features instead of sparse indicator features. Dyer et al. (2015) proposed a stack long short-term memory approach to supervised dependency parsing. To our knowledge, our work is the first attempt to incorporate neural networks into a generative grammar for unsupervised dependency parsing.

3 Neural DMV

In this section, we introduce our neural based grammar induction approach. We describe the model in section 3.1 and the learning method in section 3.2.

3.1 Model

Our model is based on the DMV model (section 2.1), except that the CHILD and DECISION probabilities are calculated through two neural networks. We do not compute the ROOT probabilities using a neural network because doing that complicates the model while leads to no significant improvement in the parsing accuracy. Parsing a sentence using our model can be done in the same way as using DMV.

Below we show how the CHILD rule probabilities are computed in our neural based DMV model. Denote the set of all possible POS tags by T. We build a neural network to compute the probabilities of producing child tag $c \in T$ conditioned on the head, direction and valence (h, dir, val).

The full architecture of the neural network is shown in Figure 1. First, we represent each head tag h as a d dimensional vector $v_h \in \mathbb{R}^d$, represent each value of valence val as a d' dimensional vector $v_{val} \in \mathbb{R}^{d'}$. We concatenate v_h and v_{val} as the input embedding vector. Then we map the input layer to a hidden layer with weight matrix W_{dir} through a ReLU activation function. We have two versions of weight matrix W_{dir} for the direction dir being left and right respectively.

$$f(h, dir, val) = \mathrm{ReLU}(W_{dir}[v_h; v_{val}])$$

We then take the inner product of f and all the child POS tag vectors and apply a softmax function to obtain the rule probabilities:

$$[p_{c_1}, p_{c_2}, ..., p_{c_{\|T\|}}] = \mathrm{Softmax}(W^{\mathrm{T}} f)$$

where $W = [v_{c_1}, v_{c_2}, ..., v_{c_{\|T\|}}]$ is an embedding matrix composed of all the child POS tag vectors.

We use the same neural architecture to predict the probabilities of DECISION rules. The difference is that the neural network for DECISION has only two outputs (STOP and CONTINUE). Note that the two networks share parameters such as head POS tag embeddings and direction weight matrices W_{left} and W_{right}. Valence embeddings are either shared or distinct between the two networks depending on the variant of DMV we use (i.e., whether the maximal valences for CHILD and DECISION are the same).

The parameters of our neural based model include the weights of the neural network and all the POS and valence embeddings, denoted by a set $\Theta = \{v_h, v_c, v_{val}, v_{dec}, W_{dir}; h, c \in T, val \in \{0, 1, ...\}, dir \in \{left, right\}, dec \in \{\mathrm{STOP}, \mathrm{CONTINUE}\}\}$.

3.2 Learning

In this section, we describe an approach based on the EM algorithm to learn our neural DMV model. To learn the parameters, given a set of unannotated sentences $x_1, x_2, ..., x_N$, our objective function is the log-likelihood function.

$$L(\Theta) = \sum_{\alpha=1}^{N} \log \mathrm{P}(x_\alpha; \Theta)$$

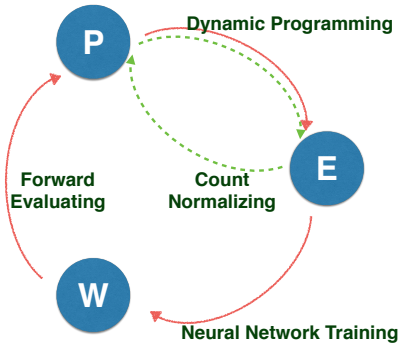


Figure 2: Learning procedure of our neural based DMV model. Green dashed lines represent the EM algorithm for learning traditional DMV. Red solid lines represent the learning procedure of our model. P represents the rule probabilities of DMV, E represents the expected counts of rules, and W represents the parameters of the neural networks. In the traditional EM algorithm, the expected counts are directly used to re-estimate the rule probabilities. In our approach, parameter re-estimation is divided into two steps: training the neural networks from the expected counts and forward evaluation of the neural networks to produce the rule probabilities.

The approach is visualized in the Figure 2. The E-step computes the expected number of times each grammar rule used in parsing each training sentence x_i, denoted by $e_c(x_i)$ for CHILD rule c, $e_d(x_i)$ for DECISION rule d, and $e_r(x_i)$ for ROOT rule r. In the M-step of traditional DMV learning, these expected counts are normalized to re-estimate the parameters of DMV. This maximizes the expected log likelihood (ELL) with respect to the DMV model parameters.

$$ELL(\Theta) = \sum_{\alpha=1}^{N} \left(\sum_{c} e_c(x_i) \log p_c + \sum_{d} e_d(x_i) \log p_d + \sum_{r} e_r(x_i) \log p_r \right)$$

In our model, however, we do not directly assign the optimal rule probabilities of CHILD and DECISION; instead, we train the neural networks to output rule probabilities that optimize ELL, which is equivalent to a weighted cross-entropy loss function for each neural network. Note that while the traditional M-step produces the global optimum of ELL, our neural-based M-step does not. This is because a neural network tends to produce similar outputs for correlated inputs. In our case, the neural network is able to capture the correlations between different POS tags as well as different valence values and smooth the probabilities involving correlated tags and valences. In other words, our M-step can be seen as optimizing the ELL with a regularization term taking into account the input correlations. We use momentum based batch stochastic gradient descent algorithm to train the neural network and learn all the embeddings and weight matrices.

In addition to standard EM, we can also learn our neural based DMV model based on the Viterbi EM algorithm. The difference from standard EM is that in the E-step, we compute the number of times each grammar rule is used in the best parse of a training sentence instead of considering all possible parses.

4 Experiments

4.1 Setup

We used the Wall Street Journal corpus (with section 2-21 for training, section 22 for validation and section 23 for testing) in section 4.2 and 4.3. Then we reported the results on eight additional languages in section 4.4. In each experiment, we trained our model on gold POS tags with sentences of length less than 10 after punctuation has been stripped off. As the EM algorithm is very sensitive to initializations, we used the informed initialization method proposed in (Klein and Manning, 2004).

The length of embeddings is set to 10 for both POS tags and valence. We trained the neural networks with batch size 10 and used the change of the validation set loss function as the stop criteria. We ran our model for five times and reported the averaged directed dependency accuracy (DDA) of the learned grammars on the test sentences with length less than 10 and all sentences.

4.2 Comparisons of Approaches based on POS Correlations

We first evaluated our approach in learning the basic DMV model and compared the results against (Cohen and Smith, 2009) and (Berg-Kirkpatrick et al., 2010), both of which have very similar motivation as ours in that they also utilize the correlation between POS tags to learn the basic DMV model. Table 1

Methods	WSJ10	WSJ
Standard EM	46.2	34.9
Viterbi EM	58.3	39.4
LN (Cohen et al., 2008)	59.4	40.5
Shared LN (Cohen and Smith, 2009)	61.3	41.4
Feature DMV (Berg-Kirkpatrick et al., 2010)	63.0	-
Neural DMV (Standard EM)	51.3	37.1
Neural DMV (Viterbi EM)	**65.9**	**47.0**

Table 1: Comparisons of Approaches based on POS Correlations

shows the results. It can be seen that our approach with Viterbi EM significantly outperforms the EM and viterbi EM baselines and also outperforms the two previous approaches.

4.3 Results on the extended DMV model

We directly apply our neural approach to learning the extended DMV model (Headden III et al., 2009; Gillenwater et al., 2010) (with the maximum valence value set to 2 for both CHILD and DECISION rules). As shown in Table 2, we achieve comparable accuracy with recent state-of-the-art systems. If we initialize our model with the grammar learned by Tu and Honavar (2012), the accuracy of our approach can be further improved.

Most of the recent state-of-the-art systems employ more complicated models and learning algorithms. For example, Spitkovsky et al. (2013) take several grammar induction techniques as modules and connect them in various ways; Le and Zuidema (2015) use a neural-based supervised parser and reranker that make use of high-order features and lexical information. We expect that the performance of our approach can be further improved when these more advanced techniques are incorporated.

4.4 Results on other languages

We also applied our approach on datasets of eight additional languages from the PASCAL Challenge on Grammar Induction (Gelling et al., 2012). We ran our approach using the hyper-parameters from experiment 4.2 on the new datasets without any further tuning. We tested three versions of our approach based on standard EM, softmax EM (Tu and Honavar, 2012) and Viterbi EM respectively. The results are shown in Table 3 for test sentence length no longer than ten and Table 4 for all test sentences.

Methods	WSJ10	WSJ
Systems with Basic Setup		
EVG (Headden III et al., 2009)	65.0	-
TSG-DMV (Blunsom and Cohn, 2010)	65.9	53.1
PR-S (Gillenwater et al., 2010)	64.3	53.3
UR-A E-DMV (Tu and Honavar, 2012)	71.4	57.0
Neural E-DMV	69.7	52.5
Neural E-DMV (Good Init)	**72.5**	**57.6**
Systems Using Extra Info		
LexTSG-DMV (Blunsom and Cohn, 2010)	67.7	55.7
L-EVG (Headden III et al., 2009)	68.8	-
CS (Spitkovsky et al., 2013)	72.0	64.4
MaxEnc (Le and Zuidema, 2015)	73.2	65.8

Table 2: Comparison of recent unsupervised dependency parsing systems. Basic setup means learning from POS tags with sentences of length $\leq$ 10 and punctuation stripped off. Extra information may contain punctuations, longer sentences, lexical information, etc. For Neural E-DMV, "Good Init" means using the learned DMV model from Tu and Honavar (2012) as our initialization.

Our neural based methods achieve better results than their corresponding baselines in 75.0% of the cases for test sentences no longer than 10 and 77.5% for all test sentences. The good performance of our approach without data-specific hyper-parameter tuning demonstrates the robustness of our approach. Carefully tuned hyper-parameters on validation datasets, in our experience, can further improve the performance of our approach, in some cases by a large margin.

4.5 Effects of Hyper-parameters

We examine the influence of hyper-parameters on the performance of our approach with the same experimental setup as in section 4.3.

Activation function We compare different linear and non-linear functions: ReLU, Leaky ReLU, Tanh, Sigmoid. The results are shown in Table 5. Non-linear activation functions can be seen to significantly outperform linear activation functions.

Length of the embedding vectors The dimension of the embedding space is an important hyper-parameter in our system. As Figure 3 illustrates, when the dimension is too low (such as $dim = 5$), the performance is bad probably because the embedding vectors cannot effectively discriminate between

	Arabic	Basque	Czech	Danish	Dutch	Portuguese	Slovene	Swedish
Standard EM								
DMV	**45.8**	41.1	31.3	50.8	47.1	**36.7**	36.7	43.5
Neural DMV	43.4	**46.5**	**33.1**	**55.6**	**49.0**	30.4	**42.2**	**44.3**
Softmax EM $\sigma = 0.25$								
DMV	49.3	45.6	30.4	43.6	**46.1**	33.5	29.8	**50.3**
Neural DMV	**54.2**	**46.3**	**36.8**	**44.0**	39.9	**35.8**	**31.2**	49.7
Softmax EM $\sigma = 0.5$								
DMV	**54.2**	47.6	**43.2**	38.8	**38.0**	33.7	23.0	37.2
Neural DMV	44.6	**48.9**	33.4	**50.3**	37.5	**35.3**	**32.2**	**43.3**
Softmax EM $\sigma = 0.75$								
DMV	42.2	**48.6**	22.7	41.0	**33.8**	33.5	**23.2**	41.6
Neural DMV	**56.7**	45.3	**31.6**	**41.3**	33.7	**34.7**	22.9	**42.0**
Viterbi EM								
DMV	32.5	47.1	27.1	39.1	37.1	32.3	23.7	42.6
Neural DMV	**48.2**	**48.1**	**28.6**	**39.8**	**37.2**	**36.5**	**39.9**	**47.9**

Table 3: DDA results (on sentences no longer than 10) on eight additional languages. Our neural based approaches are compared with traditional approaches using standard EM, softmax EM (parameterized by σ) and Viterbi EM.

Activation function	**WSJ10**
ReLU	**69.7**
Leaky ReLU	67.0
Tanh	66.2
Sigmoid	62.5
Linear	55.1

Table 5: Comparison between activation functions.

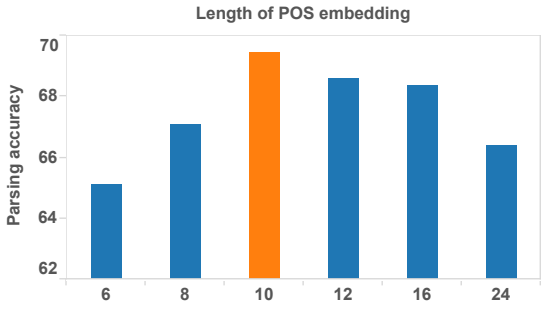


Figure 3: Parsing accuracy vs. length of POS embedding

different POS tags. On the other hand, when the dimension is too high (such as $dim = 30$), since we have only 35 POS tags, the neural network is prone to overfitting.

Shared parameters An alternative to our neural network architecture is to have two separate neural networks to compute `CHILD` and `DECISION` rule probabilities respectively. The embeddings of the head POS tag and the valence are not shared between the two networks. As can be seen in Table 6, sharing POS tags embeddings attribute to better performance.

	WSJ10	**WSJ**
Separate Networks	68.6	52.1
Merged Network	**69.7**	**52.5**

Table 6: Comparison between using two separate networks and using a merged network.

5 Model Analysis

In this section, we investigate what information our neural based DMV model captures and analyze how it contributes to better parsing performance.

5.1 Correlation of POS Tags Encoded in Embeddings

A main motivation of our approach is to encode correlation between POS tags in their embeddings so as to smooth the probabilities of grammar rules involving correlated POS tags. Here we want to examine whether the POS embeddings learned by our approach successfully capture such correlation.

We collected the POS embeddings learned in the experiment described in section 4.3 and visualized them on a 2D plane using the t-SNE algorithm (Van der Maaten and Hinton, 2008). t-SNE is a dimensionality reduction algorithm that maps data from a high dimensional space to a low dimensional one (2 or 3) while maintaining the distances between

	Arabic	Basque	Czech	Danish	Dutch	Portuguese	Slovene	Swedish
Standard EM								
DMV	28.0	31.2	28.1	40.3	44.2	**23.5**	25.2	32.0
Neural DMV	**30.6**	**38.5**	**29.3**	**46.1**	**46.2**	16.2	**36.6**	**32.8**
Softmax EM $\sigma = 0.25$								
DMV	30.0	38.1	27.1	35.1	**42.5**	**27.4**	23.1	**41.6**
Neural DMV	**31.5**	**40.5**	**32.6**	**38.0**	35.7	26.7	**24.2**	41.3
Softmax EM $\sigma = 0.5$								
DMV	**32.3**	41.0	**33.0**	32.2	33.9	27.6	15.0	29.6
Neural DMV	22.5	**42.6**	30.6	**40.8**	**37.5**	**28.6**	**25.0**	**33.7**
Softmax EM $\sigma = 0.75$								
DMV	30.1	**43.0**	15.6	33.9	**29.9**	25.8	**15.2**	32.7
Neural DMV	**34.9**	37.4	**24.7**	**34.2**	29.5	**28.9**	15.1	**33.3**
Viterbi EM								
DMV	23.9	40.9	20.4	32.6	33.0	26.9	16.5	36.2
Neural DMV	**31.0**	**41.8**	**23.8**	**34.2**	**33.6**	**29.4**	**30.8**	**40.2**

Table 4: DDA results (on all the sentences) on eight additional languages. Our neural based approaches are compared with traditional approaches using standard EM, softmax EM (parameterized by σ) and viterbi EM.

the data points in the high dimensional space. The "perplexity" hyper-parameter of the algorithm was set to 20.0 and the distance metric we used is the Euclidean distance.

Figure 4 shows the visualization result. It can be seen that in most cases, nearby POS tags in the figure are indeed similar. For example, VBP (Verb, non-3rd person singular present), VBD (Verb, past tense) and VBZ (Verb, 3rd person singular present) can be seen to be close to each other, and they indeed have very similar syntactic behavior. Similar observation can be made to NN (Noun, singular or mass), NNPS (Proper noun, plural) and NNS (Noun, plural).

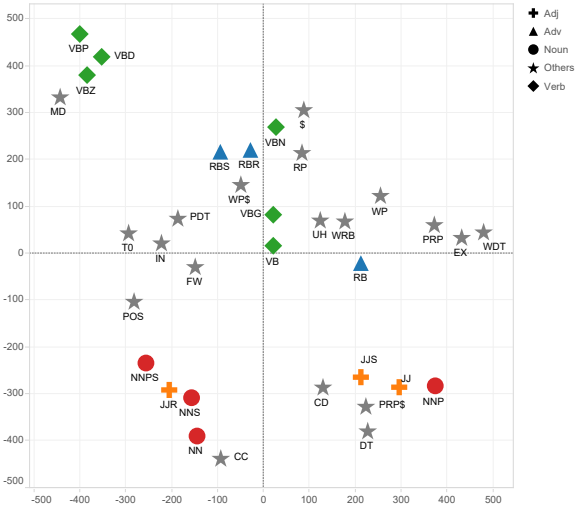


Figure 4: A visualization of the distances between embeddings of different POS tags.

5.2 Smoothing of Grammar Rule Probabilities

By using similar embeddings to represent correlated POS tags, we hope to smooth the probabilities of rules involving correlated POS tags. Here we analyze whether our neural networks indeed predict more similar probabilities for rules with correlated POS tags.

We conducted a case study on all types of verbs: VBP (Verb, non-3rd person singular present), VBZ (Verb, 3rd person singular present), VBD (Verb, past tense), VBN (Verb, past participle), VB (Verb, base form), VBG (Verb, gerund or present participle). We used the neural networks in our N-DMV model learned in the experiment described in section 4.2 to predict the probabilities of all the CHILD rules headed by a verb. For each pair of verb tags, we computed the total variation distance between the multinomial distributions of CHILD rules headed by the two verb tags. We also computed the total variation distances between CHILD rules of verb tags in the baseline DMV model learned by EM.

In Figure 5, We report the differences between the total variation distances computed from our model and from the baseline. A positive value means the distance is reduced in our model compared with that in the baseline. It can be seen that overall the distances between CHILD rules of different verb tags

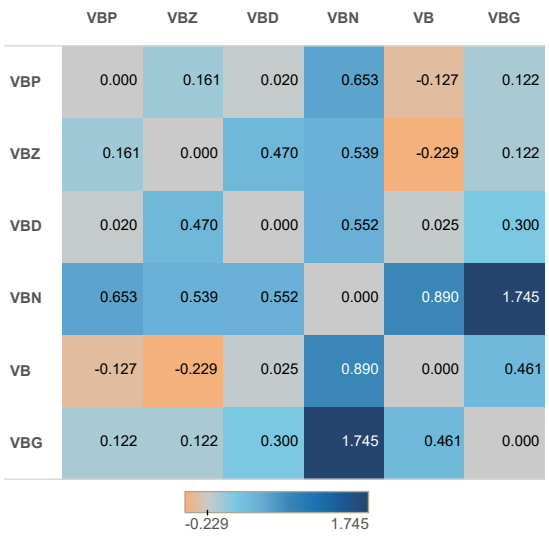


Figure 5: The change of the total variation distances between probabilities of CHILD rules headed by different verb tags in our model vs. the baseline. A positive value means the distance is reduced in our model compared with that in the baseline.

become smaller in our model. This verifies that our approach smooths the probabilities of rules involving correlated POS tags. From the figure one can see that the distance that reduces the most is between VBG and VBN. These two verb tags indeed have very similar syntactic behaviors and thus have similar embeddings as shown in figure 4. One the other hand, the distances between VB and VBZ/VBP become larger. This is reasonable since VB is syntactically different from VBZ/VBP in that it is very likely to generate a child tag TO to the left while VBZ/VBP always generate a subject (e.g., a noun or a pronoun) to the left.

6 Conclusion

We propose a neural based DMV model to do unsupervised dependency parsing. Our approach learns neural networks with continuous representations of POS tags to predict the probabilities of grammar rules, thus automatically taking into account the correlations between POS tags. Our experiments show that our approach outperforms previous approaches utilizing POS correlations and is competitive with recent state-of-the-art approaches on nine different languages.

For future work, we plan to extend our approach in learning lexicalized DMV models. In addition, we plan to apply our approach to other unsupervised tasks such as word alignment and sentence clustering.

References

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

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics.

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

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82. Association for Computational Linguistics.

Shay B Cohen and Noah A Smith. 2010. Covariance in unsupervised learning of probabilistic grammars. *The Journal of Machine Learning Research*, 11:3017–3051.

Shay B Cohen, Kevin Gimpel, and Noah A Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, pages 321–328.

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 209–216. ACM.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Nikhil Garg and James Henderson. 2011. Temporal restricted boltzmann machines for dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 11–17. Association for Computational Linguistics.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graça. 2012. The pascal challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80. Association for Computational Linguistics.

Jennifer Gillenwater, Kuzman Ganchev, Joao Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199. Association for Computational Linguistics.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado, May–June. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Heshaam Faili. 2012. Fast unsupervised dependency parsing with arc-standard transitions. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 1–9. Association for Computational Linguistics.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759. Association for Computational Linguistics.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *EMNLP*, pages 1983–1995.

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334. Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Generating Coherent Summaries of Scientific Articles Using Coherence Patterns

Daraksha Parveen **Mohsen Mesgar** **Michael Strube**

NLP Group and Research Training Group AIPHES
Heidelberg Institute for Theoretical Studies gGmbH
Heidelberg, Germany
{daraksha.parveen|mohsen.mesgar|michael.strube}@h-its.org

Abstract

Previous work on automatic summarization does not thoroughly consider coherence while generating the summary. We introduce a graph-based approach to summarize scientific articles. We employ coherence patterns to ensure that the generated summaries are coherent. The novelty of our model is twofold: we mine coherence patterns in a corpus of abstracts, and we propose a method to combine coherence, importance and non-redundancy to generate the summary. We optimize these factors simultaneously using Mixed Integer Programming. Our approach significantly outperforms baseline and state-of-the-art systems in terms of coherence (summary coherence assessment) and relevance (ROUGE scores).

1 Introduction

The growth in the scientific output of many different fields makes the task of automatic summarization imperative. Automatic summarizers assist researchers to have an informative and coherent gist of long scientific articles. An automatic summarizer produces summaries considering three properties:
Importance: The summary should contain the important information of the input document.
Non-redundancy: The summary should contain non-redundant information. The information should be diverse in the summary.
Coherence: Though the summary should comprise diverse and important information of the input document, its sentences should be connected to one another such that it becomes coherent and easy to read.

If we do not ensure that a summary is coherent, its sentences may not be properly connected. This results in an obscure summary. In previous work coherence has not been thoroughly considered. Parveen and Strube (2015) use single sentence connectivity in the input document as a coherence measure. They measure coherence by calculating the outdegree of a sentence in a graph representation of an input document. This has two disadvantages: first, since it is computed only based on one sentence, it is not sufficient to generate coherent summaries; second, it is obtained based on sentence connectivity in the input document rather than in the summary.

In this work, we focus on the coherence aspect of summarization. We use discourse entities as the unit of information that relate sentences. Here, discourse entities are referred to as head nouns of noun phrases (see Section 2). The main goal is to extract sentences which refer to those entities which are important and unique, and also to entities which connect the extracted sentences in a coherent manner. Entities in connected sentences can be used to create linguistically motivated coherence patterns (Daneš, 1974). Recently, Mesgar and Strube (2015) modeled these coherence patterns by subgraphs of the graph representation (nodes represent sentences and edges represent entity connections among sentences) of documents. They show that the frequency of coherence patterns can be used as features for coherence.

The key idea of this paper is to apply coherence patterns to long scientific articles to extract (possibly) non-adjacent sentences which, however, are already coherent. Based on the assumption that ab-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 772–783,
Austin, Texas, November 1-5, 2016.

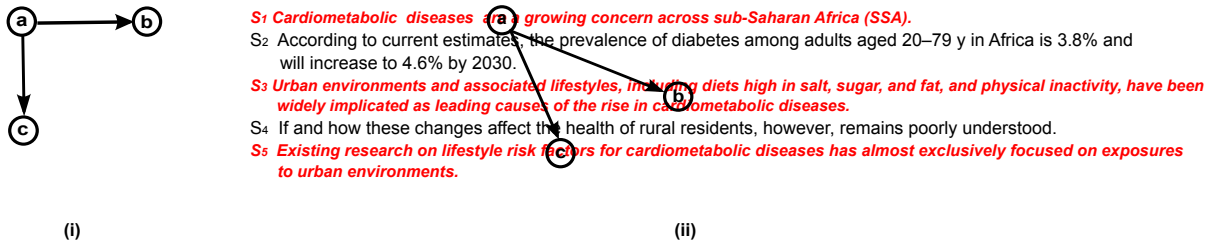


Figure 1: (i) A sample of mined coherence patterns from abstracts; nodes are sentences and edges are entity connections; (ii) Sentences S_1, S_3 and S_5 constitute the pattern in an input document.

stracts of scientific articles are similar in style to coherent summaries, we obtain coherence patterns by analyzing a corpus of abstracts of articles from bio-medicine (*PubMed* corpus). Then we apply the most frequent coherence patterns to input documents, i.e. long scientific articles from bio-medicine (*PLOS Medicine* dataset), extract corresponding sentences to generate coherent summaries, and evaluate them by comparing with summaries written by a *PLOS Medicine* editor. Figure 1 illustrates the extraction of sentences from an input document (Figure 1, (ii)) which constitute a coherence pattern (Figure 1, (i)). If we overlay the input document with coherence patterns and extract the sentences which constitute those patterns, then the extracted sentences are already coherent. We also take into account importance and non-redundancy. We capture all three factors in an objective function maximized by Mixed Integer Programming (MIP) (Section 2).

We evaluate our method on two different datasets: *PLOS Medicine* (Parveen and Strube, 2015) and *DUC 2002*. We extract frequent coherence patterns from all abstracts in the *PubMed* corpus, and generate summaries of unseen scientific articles of the *PLOS Medicine* dataset (Section 3.1). For DUC 2002 we extract coherence patterns from the human summaries of DUC 2005 (Dang, 2005). We evaluate our model on DUC 2002 to compare with state-of-the-art systems.

Our experimental results show that using coherence patterns for summarization produces more informative (but not redundant) and coherent summaries as compared to several baseline methods and state-of-the-art methods based on ROUGE scores and human judgements.

2 Method

We solve the task of creating coherent summaries by employing coherence patterns. We tightly integrate determining importance, non-redundancy and coherence by applying global optimization, i.e., MIP.

2.1 Document Representation

We use the entity graph (Guinaudeau and Strube, 2013) to represent scientific articles. The entity graph is a bipartite graph which consists of entities and sentences as two disjoint sets of nodes (Figure 2, ii). Entity nodes are connected only with sentence nodes and not among each other. An entity node is connected with a sentence node if and only if the entity is present in the sentence. Entities are the head nouns of noun phrases.

We perform a one-mode projection on sentence nodes to create a directed one-mode projection graph (Figure 2, iii). Two sentence nodes in the one-mode projection graph are connected if they share at least one entity in the entity graph. Edge directions encode the sentence order in the input document.

2.2 Mining Coherence Patterns

We use one-mode projection graphs of abstracts in the *PubMed* corpus (see Section 3.1) to mine coherence patterns. The weight of a coherence pattern, $weight(pat_u)$, is its frequency in the *PubMed* corpus normalized by the maximum number of its occurrence in abstracts in the *PubMed* corpus (Equation 1).

$$weight(pat_u) = \frac{\sum_{k=1}^{q} freq(pat_u, g_k)}{max_{k=1}^{q} freq(pat_u, g_k)}, \quad (1)$$

where q is the number of graphs associated with abstracts in the corpus, and g_k represents the graph of the k^{th} abstract in the *PubMed* corpus.

S_1 The overall $[rates]_{e_1}$ of cesarean $[delivery]_{e_2}$ are increasing significantly in the $[world]_{e_3}$.

S_2 In $[parts]_{e_4}$ of $[England]_{e_5}$ in 2010, the $[proportion]_{e_6}$ of total $[births]_{e_7}$ by cesarean $[section]_{e_8}$ was almost 25%, compared with just 2% in the 1950s.

S_3 In the United States and Australia rates of greater than 33% have been reported and in $[China]_{e_9}$ and $[parts]_{e_4}$ of South $[America]_{e_{10}}$, including Brazil and $[Paraguay]_{e_{11}}$, cesarean $[rates]_{e_1}$ of between 40% and 50% are common.

S_4 $[Concerns]_{e_{12}}$ have been expressed regarding the $[impact]_{e_{13}}$ of a cesarean $[section]_{e_8}$ on subsequent $[pregnancy]_{e_{14}}$ $[outcome]_{e_{15}}$ particularly the $[rate]_{e_{16}}$ of subsequent $[stillbirth]_{e_{17}}$, $[miscarriage]_{e_{18}}$, and ectopic pregnancy.

S_5 Hypothesized biological $[mechanisms]_{e_{19}}$ include placental $[abnormalities]_{e_{20}}$, prior $[infection]_{e_{21}}$, and adhesion $[formation]_{e_{22}}$ due to cesarean $[section]_{e_8}$.

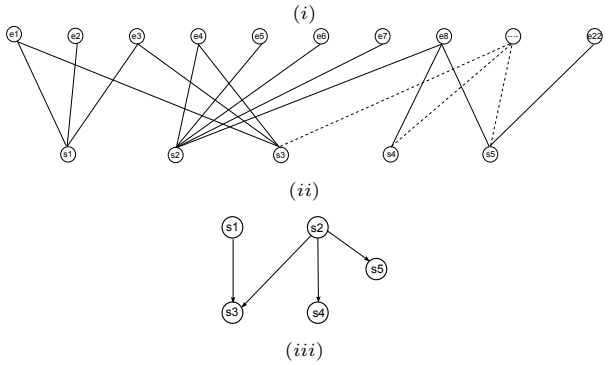


Figure 2: (i) A sample text from *PLOS Medicine*; (ii) entity graph; (iii) projection graph of the text.

The weights of the coherence patterns are not on the same scale. We normalize the weights using the standard score $\left(\frac{x-\mu}{\sigma}\right)$, where μ is the mean and σ is the standard deviation. A sigmoid function scales weights to the interval $[0, 1]$.

2.3 Summary Generation

We maximize importance, non-redundancy and pattern-based coherence with their respective weights λ to generate coherent summaries. The objective function is:

$$max(\lambda_I f_I(S) + \lambda_R f_R(E) + \lambda_C f_C(P)), \quad (2)$$

where S is a set of binary variables for sentences in an article, E is a set of binary variables for entities and P is a set of binary variables for coherence patterns.

Importance ($\mathbf{f_I(S)}$): The importance function quantifies the overall importance of information in the summary, which is calculated by considering the ranks of selected sentences for the summary:

$$f_I(S) = \sum_{i=1}^{n} Rank(sent_i) \cdot s_i. \quad (3)$$

In Equation 3, $Rank\,(sent_i)$ represents the rank of sentence $sent_i$ and s_i is the binary variable of sentence $sent_i$. n is the number of sentences.

Kleinberg (1999) develops the Hubs and Authorities algorithm (HITS) to rank web pages. He divides web pages into two sets: Hubs, pages which contain links to informative web pages, and Authorities, informative web pages. Here, Hubs are entities and Authorities are sentences. We calculate the rank of sentences using the HITS algorithm (Parveen and Strube, 2015). Initial ranks for sentences and entities are computed by Equations 4 and 5 in an entity graph:

$$Rank_{init}(sent_i) = 1 + sim(sent_i, title), \quad (4)$$

$$Rank_{init}(ent_j) = 1. \quad (5)$$

In Equation 4, $sim(sent_i, title)$ is the cosine similarity between the scientific article's title and sentence $sent_i$. In Equation 5, ent_j refers to the j^{th} entity in the entity graph. After applying the HITS algorithm on the entity graph using the above initialization, the final rank of a sentence is its importance.

Non-redundancy ($\mathbf{f_R}$ (E)): In the objective function, $f_R(E)$ represents the non-redundancy of information in the summary. Intuitively, if the summary has unique information in every sentence then the summary is non-redundant. We measure non-redundancy as follows:

$$f_R(E) = \sum_{j=1}^{m} e_j, \quad (6)$$

where m is the number of entities and e_j is a binary variable for each entity. The summary becomes non-redundant if we include only unique entities.

On the basis of $f_I(S)$ and $f_R(E)$ we define the following optimization constraints:

$$\sum_{i=1}^{n} |Sent_i| \cdot s_i \leq l_{max}, \quad (7)$$

$$\sum_{j \in E_i} e_j \geq |E_i| \cdot s_i \quad \text{for } i = 1, \ldots, n, \quad (8)$$

$$\sum_{s_i \in S_j} s_i \geq e_j \quad \text{for } j = 1, \ldots, m. \tag{9}$$

The constraint in Equation 7 limits the length of the summary. l_{max} is the maximal length of the summary and $|Sent_i|$ is the length of sentence $sent_i$.

In Equation 8, the constraint ensures that if sentence $sent_i$ is selected ($s_i = 1$), then all entities E_i present in sentence $sent_i$ must also be selected. In Equation 9, S_j represents the set of binary variables of sentences which contain entity ent_j. This constraint prescribes that if entity ent_j is selected ($e_j = 1$), then at least one of the sentences in S_j must be selected, too.

Coherence ($\mathbf{f_C(P)}$): We use the mined patterns to extract sentences from the input document of *PLOS Medicine* to create a coherent summary. We extract sentences, if the connectivity among nodes in their projection graph matches the connectivity among nodes in a coherence pattern. In Figure 3 we overlay the projection graph from Figure 2, ii with the coherence pattern from Figure 1, i. This results in three instances of this coherence pattern. However, we select only one since we simultaneously optimize for importance and non-redundancy.

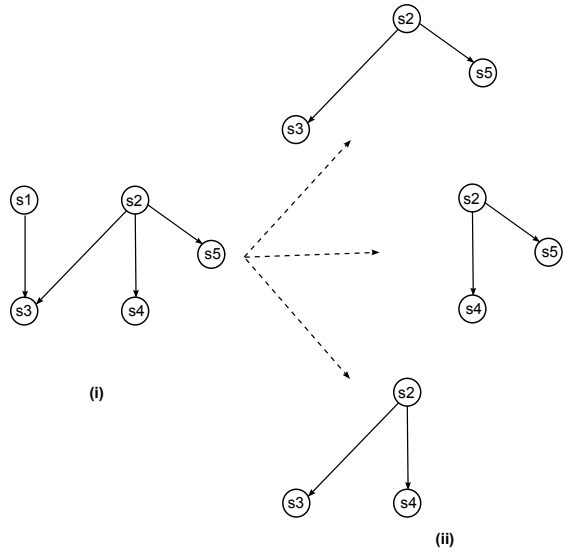


Figure 3: (i) A projection graph; (ii) several instances of a coherence pattern in Figure 1, ii.

In the objective function, $f_C(P)$ measures the coherence of the summary based on the weights of the coherence patterns occurring in it (Section 2.2):

$$f_C(P) = \sum_{u=1}^{U} weight(pat_u) \cdot p_u, \tag{10}$$

where p_u is a boolean variable associated with coherence pattern pat_u.

The optimization considers pattern pat_u for summarizing the input article, if pat_u is a subgraph of the projection graph of the article. To find the coherence pattern in a projection graph we apply a graph matching algorithm (Lerouge et al., 2015).

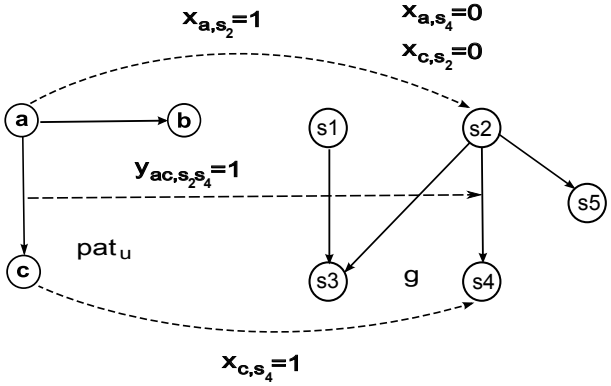


Figure 4: An illustration of mapping variables to overlay graph g with coherence pattern pat_u.

To model the graph matching problem between projection graph $g = (V_g, E_g)$ and patterns $pat_u = (V_{pat_u}, E_{pat_u})$, two kinds of mapping binary variables are used: $x_{i,k}$ for the node map, and $y_{ij,kl}$ for the edge map. $x_{i,k,} = 1$, if vertices $i \in V_{pat_u}$ and $k \in V_g$ match. $y_{ij,kl} = 1$, if for each pair of edges $ij \in E_{pat_u}$ and $kl \in E_g$ match (Figure 4). Constraints for graph matching are as follows:

- Every node of the pattern matches at most one unique node of the graph:

$$\sum_{k \in V_g} x_{i,k} \leq 1 \quad \forall i \in V_{pat_u}. \tag{11}$$

- Every edge of the pattern matches at most one unique edge of the graph:

$$\sum_{kl \in E_g} y_{ij,kl} \leq 1 \quad \forall ij \in E_{pat_u}. \tag{12}$$

- Every node of the graph matches at most one node of the pattern:

$$\sum_{i \in V_{pat_u}} x_{i,k} \leq 1 \quad \forall k \in V_g. \tag{13}$$

- A node of pattern pat_u matches a node of graph g if an edge originating from the node of pat_u matches an edge originating from the node of g:

$$\sum_{kl \in E_g} y_{ij,kl} = x_{i,k} \ \forall k \in V_g, \forall ij \in E_{pat_u}. \tag{14}$$

- A node of pattern pat_u matches a node of graph g if an edge targeting the node of pat_u matches an edge targeting the node of g:

$$\sum_{kl \in E_g} y_{ij,kl} = x_{j,l} \ \forall l \in V_g, \forall ij \in E_{pat_u}. \tag{15}$$

- We need a constraint to extract *induced* patterns[1]:

$$\sum_{i \in V_{pat_u}} x_{i,k} + \sum_{j \in V_{pat_u}} x_{j,l} - \sum_{ij \in E_{pat_u}} y_{ij,kl} \leq 1 \quad \forall kl \in E_g. \tag{16}$$

The constraints in Equations $11-16$ are defined to find pattern pat_u in projection graph g of the input article. However these constraints do not ensure that the pattern is in the summary. For this, we define constraints in Equations $17-19$ to assure that an existing pattern in an article is selected if there are some sentences in the summary which constitute the pattern.

- The constraint in Equation 17 ensures that if sentences s_k and s_l are selected for the summary then the edge between them is selected ($z_{kl} = 1$), too:

$$s_k \cdot s_l = z_{kl} \quad \forall k, l \in V_g. \tag{17}$$

- Pattern pat_u is present in the summary ($p_u = 1$) if and only if one of its instances in the projection graph is included in the summary, i.e., some of the selected sentence nodes must be present in an instance of pattern pat_u. $|V_{pat_u}|$ is the number of nodes in pattern pat_u, and $|E_{pat_u}|$ is the number of edges in pattern pat_u. This constraint is shown below:

$$\sum_{i \in v_{pat_u}} \sum_{k \in v_g} s_k \cdot x_{i,k} + \sum_{ij \in e_{pat_u}} \sum_{kl \in e_g} z_{kl} \cdot y_{ij,kl} = p_u(|V_{pat_u}| + |E_{pat_u}|). \tag{18}$$

- If a sentence is selected then it has to match a node of at least one of the patterns:

$$\sum_{pat_u \in P} \sum_{i \in V_{pat_u}} x_{i,k} \geq s_k \quad \forall k \in V_g. \tag{19}$$

3 Experiments

In this section we discuss the datasets and the experimental setup. We evaluate our model using ROUGE scores and human judgements.

3.1 Datasets

PLOS Medicine: This dataset contains 50 scientific articles. In this dataset every scientific article is accompanied by a summary written by an editor of the month. This editor's summary has a broader perspective than the authors' abstract. We use the editor's summary as a gold summary for calculating the ROUGE scores. We use 700 different *PLOS Medicine* articles from the PubMed[2] corpus to mine coherence patterns from their abstracts and to calculate patterns' weights.

DUC: The DUC 2002 dataset has been annotated for the Document Understanding Conference 2002. It contains 567 news articles for summarization. Every article is accompanied by at least two gold summaries. DUC 2002 articles are shorter than *PLOS Medicine* articles (25 vs. 154 sentences average length). We use all (300) DUC 2005 human summaries to mine coherence patterns and to calculate their weights.

3.2 Experimental Setup

First, we extract the text of an article. We remove figures, tables, references and non-alphabetical characters. Then we use the Stanford parser (Klein and

[1] Pattern pat_u is an induced subgraph of graph g if pat_u contains all possible edges which appear in g.

[2] http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/

Manning, 2003) to determine sentence boundaries. We apply the Brown coherence toolkit (Elsner and Charniak, 2011) to convert the articles into entity grids (Barzilay and Lapata, 2008) which then are transformed into entity graphs. We use gSpan (Yan and Han, 2002) to extract all subgraphs from the projection graphs of the abstracts of the *PubMed* corpus.

It is possible that patterns with a large number of nodes are not at all present in the projection graph. Hence, we use coherence patterns with 3 and 4 nodes, referred to as CP_3 and CP_4, respectively. We use Gurobi (Gurobi Optimization, Inc., 2014) to solve the MIP problem. We use a pronoun resolution system (Martschat, 2013) to replace all pronouns in the summary with their antecedents.

We determine the best values for λ_I, λ_R, and λ_c on the development sets. $\lambda_I = 0.4$, $\lambda_R = 0.3$, and $\lambda_c = 0.3$ are the best weights for the *PLOS Medicine* development set. Weights for the DUC 2002 development set are $\lambda_I = 0.5$, $\lambda_R = 0.2$ and $\lambda_c = 0.3$.

3.3 Results

We evaluate our model in two ways. First, we use ROUGE scores to compare our model with other models. Second, we explicitly evaluate the coherence of the summaries by human judgements.

3.3.1 ROUGE Assessment

The ROUGE score (Lin, 2004) is a standard evaluation score in automatic text summarization. It calculates the overlap between gold summary and system summary. In automatic text summarization ROUGE 1, ROUGE 2 and ROUGE SU4 are usually reported (see Graham (2015) for an assessment of evaluation metrics for summarization).

We compare our system (CP_3 and CP_4) with four baselines: *Lead, Random, Maximal Marginal Relevance (MMR)* and *TextRank*. *Lead* selects adjacent sentences from the beginning of an input article. *Random* selects sentences randomly. *MMR* (Carbonell and Goldstein, 1998) uses a trade-off between relevance and redundancy. *TextRank* is a graph-based system using sentences as nodes and edges weighted by cosine similarity between sentences (Mihalcea and Tarau, 2004).

We compare our system with three state-of-the-art systems: E_{Coh} (Parveen and Strube, 2015), T_{Coh} (Parveen et al., 2015), and *Mead* (Radev et al., 2004). E_{Coh} uses entity graphs which consists of entities and sentences, and T_{Coh} uses topical graphs where entities are replaced by the topics. They both use the outdegree of sentence nodes in the unweighted and the weighted projection graph, respectively, as the coherence measure of each sentence. *Mead* employs a linear combination of three features: centroid score, position score and overlap score. The linear combination is used to add sentences to the summary up to the required length. The centroid score gives the highest score to the most central sentence in the cluster of sentences, the position score gives a higher score to the sentences which are in the beginning of the document, and the overlap score computes the similarity between the sentences of a document. All three features do not take care of the coherence of a summary as they do not have any notion of the order and the structure of a summary.

Systems	R-SU4	R-2
Baselines		
Lead	0.067	0.055
Random	0.048	0.031
MMR	0.069	0.048
TextRank	0.068	0.048
State-of-the-art		
E_{Coh}	0.131	0.098
T_{Coh}	0.129	0.095
Mead	0.084	0.068
Our Model		
$\mathbf{CP_3}$	**0.135**	**0.103**

Table 1: *PLOS Medicine*, editor's summaries with **5 sentences**.

To compare with the state-of-the-art systems on PLOS Medicine, E_{Coh} (Parveen and Strube, 2015) and T_{Coh} (Parveen et al., 2015), we limit the length of summaries to 5 sentences. Table 1 reports ROUGE scores of different systems. Our system outperforms baselines and state-of-the-art systems.

Since the word length limit of a summary is more meaningful than the sentence length limit of a summary, we limit the length of a summary to the average length of editor's summaries in the dataset (750 words). Table 2 shows the performance of different systems with 750 words limit for a summary. In Table 2, we use different versions of ROUGE-SU4 and ROUGE-2 where *W/WO* stands

PLOS Medicine Editor's summaries	WO_{Stop} W_{Stem}	WO_{Stop} WO_{Stem}	W_{Stop} W_{Stem}	W_{Stop} WO_{Stem}	WO_{Stop} W_{Stem}	WO_{Stop} WO_{Stem}	W_{Stop} W_{Stem}	W_{Stop} WO_{Stem}
	ROUGE SU4 (*$p_{value} < 0.05$)				ROUGE 2 (*$p_{value} < 0.01$)			
Upper Bound	0.423	0.354	0.519	0.470	0.344	0.304	0.430	0.399
Baselines								
Lead	0.191	0.158	0.246	0.222	0.158	0.140	0.185	0.171
Random	0.140	0.113	0.169	0.153	0.102	0.088	0.125	0.116
MMR	0.183	0.149	0.240	0.215	0.141	0.125	0.171	0.157
TextRank	0.148	0.104	0.161	0.159	0.115	0.084	0.126	0.118
State-of-the-art								
E_{Coh}	0.204*	0.167	0.254	0.228	0.160*	0.145	0.187	0.173
T_{Coh}	0.195	0.161	0.231	0.206	0.157	0.140	0.169	0.165
Mead	0.197	0.165	0.246	0.222	0.156	0.139	0.186	0.172
Our Model								
CP_3	0.215*	0.178	0.268	0.241	0.172*	0.153	0.200	0.184
CP_4	0.218	0.179	0.270	0.245	0.175	0.156	0.201	0.187

Table 2: ROUGE scores on *PLOS Medicine* with **750 words**.

for *With/Without*. Here, WO_{Stop} means without considering stopwords while calculating ROUGE scores, and WO_{Stem} means without applying the Porter Stemmer on summaries while calculating ROUGE scores. Our models outperform baseline and state-of-the-art systems (Table 2). We compute statistical significance between E_{Coh} and CP_3 on both scores, ROUGE SU4 is significantly different by 95%. ROUGE 2 is significantly different by 99%.

Upper Bound in Table 2 represents maximum ROUGE scores that can be achieved in extractive summarization on the *PLOS Medicine* dataset. It is calculated by considering the whole scientific article as a summary and the corresponding editor's summary as the gold standard. The *Upper Bound* scores are not very high showing that a significant improvement in ROUGE scores on the *PLOS Medicine* dataset is difficult. Thus, the performance achieved by our systems, CP_3 and CP_4, is a considerable improvement on the *PLOS Medicine* dataset.

Furthermore, we apply CP_3 on the dataset introduced by Liakata et al. (2013). The dataset consists of 28 scientific articles from the chemistry domain. The state-of-the-art system on this dataset is *CoreSC*, which is developed by Liakata et al. (2013). *CoreSC* considers discourse information while summarizing a scientific article. The ROUGE-1 score of CP_3 (0.96) is significantly better than *CoreSC* (0.75) and *Microsoft Office Word 2007 AutoSummarize* (0.73) (García-Hernández et al., 2009), in respect of abstracts. This shows that our system performs well in other domains.

We further calculate the average number of sentences per summary obtained by *Mead* and CP_3. On average *Mead* produces 17.5 sentences per summary whereas CP_3 produces 27.2 sentences per summary. The possibility of longer sentences containing more topic irrelevant entities is higher than shorter sentences (Jin et al., 2010).

We calculate the average percentage of sentences selected from the sections Introduction, Method, Results and Discussion by different systems. CP_3 extracts sentences mainly from Introduction (32.5%) and Method (38.5%), but also a considerable number of sentences from Results (17.67%) and Discussion (11.33%). The distribution is quite similar to *TextRank* and *MMR*. *Lead*, obviously, extracts only from Introduction (80.59%) and Method (19.41%). *Mead* extracts maximum sentences from the beginning of the document using its positional feature. The sentences in a summary extracted by CP_3 are evenly distributed indicating that they are not biased to any sections. This clearly represents that coherence patterns not only seeks for nearby sentences but also for any distant sentences of a scientific article.

Table 3 shows the results on DUC 2002 to compare the results with state-of-the-art systems. There is no significant difference between the ROUGE scores of using CP_3 and CP_4 on DUC 2002. Thus, we only report the results of using CP_3 on DUC 2002.

In Table 3, *LREG* is a baseline system us-

Systems	R-1	R-2	R-SU4
Baselines			
Lead	0.459	0.180	0.201
DUC 2002 Best	0.480	0.228	
TextRank	0.470	0.195	0.217
LREG	0.438	0.207	
State-of-the-art			
Mead	0.445	0.200	0.210
ILP_{phrase}	0.454	0.213	
URANK	0.485	0.215	
UniformLink (k = 10)	0.471	0.201	
E_{Coh}	0.485	0.230	0.253
T_{Coh}	0.481	0.243	0.242
NN-SE	0.474	0.230	
Our Model			
$\mathbf{CP_3}$	**0.490**	**0.247**	**0.258**

Table 3: ROUGE scores on DUC 2002.

ing logistic regression and hand-made features (Cheng and Lapata, 2016). We compare our model to previously published state-of-the-art systems. These systems show reasonable performance on the DUC 2002 summarization task. ILP_{phrase} is a phrase-based extraction model, which selects important phrases and combines them via integer linear programming (Woodsend and Lapata, 2010). *URANK* utilizes a unified ranking process for single-document and multi-document summarization tasks (Wan, 2010). *UniformLink (k=10)*, considers similar documents for document expansion in the single-document summarization task (Wan and Xiao, 2010). The more recent system, *NN-SE*, utilizes a neural network hierarchical document encoder and an attention-based extractor to extract sentences from a document for a summary (Cheng and Lapata, 2016). ROUGE scores of our approach on this dataset are better than baselines and state-of-the-art systems. This shows that our system performs well even in a different genre (robust) and with considerably shorter input documents (scalable).

3.3.2 Coherence Assessment

ROUGE scores do not evaluate summary coherence, since ROUGE only calculates overlapping recall scores and does not consider the structure of the summary. Haghighi and Vanderwende (2009), Celikyilmaz and Hakkani-Tür (2010) and Christensen et al. (2013) evaluate the overall summary quality by asking human subjects to rank system generated summaries. Parveen and Strube (2015) and Parveen et al. (2015) assess the coherence by asking human assessors to rank system generated summaries and compare their system with baseline systems.

We perform summary coherence assessment by asking one Postdoc, two PhD students and one Masters student from the field of natural language processing. We provide them with the output summaries of four different systems for ten articles. We ask them to rank the summaries, i.e., the best summary gets rank 1, the second best gets rank 2, the third best gets rank 3, and the worst gets rank 4.

The four systems assessed are CP_3, E_{Coh}, *TextRank*, and *Lead*. We apply the Kendall concordance coefficient (W) (Siegel and Castellan, 1988) to measure whether the human assessors agree in ranking the four systems. With W $= 0.6725$ the correlation between the human assessors is high. Applying the χ^2 test shows that W is significant at least at the 99% level indicating that the ranks provided by the human assessors are reliable and informative. Table 4 shows the overall average rank of a system given by the four human assessors. The lower the value of average human scores the more coherent the summary. Unsurprisingly *Lead* gets the best overall average rank. *Lead* extracts adjacent sentences from the beginning of the document. Hence, these summaries are as coherent as the author intends them to be, but they are not informative. However, CP_3 is very close in coherence to *Lead* indicating that our strategy is successful. It also performs substantially better than *TextRank* and E_{Coh}. This confirms that using coherence patterns for sentence extraction yields more coherent summaries.

PLOS Medicine System	Average Human Score
TextRank	3.950
E_{Coh}	2.325
$\mathbf{CP_3}$	**1.875**
Lead	1.625

Table 4: The average human scores.

4 Related Work

Summarizing scientific articles is as difficult as multi-document summarization because scientific articles are tend to be long and the important infor-

mation is spread all over the article unlike information in news articles (Teufel and Moens, 2002).

There are various approaches for summarizing scientific articles. Citations have been used by many researchers for summarization in this domain (Elkiss et al., 2008; Mohammad et al., 2009; Qazvinian and Radev, 2008; Abu-Jbara and Radev, 2011). Nanba and Okumura (2000) develop rules for categorizing citations by analyzing citation sentences. Newman (2001) analyzes the structure using a citation network. Similarly, Siddharthan and Teufel (2007) discover scientific attributions using citations. Discourse structure (but not necessarily coherence) has been used by Teufel and Moens (2002), Liakata et al. (2013) and others for summarizing scientific articles.

Several state-of-the-art extractive summarization systems implement summarization as maximizing an objective function using constraints. McDonald (2007) interprets text summarization as a global inference problem, where he is maximizing the importance score of a summary by considering the length constraint. Similarly, various approaches for summarization are based on optimization using ILP (Gillick et al., 2009; Nishikawa et al., 2010; Galanis et al., 2012; Parveen and Strube, 2015).

Until now, only few works have considered coherence while summarizing scientific articles. Abu-Jbara and Radev (2011) work on citation based summarization. They preprocess the citation sentences to filter out irrelevant sentences or sentence fragments, then extract sentences for the summary. Eventually, they refine the summary sentences to improve readability. Jha et al. (2015) consider Minimum Independent Discourse Contexts (MIDC) to solve the problem of non-coherence in extractive summarization. However, none of them deals with the problem of coherence within the task of sentence selection. Sentence selection and ensuring the coherence of summaries are not tightly integrated in their techniques. They model coherence in summarization by only considering adjacent sentences.

There are few methods (Hirao et al., 2013; Parveen and Strube, 2015; Gorinski and Lapata, 2015) which integrate coherence in optimization. These methods do not take into account the overall structure of the summary. Unlike earlier methods, we incorporate coherence patterns in optimization.

5 Conclusion

We introduce a novel graph-based approach to generate coherent summaries of scientific articles. Our approach takes care of coherence distinctively by coherence patterns. We have experimented with *PLOS Medicine* and DUC 2002. The results show that the approach is robust, works on both scientific and news documents and with input documents of different length. It considerably outperforms state-of-the-art systems on both datasets. We collected human assessments to evaluate the coherence of summaries. Our system substantially outperforms baselines and state-of-the-art systems, i.e., incorporating coherence patterns produces more coherent summaries. The results show that our approach performs well in human summary coherence assessment and relevance evaluation (ROUGE scores).

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first and second authors have been supported by a Heidelberg Institute for Theoretical Studies Ph.D. scholarship. This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1. We would like to thank our colleagues Alexander Judea, Isabell Wolter, Mark-Christoph Müller and Nafise Moosavi who became human subjects for coherence assessment evaluation.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 500–509.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information*

Retrieval, Melbourne, Australia, 24–28 August 1998, pages 335–336.

Asli Celikyilmaz and Dilek Hakkani-Tür. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 815–824.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 484–494.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 1163–1173.

František Daneš, editor. 1974. *Papers on Functional Sentence Perspective*. Academia, Prague.

Hoa Trang Dang. 2005. Overview of DUC 2005. In *Proceedings of the 2005 Document Understanding Conference held at the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 9–10 October 2005.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Portland, Oreg., 19–24 June 2011, pages 125–129.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *Proceedings of the 24th International Conference on Computational Linguistics,* Mumbai, India, 8–15 December 2012, pages 911–926.

René Arnulfo García-Hernández, Yulia Ledeneva, Griselda Matías Mendoza, Ángel Hernández Dominguez, Jorge Chavez, Alexander Gelbukh, and José Luis Tapia Fabela. 2009. Comparing commercial tools and state-of-the-art methods for generating text summaries. In *Proceedings of Advances in Artificial Intelligence, 8th Mexican International Conference on Artificial Intelligence,* Guanajuato, Mexico, 9-13 November 2009, pages 92–96.

Daniel Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. 2009. A global optimization framework for meeting summarization. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing,* Taipei, Taiwan, 19–24 June 2009, pages 4769–4772.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Denver, Col., 31 May – 5 June 2015, pages 1066–1076.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 128–137.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 93–103.

Gurobi Optimization, Inc. 2014. Gurobi optimizer reference manual.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies 2009: The Conference of the North American Chapter of the Association for Computational Linguistics,* Boulder, Col., 31 May – 5 June 2009, pages 362–370.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1515–1520.

Rahul Jha, Reed Coke, and Dragomir Radev. 2015. Surveyor: A system for generating coherent survey articles for scientific topics. In *Proceedings of the 29th Conference on the Advancement of Artificial Intelligence,* Austin, Texas, 25–30 January 2015, pages 2167–2173.

Feng Jin, Minlie Huang, and Xiaoyan Zhu. 2010. A comparative study on ranking and selection strategies for multi-document summarization. In *Proceedings of Coling 2010: Poster Volume,* Beijing, China, 23–27 August 2010, pages 525–533.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational*

Linguistics, Sapporo, Japan, 7–12 July 2003, pages 423–430.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Julien Lerouge, Pierre Le Bodic, Pierre Héroux, and Sébastien Adam. 2015. GEM++: A tool for solving substitution-tolerant subgraph isomorphism. In C.-L. Liu, B. Luo, W.G. Kropatsch, and J. Cheng, editors, *Graph-Based Representations in Pattern Recognition*, pages 128–137. Springer, Heidelberg, Germany.

Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2013. A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 747–757.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Text Summarization Branches Out Workshop at ACL '04,* Barcelona, Spain, 25–26 July 2004, pages 74–81.

Sebastian Martschat. 2013. Multigraph clustering for unsupervised coreference resolution. In *51st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Student Research Workshop,* Sofia, Bulgaria, 5–7 August 2013, pages 81–88.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the European Conference on Information Retrieval,* Rome, Italy, 2-5 April 2007.

Mohsen Mesgar and Michael Strube. 2015. Graph-based coherence modeling for assessing readability. In *Proceedings of STARSEM 2015: The Fourth Joint Conference on Lexical and Computational Semantics,* Denver, Col., 4–5 June 2015, pages 309–318.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing,* Barcelona, Spain, 25–26 July 2004, pages 404–411.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *Proceedings of Human Language Technologies 2009: The Conference of the North American Chapter of the Association for Computational Linguistics,* Boulder, Col., 31 May – 5 June 2009, pages 584–592.

Hidetsugu Nanba and Manabu Okumura. 2000. Producing more readable extracts by revising them. In *Proceedings of the 18th International Conference on Computational Linguistics,* Saarbrücken, Germany, 31 July – 4 August 2000, pages 1071–1075.

Mark E.J. Newman. 2001. Scientific collaboration networks. I. Network construction and fundamental results. *Physical Review E*, 64(1):016131.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Proceedings of the 23rd International Conference on Computational Linguistics,* Beijing, China, 23–27 August 2010, pages 910–918.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence,* Buenos Aires, Argentina, 25–31 July 2015, pages 1298–1304.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 1949–1954.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics,* Manchester, U.K., 18–22 August 2008, pages 689–696.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celibi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD – a platform for multidocument multilingual text summarization. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26–28 May 2004.

Advaith Siddharthan and Simone Teufel. 2007. Whose idea was this, and why does it matter? Attributing scientific work to citations. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pages 316–223.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York, 2nd edition.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Xiaojun Wan and Jianguo Xiao. 2010. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Transactions on Information Systems*, 28(2):8 pages.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd International Conference on Computational Linguistics,* Beijing, China, 23–27 August 2010, pages 1137–1145.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 565–574.

Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-based substructure pattern mining. In *Proceedings of the International Conference on Data Mining,* Maebashi City, Japan, 9–12 December 2002, pages 721–724.

News Stream Summarization using Burst Information Networks

Tao Ge[1,2]*, **Lei Cui**[3], **Baobao Chang**[1,2], **Sujian Li**[1,2], **Ming Zhou**[3], **Zhifang Sui**[1,2]
[1]Key Laboratory of Computational Linguistics, Ministry of Education,
School of EECS, Peking University, Beijing, 100871, China
[2]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
[3]Microsoft Research
getao@pku.edu.cn, lecu@microsoft.com, chbb@pku.edu.cn
lisujian@pku.edu.cn, mingzhou@microsoft.com, szf@pku.edu.cn

Abstract

This paper studies summarizing key information from news streams. We propose simple yet effective models to solve the problem based on a novel and promising representation of text streams – *Burst Information Networks (BINets)*. A BINet can be aware of redundant information, allows global analysis of a text stream, and can be efficiently built and dynamically updated, which perfectly fits the demands of text stream summarization. Extensive experiments show that the BINet-based approaches are not only efficient and can be used in a real-time online summarization setting, but also can generate high-quality summaries, outperforming the state-of-the-art approach.

1 Introduction

Text stream summarization aims to summarize key information from a text stream containing huge numbers of documents, which is an important and useful task that can be used for many real-world applications. For example, a news portal website editor needs to summarize news streams in the past day for generating a list of headline news; an editor of Sports Weekly may want a summary of the past week news stream for editing the magazine; and geologists and meteorologists will benefit from a summary of disaster events from the past year news stream (as shown in Table 1) for their study.

In contrast to traditional text summarization tasks (e.g., single and multi-document summarization) that have been extensively studied for decades, the task of stream summarization is a younger research problem which attempts to solve a summarization problem in the big-data setting. For a text stream with millions of documents involving various topics and events, traditional single- and multi-document summarization approaches cannot address the information overload challenge. For example, a single-document summarization model will generate 1 million document summaries for a text stream with 1 million documents, which are still overwhelming for a person to learn the key information in the stream. In such cases, one needs to a summary of the whole stream instead of summaries of each document.

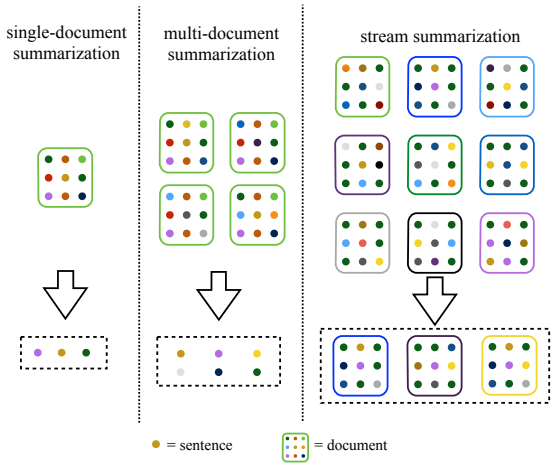


Figure 1: Stream summarization paradigm.

Figure 1 shows the paradigm of stream summarization. Compared with single- and multi-document summarization, stream summarization has three differences: (1) it summarizes a text stream containing millions of documents involving a variety of topics and events while single- and

* This work was done when the first author was visiting Microsoft Research Asia

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 784–794,
Austin, Texas, November 1-5, 2016.

2009 disaster summary	2010 disaster summary
• ... • Sep 2, 2009: About 60 people die when a 7.1-magnitude earthquake hit the island of Java. • Sep 9, 2009: More than 30 people are killed when fast moving floods caused by heavy rain sweep through Istanbul. • Sep 30, 2009: A 7.6-magnitude earthquake hit the island of Sumatra, leaving more than 1,000 people dead and thousands injured. • ...	• ... • Jan 12, 2010: A 7.0-magnitude earthquake hit Haiti, killing about 200,000 people. • Feb 27, 2010: An 8.8-magnitude earthquake rocked Chile, killing at least 700 people dead and affecting more than 1.5 million people. • Apr 5, 2010: An explosion in a West Virginia coal mine kills at least 25 people and leaves 4 unaccounted for. • ...

Table 1: Stream summary about disasters in 2009 and 2010. The disaster summary of 2009 can be used a reference summary to supervise generating a disaster summary for the 2010 news stream.

multi-document summarization summarizes one or a handful of documents about the same news event; (2) instead of selecting sentences to generate a summary, stream summarization selects representative documents to summarize a text stream; (3) summaries for a text stream may vary significantly for users who have different interests and preferences (e.g., summaries for an environmental expert and a sports fan should not be the same). Therefore, in order to generate targeted summaries for specific users, a stream summary needs to be generated based on a reference summary. For instance, one can use the 2009 disaster summary (the left part in Table 1) as a reference to learn how to write the 2010 disaster summary (the right part in Table 1).

In general, there are three challenges for summarizing a text stream. First, a stream summarization model should be able to be aware of redundant information in the stream for avoiding generating redundant content in the summary; second, a stream summarization algorithm should be capable of analyzing text content on the stream level for identifying the most important information in the stream; third, a stream summarization model should be efficient, scalable and able to run in an online fashion because data size of a text stream is usually huge, and it is dynamic and updated every second.

The previous approaches (e.g., (Ge et al., 2015b)) tend to cluster similar documents as event detection to avoid redundancy, rank the clusters based on their sizes and topical relevance to the reference summaries, and select one document from each cluster as representative documents. Due to the high time complexity of clustering models, their approaches usually run slowly and are not scalable.

To overcome the limitations, we propose Burst Information Networks (BINet) as a novel representation of a text stream. In a BINet (Figure 2), a node is a burst word (including entities) with the time span of one of its burst periods, and an edge between two nodes indicates how strongly they are related. Based on the BINet representation, we propose two models – NodeRank and AreaRank – for summarizing a news stream. We conduct extensive experiments to evaluate our approaches by comparing several baselines and the state-of-the-art approaches in various settings and show that the BINet-based approaches are efficient, scalable and can work in an online fashion and that they can generate high-quality summaries for a news stream, outperforming the state-of-the-art.

The major contributions of this paper are:

- We propose BINets as a novel representation of text streams. BINets can perfectly address the challenges of text stream summarization, which can be aware of information redundancy (Section 3), enables global analysis of the text stream (Section 4.1 and 4.2), and be efficiently built and updated incrementally (Section 4.3).
- We propose two ranking-based models based on the BINet representation, which can effectively learn to summarize a text stream from a reference summary, and outperform the state-of-the-art model.
- We create and release a new benchmark dataset for evaluating real-time stream summarization.

2 Stream Summarization

The task of text stream summarization is to generate a summary including key information from a

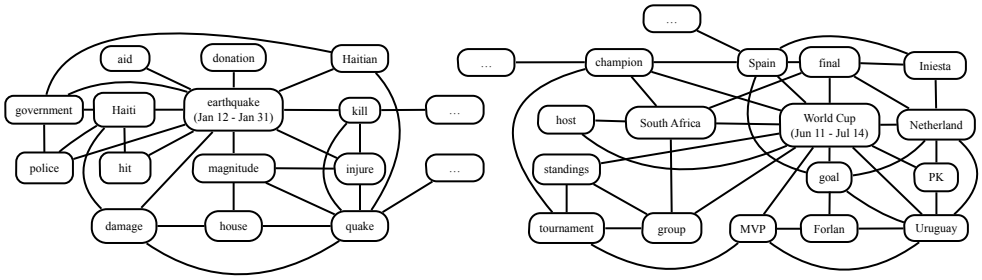


Figure 2: Illustration of a BINet. Due to space limitation, we only show the burst period of some nodes.

given text stream (e.g., 1-year news stream). In contrast to traditional summarization tasks which summarize a single or a handful of documents related to the same event by extracting sentences, the task of stream summarization aims to summarize a text stream which contains huge numbers of documents involving a variety of topics and events by selecting representative documents, as Figure 1 shows. In a stream summary, each selected document is considered as an entry which can be shown using the title or the first paragraph of the document. Since documents in a news stream are always about news events, we also call an entry as *an event entry* and call a stream summary as an *event chronicle* which is a list of event entries, as shown in Table 1. In a stream summary, entries should not be redundant. Formally, we define a stream summary (i.e., event chronicle) $\mathcal{E} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, \cdots, \boldsymbol{e_K}\}$ where $\boldsymbol{e_k} = (t_{\boldsymbol{e_k}}, \boldsymbol{w_{e_k}})$ is an event entry including the event's time information $t_{\boldsymbol{e_k}}$ and text description $\boldsymbol{w_{e_k}}$ which is set of words in text.

Due to the diversity of ways to summarize a text stream as Section 1 discusses, we use a reference summary of a text stream during an early period to supervise summary generation for new text streams. It is a practical setting since many historical manually edited summaries of early streams are available and can be used as an example to demonstrate what kind of information is preferred in a stream summary.

3 Representing a text stream using Burst Information Network

3.1 Burst

A word's burst refers to a remarkable increase in the number of occurrences of the word during a period and might indicate important events or trending topics. For example, as shown in Figure 3, the word *earthquake* has bursts from the Jan 12 to Jan 31, 2010 and from Feb 27 to Mar 8, 2010 because of the strong earthquakes occurring in Haiti and Chile respectively.

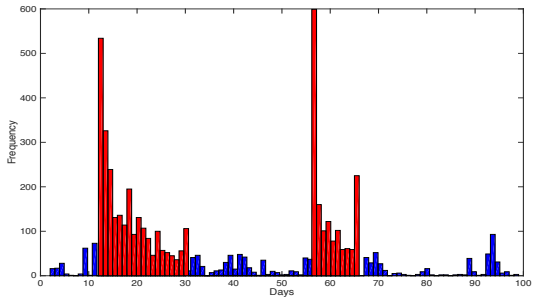


Figure 3: Frequency of *earthquake* during the first 90 days in the 2010 news stream.

Specifically, if a word w is in a burst state at every time t during a period, we call this period as a burst period of w, and w has a burst during this period. In Figure 3, *earthquake* has 2 burst periods (i.e., (Jan 12 - Jan 31) and (Feb 27 - Mar 8))

Formally, we define $\mathcal{P}$ as one burst period of the word w. $\mathcal{P}$ is a consecutive time sequence during which w bursts at every time epoch t:

$$\mathcal{P} = (t_i, t_{i+1}, t_{i+2}, ..., t_{i+n})$$
$$\forall t \in \mathcal{P} \;\; s_t = 1$$

where s_t is a binary indicator of the burst state of w at time t.

3.2 Burst Information Network

To build an information network which can represent associations between key facts in a text stream, we propose a new representation called "*Burst Information Network (BINet)*" by using burst elements as nodes:

A Burst Element is a burst of a word. It can be represented by a tuple: $\langle w, \mathcal{P} \rangle$ where w denotes the

word and $\mathcal{P}$ denotes one burst period of w.

According to the above definition, a burst element is a joint representation of a word type and one of its burst periods. A word may have multiple burst periods while a burst element only has one burst period. A word during its different burst periods will be regarded as different burst elements.

Formally, we define the BINet $G = \langle V, E \rangle$ as follows. Each node $v \in V$ is a burst element and each edge $e \in E$ denotes the association between burst elements. Intuitively, if two burst elements frequently co-occur, the edge between them should be highly weighted. We define $\omega_{i,j}$ as the weight of an edge between v_i and v_j, which is equal to the number of documents where v_i and v_j co-occur.

Besides $w(v)$ and $\mathcal{P}(v)$ that denote a node v's word and burst period respectively, we also record a node's context words[1] and its source documents which the node is from during constructing a BINet. Formally, we use $\mathcal{C}(v)$ and $\mathcal{D}(v)$ to denote the context word set and source document set of v. Also, for a document d in the stream, we use $\mathcal{A}(d)$ to denote the set of nodes whose source documents include d. Since nodes in $\mathcal{A}(d)$ are usually adjacent, we also call $\mathcal{A}(d)$ document d's area on the BINet. The construction of a BINet is efficient: the time complexity of building a BINet is $O(n)$ where n is the number of documents in a stream.

BINets can be properly aware of redundant information: since nodes in a community in a BINet are topically and temporally coherent, information about the same news event tends to be adjacent and redundant information of the same event is naturally removed. For example, assuming that there are hundreds of documents about *Haiti earthquake* in a text stream, by using the BINet representation, the information is concentrated in a few adjacent nodes without redundancy (left part in Figure 2). Moreover, information about different events is not considered as redundant. For example, the information regarding *Haiti earthquake* and *Chile earthquake* is not treated as redundant, which is allocated to different areas in the BINet, as Figure 2 shows. Therefore, as long as we do not select overlapping areas on the BINet, we can avoid selecting redundant content as entries.

[1]Here, the context window size is set to 10. Note that in our experiments, only words frequently (more than 5 times) co-occur in the context will be reserved.

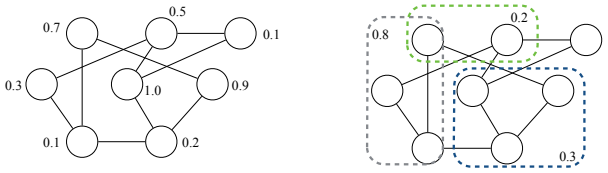


Figure 4: NodeRank (left) and AreaRank (right).

In addition to the awareness of information redundancy, BINets also allow global importance analysis on the stream level and online stream summarization, which will be discussed in Section 4.

4 Summarizing a text stream on the BINet

Based on the BINet representation, we propose two models – NodeRank and AreaRank – to summarize a text stream by generating entries of the summary. As Figure 4 shows, the NodeRank model scores every node on the BINet independently for identifying the most valuable information to be included in the stream summary, while the AreaRank model attempts to score an area that covers a handful of nodes for locating the most informative information blocks.

To train NodeRank and AreaRank models, we use reference summaries and the (reference) BINets built from the text stream during the reference summary's period as supervision.

4.1 NodeRank

Intuitively, if we can find the most valuable information on the BINet that should be included in the summary, then we can generate a high-quality summary of a text stream. For this goal, we label the corresponding nodes of words appearing in the reference summary on the reference BINet as score 1 (positive). Formally, for a reference summary $\mathcal{E}$, we label the following set of nodes in the reference BINet $G_r = \langle V_r, E_r \rangle$ as score 1:

$$V_{pos} = \bigcup_{\boldsymbol{e_k} \in \mathcal{E}} \{v | v \in V_r \wedge w(v) \in \boldsymbol{w}_{\boldsymbol{e_k}} \wedge t_{\boldsymbol{e_k}} \in \mathcal{P}(v)\} \quad (1)$$

where $w(v)$ and $\mathcal{P}(v)$ are word and burst period of node v respectively, $\boldsymbol{e_k}$ is an event entry in the reference summary $\mathcal{E}$, $\boldsymbol{w}_{\boldsymbol{e_k}}$ is the set of words in $\boldsymbol{e_k}$'s text, and $t_{\boldsymbol{e_k}}$ is $\boldsymbol{e_k}$'s time. The nodes that are not in V_{pos} in the reference BINet will be labeled as 0 (negative).

After labeling the reference BINet, we train a learning to rank (L2R) model[2] using the following features for scoring nodes in the target BINet $G_\tau = \{V_\tau, E_\tau\}$ (shown in Figure 4):

- $w(v)$: the word of node v, indicating its semantic information.
- $pr(v)$: node v's PageRank value can reflect the global importance of the node on the stream level, which can be easily obtained by running the PageRank algorithm on the BINet.
- $\mathcal{C}(v)$: the context words of node v defined in Section 3.2, indicating the topic information.

After scoring nodes in the target BINet, we greedily choose a document area $\mathcal{A}(d)$ that covers a set of nodes whose score is the largest:

$$d^* = arg \max_{d \in D_\tau} \sum_{v \in \mathcal{A}(d)} score_{NR}(v) \quad (2)$$

where D_τ is the document sets in the target stream and $score_{NR}(v)$ is the score of node v outputted by NodeRank model. Document d^*'s first paragraph and its document creation time (DCT) will be used to generate an event entry for the summary of the target stream. Note that though we do not normalize the length of a document in Eq (2), we constrain the maximum length of a document's first paragraph is 50 words and will not select the document whose first paragraph is longer than 50 words.

By repeating this step for k times, we can generate a stream summary with k event entries. Note that in order to avoid generating redundant entries in the summary, we will not choose d^* if its document area $\mathcal{A}(d^*)$ overlaps with the areas of the documents that have been already chosen as entries.

4.2 AreaRank

Instead of scoring nodes independently like NodeRank, we propose AreaRank model for scoring an area on the BINet for finding areas that corresponds to the most important news events in the stream.

Different from NodeRank where each instance is one node in the BINet, instances are areas on the BINet in the AreaRank model, as shown in Figure 4. In this paper, we mainly consider document area $\mathcal{A}(d)$ since we select representative documents as entries in the summary.

As NodeRank, we first label reference BINet using the reference summary. In the AreaRank model, we find the areas on the reference BINet corresponding to each event entry in the reference summary and label such areas as score 1 (positive). Formally, for a reference summary $\mathcal{E}$, the positive areas are in the following set:

$$\mathbb{A}_{pos} = \bigcup_{\boldsymbol{e_k} \in \mathcal{E}} \{\mathcal{A} | \mathcal{A} = V_{\boldsymbol{e_k}}\} \quad (3)$$

where $V_{\boldsymbol{e_k}} = \{v | v \in V_r \wedge w(v) \in \boldsymbol{w}_{\boldsymbol{e_k}} \wedge t_{\boldsymbol{e_k}} \in \mathcal{P}(v)\}$ is the set of nodes to which words in $\boldsymbol{e_k}$ correspond in the reference BINet.

We label other document areas that do not overlap any positive area on the reference BINet as score 0. Then, we use the training data to train AreaRank using the following features:

- $\boldsymbol{w}(\mathcal{A})$: words of nodes in area $\mathcal{A}$, indicating the area's semantic and topic information.
- $\boldsymbol{pr}(\mathcal{A})$: this feature includes maximum, sum and average of PageRank value of nodes in the area and sum of top 3 PageRank value of nodes in the area, indicating the area's general importance, which can reflects the impact of the events corresponding to the area in the stream.
- $\mathcal{C}(\mathcal{A})$: context of nodes in area $\mathcal{A}$. This feature is useful for indicating topical information.

In the test phase, we use AreaRank model to score all possible document areas on the target BINet. Then, we greedily choose the document area with the top score to generate an event entry for the summary:

$$d^* = arg \max_{d \in D_\tau} score_{AR}(\mathcal{A}(d)) \quad (4)$$

As NodeRank, d^*'s first paragraph and DCT will be used to generate an event entry for the stream summary if d^*'s area $\mathcal{A}(d)$ does not overlap the areas of the documents that have been already selected for generating event entries. The maximum length of the first paragraph of a document is 50 words. This step will be repeated for multiple times for generating event entries of the summary.

[2]We use SVMRank (Joachims, 2006). During training, we randomly sample 50% of negative examples which are used to generate the training set with positive examples.

4.3 Online stream summarization

An advantage of the BINet is that it can be incrementally updated when new streams arrive, which is useful for online stream summarization. Assuming we have a news stream from time t_0 to t_k at hand, we can detect word bursts and construct a BINet G based on the stream. When the news stream at t_{k+1} comes, we first detect burst words in the newly arriving data, update the BINet and calculate the PageRank value for $G(t_{k+1})$ which denotes the slice of BINet G at time t_{k+1}, which is defined as follows:

$$G(t) = \langle V(t), E(t) \rangle$$

where $V(t) = \{v|t \in \mathcal{P}(v)\}$ and $E(t) = \{e_{i,j}|e_{i,j} \in E \wedge i \in V(t) \wedge j \in V(t)\}$. Then, we can apply NodeRank and AreaRank on $G(t_{k+1})$ to generate a stream summary at t_{k+1}.

5 Experiments and Evaluations

5.1 Experiments on Gigaword corpus

For comparison to the previous work, we use the same data with Ge et al. (2015b) (i.e., 2009 and 2010 APW and XIN news stories in English Gigaword (Graff et al., 2003)) as a news stream. We detect burst words using Kleinberg algorithm (Kleinberg, 2003), which models word burst detection as a burst state decoding problem. In total, there are 140,557 documents in the dataset.

Topic	#Entry	#Entry in corpus
Disaster	35	28
Sports	19	12
Politics	8	5
Military	14	13
Comprehensive	85	64

Table 2: The number of event entries in the reference summaries. The third column is the number of event entries excluding those events that do not appear in the corpus.

We removed stopwords and used Stanford CoreNLP (Manning et al., 2014) to do lemmatization and named tagging, and built BINets on the news stream during 2009 and 2010 separately. On the 2009 news stream, there are 31,888 nodes and 833,313 edges while there are 32,997 nodes and 825,976 edges on the 2010 stream.

Ge et al. (2015b) used manually edited event chronicles of various topics on the web[3] during 2009 as reference summaries for summarizing the news stream during 2010. The information of the reference summaries is summarized in Table 2. In evaluation, they pooled entries in stream sumamries generated by various approaches, annotated each entry based on the reference summary and the manually edited event chronicles on the web, and used *precision@K* to evaluate the quality of top K event entries in a stream summary instead of using *ROUGE* (Lin, 2004) because news stream summaries are event-centric.

In this paper, we adopt the same evaluation setting and use the same reference summaries and the annotations with our previous work (Ge et al., 2015b) to evaluate our summaries' quality. For the event entries that are not in Ge et al. (2015b)'s annotations, we have 3 human judges annotate them according to the previous annotation guideline and consider an entry correct if it is annotated as correct by at least 2 judges.

We evaluate our approaches by comparing to Ge et al. (2015b)'s approach and the baselines in their work:

- RANDOM: this baseline randomly selects documents in the dataset as event entries.
- NB: this baseline uses Naive Bayes to cluster documents for event detection and ranks the clusters based on the combination score of topical relevance and the event impact (i.e., event cluster size). The earliest documents in the top-ranked clusters are selected as entries.
- B-HAC: similar to NB except that BurstVSM representation (Zhao et al., 2012) is used for event detection using Hierarchical Agglomerative Clustering algorithm.
- TAHBM: similar to NB except that the state-of-the-art event detection model (TaHBM) proposed by Ge et al. (2015b) is used for event detection.
- Ge et al. (2015b): the state-of-the-art stream summarization approach which used TaHBM to detect events and L2R model to rank events.

Note that we did not compare with previous multi-document summarization models because the goal and setting of stream summarization are different from multi-document summarization, as Section 1

[3] http://www.mapreport.com; http://www.infoplease.com; https://en.wikipedia.org/wiki/2009

	sports		politics		disaster		military		comprehensive	
	P@50	P@100	P@50	P@100	P@50	P@100	P@50	P@100	P@50	P@100
Random	0.02	0.08	0	0	0.02	0.04	0	0	0.02	0.03
NB	0.08	0.12	0.18	0.19	0.42	0.36	0.18	0.17	0.38	0.31
B-HAC	0.10	0.13	0.30	0.26	0.50	0.47	0.30	0.22	0.36	0.32
TaHBM	0.18	0.15	0.30	0.29	0.50	0.43	0.46	0.36	0.38	**0.33**
Ge et al. (2015b)	0.20	0.15	0.38	**0.36**	0.64	0.53	**0.54**	0.41	**0.40**	**0.33**
BINet-NodeRank	0.24	0.20	0.38	0.30	0.54	0.51	0.48	0.43	0.36	**0.33**
BINet-AreaRank	**0.40**	**0.33**	**0.40**	0.34	**0.80**	**0.62**	0.50	**0.49**	0.32	0.30

Table 3: Performance of various approaches on stream summarization on five topics.

discussed. Moreover, these two tasks differ greatly in the data size and redundancy identification mechanism. Therefore, it is not feasible to directly compare multi-document summarization models to our approaches unless they are adapted for our setting.

The results are shown in Table 3. It can be clearly observed that BINet-based approaches outperform baselines and perform comparably to the state-of-the-art model on generating the summaries on most topics: AreaRank achieves the significant improvement over the state-of-the-art model on sports and disasters, and performs comparably on politics and military and NodeRank's performance achieves the comparable performance to previous state-of-the-art model though it is inferior to AreaRank on most topics. Among these five topics, almost all models perform well on disaster and military topics because disaster and military reference summaries have more entries than the topics such as politics and sports and topics of event entries in the summaries are focused. The high-quality training data benefits models' performance especially for AreaRank which is purely data-driven. In contrast, on sports and politics, the number of entries in the reference summaries is small, which results in weaker supervision and affect the performance of models. It is notable that AreaRank does not perform well on generating the comprehensive summary in which topics of event entries are miscellaneous. The reason for the undesirable performance is that the topics of event entries in the comprehensive reference summary are not focused, which results in very few reference (positive) examples for each topic. As a result, the miscellaneousness of topics of positive examples makes them tend to be overwhelmed by large numbers of negative examples during training the model, leading to very week supervision and making it difficult for AreaRank to learn the patterns of positive examples. Compared to AreaRank, the strategy of selecting documents for generating event entries in other baselines and NodeRank use more or less heuristic knowledge, which makes these models perform stably even if the training examples are not sufficient.

Model	**Features**	**Precision@100**
NodeRank	$w(v)$	0.18
	$w(v)$+$pr(v)$	0.22
	$w(v)$+$\mathcal{C}(v)$	0.46
	$w(v)$+$pr(v)$+$\mathcal{C}(v)$	**0.51**
AreaRank	$\boldsymbol{w}(\mathcal{A})$	0.25
	$\boldsymbol{w}(\mathcal{A})$ + $\boldsymbol{pr}(\mathcal{A})$	0.34
	$\boldsymbol{w}(\mathcal{A})$+$\mathcal{C}(\mathcal{A})$	0.58
	$\boldsymbol{w}(\mathcal{A})$+$\boldsymbol{pr}(\mathcal{A})$+$\mathcal{C}(\mathcal{A})$	**0.62**

Table 4: Ablation test on feature combination for generating disaster summaries.

Model	Topic	Irrelevant	Minor	Redundant
NodeRank	disaster	35.3%	64.7%	0
	sports	21.3%	77.5%	1.3%
	comprehensive	-	100%	0
AreaRank	disaster	34.2%	63.1%	2.6%
	sports	7.5%	91.1%	1.5%
	comprehensive	-	100%	0

Table 5: Error analysis of BINet-based approaches.

We conducted an ablation test to study the effects of features on generating summaries in our model. Table 4 shows the performance of models using various feature combination on generating disaster summaries. In both NodeRank and AreaRank models, PageRank features enhance the models that only use word features of nodes, demonstrating the effects of global importance analysis on the stream level. Context features are also useful for improving the results because words (both burst and non-burst words) in context can help the model learn the preference of topics and styles from the reference summary.

We conducted error analysis for NodeRank and AreaRank, shown in Table 5. Among topically irrelevant, minor and redundant event entries, minor (i.e.,

Model	Module	Run time	Can be run in parallel
BINet	burst detection	14ms per word	Yes
	BINet construction	213.88s on 1-year news	Partially
	PageRank	1.36s per iteration	No
	Ranking	negligible	No
Ge et al. (2015b)	Event detection	1,018s per iteration	No
	Ranking	negligible	No

Table 6: Run time of BINet-based approaches and Ge et al. (2015b)'s approach

trivial) event entries that are not important enough to be included in the stream summary account for the majority of errors for both models. This is because it is difficult to distinguish these trivial events since the corpus we used as a text stream is not as ideal as the assumption that the more important events, the more times they are reported. As shown in Table 2, many entries in the reference summaries even do not appear or burst in our corpus because the Gigaword corpus used is just a small sample of news stream during the period. As a result, the importance features (e.g., PageRank value) in our ranking model do not work very well for distinguishing trivial events.

At last, we tested the run time of our BINet approach and compare to the state-of-the-art model proposed by Ge et al. (2015b) in terms of efficiency. The results are shown in Table 6. The run time is tested on a workstation with Intel Xeon 3.5 GHz CPU and 64GB RAM. The efficiency of our model is much better than Ge et al. (2015b)'s approach whose event detection model takes much time to iterate thousands of times for Gibbs sampling. For memory cost, the peak memory cost of our BINet-based approaches is 5GB while Ge et al. (2015b)'s approach needs more than 10GB memory to run the event detection model and thus cannot work on a large dataset.

5.2 Experiments on a real-time news stream

To evaluate our approaches in a real setting, we create a benchmark dataset[4] containing 7.9 million English news stories (without exact duplication) during Feb 5 to Mar 31, 2015, collecting from Bing news portal[5]. On average, there are approximately 150,000 news documents per day.

We applied our BINet-based approaches (i.e., NodeRank and AreaRank) on the real-time stream. Specifically, we used news stream during Feb 5 to Mar 23 for training to generate news summaries for every day during Mar 24 to Mar 30 in an online fashion. This is a practical setting and can be useful for automatically generating headline news every day.

Models	Disaster	Attack
Random	0.012	0.019
Online-B-HAC	0.096	0.138
NodeRank	0.111	0.153
AreaRank	**0.182**	**0.157**

Table 7: MRR of BINet-based approaches on generating summaries for the real-time news stream.

Daily news summaries in *Current Event Portal*[6] at Wikipedia are used as reference summaries for training and gold standard for evaluating our approaches. In this paper, we tested on generating summaries on *Disaster and accident* (Disaster) and *Armed conflicts and attacks* (Attack) topics. Instead of evaluating *Precision@K* as we did on the Gigaword corpus which is a small dataset, we used *Mean Reciprocal Rank (MRR)* which is defined as follows to see the ranking position of event entries of the gold standard in the summaries generated by our approaches:

$$MRR = \frac{\sum_{t \in T_{test}} (\sum_{\boldsymbol{e_k} \in \mathcal{E}^{(t)}_{gold}} \frac{1}{rank^{(t)}_{\boldsymbol{e_k}}})}{\sum_{t \in T_{test}} |\mathcal{E}^{(t)}_{gold}|} \quad (5)$$

where $\mathcal{E}^{(t)}_{gold}$ is the gold standard summaries at time t, T_{test} is the period of test set (i.e., Mar 24 to Mar 30) and $rank^{(t)}_{\boldsymbol{e_k}}$ is the highest rank of an event entry $\boldsymbol{e_k}$ of the gold standard summary in our summary at t. A high MRR means the event entries of gold standard tend to be ranked at top positions in our generated summaries. The evaluation is conducted manually.

Table 7 shows the performance of BINet-based

[4]The dataset and the gold standard are available at http://getao.github.io

[5]https://www.bing.com/news

[6]https://en.wikipedia.org/wiki/Portal:Current_events/

approaches on the real-time news stream. The BINet-based approaches achieve better results than the online version of B-HAC model on both topics, demonstrating the advantages of the BINet representation. It is also notable that AreaRank performs better than NodeRank because it scores a document area as a whole by taking into account various information of the area. For AreaRank, MRR on the disaster topic is about 0.2, meaning that the average ranking position of gold standard event entries is 5, which is a promising result and shows our approach can be effective to find key information. More importantly, it only takes 500 seconds to build a BINet and 388 seconds to run PageRank for 1,000 iterations for global importance analysis on the 7.9 million documents while other methods in Table 3 even cannot be applied on the stream because they cannot handle so large scale of data or work in an online fashion, which is why we did not compare to them in this setting.

6 Related Work

Stream summarization is not a hot topic in NLP community. Despite the related work that studies corpus summarization of research papers (Sipos et al., 2012), Ge et al. (2015b) is the only work exactly dealing with the news stream summarization challenge. However, they studied the problem on a static timestamped corpus instead of on a dynamic text stream and their proposed pipeline-style approach cannot be applied on a real-time text stream due to high complexity in time and space. Other previous work dealing with stream data is mainly focused on topic and event detection (Yang et al., 1998; Swan and Allan, 2000; Allan, 2002; He et al., 2007; Sayyadi et al., 2009; Sakaki et al., 2010; Zhao et al., 2012; Ge et al., 2015a), dynamic language and topic modelling (Blei and Lafferty, 2006; Iwata et al., 2010; Wang et al., 2012; Yogatama et al., 2014), incremental (temporal) summarization and timeline generation for one major news event (Allan et al., 2001; Hu et al., 2011; Yan et al., 2011; Lin et al., 2012; Li and Li, 2013; Kedzie et al., 2015; Tran et al., 2015; Yao et al., 2016), a sports match (Takamura et al., 2011) or users on the social network (Li and Cardie, 2014).

Different from traditional single and multi-document summarization (Carbonell and Goldstein, 1998; Lin, 2004; Erkan and Radev, 2004; Conroy et al., 2004; Li et al., 2007; Wan and Yang, 2008; Chen and Chen, 2012; Wan and Zhang, 2014) whose focus is to select important sentences, the focus of stream summarization is to select representative documents referring to important news events. The novel paradigm focuses on the summarization problem in the big data age and is useful for many applications.

7 Conclusions and Future work

In this paper, we study the news stream summarization problem by proposing a novel text stream representation – Burst Information Networks and presenting two summarization models based on it. The proposed approaches can efficiently generate high-quality summaries, achieving the state-of-the-art performance. Moreover, the experiments on our created benchmark dataset showed our approach can be effectively applied on the real-time news stream for finding key information, demonstrating its potential values for many real-world applications (e.g., personalized headline news recommendation).

In the future, we plan to generalize the stream summarization problem to various streams such as social (e.g., Twitter), image (e.g., Imgur) and even video streams (e.g., Youtube), which would yield many interesting and practical applications (Lu et al., 2016) to deal with the information overload challenge in the big data era.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. We also want to specially thank Prof. Heng Ji for her valuable suggestions and discussion on the early ideas of this work. This work is supported by the National Key Basic Research Program of China (No.2014CB340504) and the National Natural Science Foundation of China (No.61375074,61273318). The contact author is Zhifang Sui.

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *SIGIR*.

James Allan. 2002. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *ICML*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*.

Chien Chin Chen and Meng Chang Chen. 2012. Tscan: A content anatomy approach to temporal topic summarization. *Knowledge and Data Engineering, IEEE Transactions on*, 24(1):170–183.

John M Conroy, Judith D Schlesinger, Jade Goldstein, and Dianne P Oleary. 2004. Left-brain/right-brain multi-document summarization. In *DUC*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Tao Ge, Wenzhe Pei, Baobao Chang, and Zhifang Sui. 2015a. Distinguishing specific and daily topics. In *APWeb*.

Tao Ge, Wenzhe Pei, Heng Ji, Sujian Li, Baobao Chang, and Zhifang Sui. 2015b. Bring you to the past: Automatic generation of topically relevant event chronicles. In *ACL*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Qi He, Kuiyu Chang, and Ee-Peng Lim. 2007. Using burstiness to improve clustering of topics in news streams. In *ICDM*.

Po Hu, Minlie Huang, Peng Xu, Weichang Li, Adam K Usadi, and Xiaoyan Zhu. 2011. Generating breakpoint-based timeline overview for news topic retrospection. In *ICDM*.

Tomoharu Iwata, Takeshi Yamada, Yasushi Sakurai, and Naonori Ueda. 2010. Online multiscale dynamic topic models. In *KDD*.

Thorsten Joachims. 2006. Training linear svms in linear time. In *SIGKDD*.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Jiwei Li and Claire Cardie. 2014. Timeline generation: Tracking individuals on twitter. In *WWW*.

Jiwei Li and Sujian Li. 2013. Evolutionary hierarchical dirichlet process for timeline summarization. In *ACL*.

Sujian Li, You Ouyang, Wei Wang, and Bin Sun. 2007. Multi-document summarization using support vector regression. In *DUC*. Citeseer.

Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. 2012. Generating event storylines from microblogs. In *CIKM*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Di Lu, Clare Voss, Fangbo Tao, Xiang Ren, Rachel Guan, Rostyslav Korolov, Tongtao Zhang, Dongang Wang, Hongzhi Li, Taylor Cassidy, Heng Ji, Shih-fu Chang, Jiawei Han, William Wallace, James Hendler, Mei Si, and Lance Kaplan. 2016. Cross-media event extraction and recommendation. In *NAACL Demo Session*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*.

Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event detection and tracking in social streams. In *ICWSM*.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Temporal corpus summarization using submodular word coverage. In *CIKM*.

Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *SIGIR*.

Hiroya Takamura, Hikaru Yokono, and Manabu Okumura. 2011. Summarizing a document stream. In *Advances in Information Retrieval*, pages 177–188. Springer.

Giang Tran, Mohammad Alrifai, and Eelco Herder. 2015. Timeline summarization from relevant headlines. In *Advances in Information Retrieval*.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *SIGIR*.

Xiaojun Wan and Jianmin Zhang. 2014. Ctsum: extracting more certain summaries for news articles. In *SIGIR*.

Chong Wang, David Blei, and David Heckerman. 2012. Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *SIGIR*.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *SIGIR*.

Jin-ge Yao, Feifan Fan, Wayne Xin Zhao, Xiaojun Wan, Edward Chang, and Jianguo Xiao. 2016. Tweet timeline generation with determinantal point processes. In *AAAI*.

Dani Yogatama, Chong Wang, Bryan R Routledge, Noah A Smith, and Eric P Xing. 2014. Dynamic language models for streaming text. *Transactions of the Association for Computational Linguistics*, 2:181–192.

Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. 2012. A novel burst-based text representation model for scalable event detection. In *ACL*.

Rationale-Augmented Convolutional Neural Networks for Text Classification

Ye Zhang,[1] **Iain Marshall**,[2] **Byron C. Wallace**[3]
[1]Department of Computer Science, University of Texas at Austin
[2]Department of Primary Care and Public Health Sciences, Kings College London
[3]College of Computer and Information Science, Northeastern University
yezhang@cs.utexas.edu, iain.marshall@kcl.ac.uk
byron@ccs.neu.edu

Abstract

We present a new Convolutional Neural Network (CNN) model for text classification that jointly exploits labels on documents and their constituent sentences. Specifically, we consider scenarios in which annotators explicitly mark sentences (or snippets) that support their overall document categorization, i.e., they provide *rationales*. Our model exploits such supervision via a hierarchical approach in which each document is represented by a linear combination of the vector representations of its component sentences. We propose a sentence-level convolutional model that estimates the probability that a given sentence is a rationale, and we then scale the contribution of each sentence to the aggregate document representation in proportion to these estimates. Experiments on five classification datasets that have document labels and associated rationales demonstrate that our approach consistently outperforms strong baselines. Moreover, our model naturally provides explanations for its predictions.

1 Introduction

Neural models that exploit word embeddings have recently achieved impressive results on text classification tasks (Goldberg, 2015). Feed-forward Convolutional Neural Networks (CNNs), in particular, have emerged as a relatively simple yet powerful class of models for text classification (Kim, 2014).

These neural text classification models have tended to assume a standard supervised learning setting in which instance labels are provided. Here we consider an alternative scenario in which we assume that we are provided a set of *rationales* (Zaidan et al., 2007; Zaidan and Eisner, 2008; McDonnell et al., 2016) in addition to instance labels, i.e., sentences or snippets that support the corresponding document categorizations. Providing such rationales during manual classification is a natural interaction for annotators, and requires little additional effort (Settles, 2011; McDonnell et al., 2016). Therefore, when training new classification systems, it is natural to acquire supervision at both the document and sentence level, with the aim of inducing a better predictive model, potentially with less effort.

Learning algorithms must be designed to capitalize on these two types of supervision. Past work (Section 2) has introduced such methods, but these have relied on linear models such as Support Vector Machines (SVMs) (Joachims, 1998), operating over sparse representations of text. We propose a novel CNN model for text classification that exploits both document labels and associated rationales.

Specific contributions of this work as follows. (1) This is the first work to incorporate rationales into neural models for text classification. (2) Empirically, we show that the proposed model uniformly outperforms relevant baseline approaches across five datasets, including previously proposed models that capitalize on rationales (Zaidan et al., 2007; Marshall et al., 2016) and multiple baseline CNN variants, including a CNN equipped with an attention mechanism. We also report state-of-the-art results on the important task of automatically assessing the risks of bias in the studies described in full-text biomedical articles (Marshall et al., 2016). (3) Our model naturally provides *explanations* for its predic-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 795–804,
Austin, Texas, November 1-5, 2016.

tions, providing interpretability.

We have made available online both a Theano[1] and a Keras implementation[2] of our model.

2 Related Work

2.1 Neural models for text classification

Kim (2014) proposed the basic CNN model we describe below and then build upon in this work. Properties of this model were explored empirically in (Zhang and Wallace, 2015). We also note that Zhang et al. (2016) extended this model to jointly accommodate multiple sets of pre-trained word embeddings. Roughly concurrently to Kim, Johnson and Zhang (2014) proposed a similar CNN architecture, although they swapped in one-hot vectors in place of (pre-trained) word embeddings. They later developed a semi-supervised variant of this approach (Johnson and Zhang, 2015).

In related recent work on Recurrent Neural Network (RNN) models for text, Tang et al. (2015) proposed using a Long Short Term Memory (LSTM) layer to represent each sentence and then passing another RNN variant over these. And Yang et al. (2016) proposed a hierarchical network with two levels of attention mechanisms for document classification. We discuss this model specifically as well as attention more generally and its relationship to our proposed approach in Section 4.3.

2.2 Exploiting *rationales*

In long documents the importance of sentences varies; some are more central than others. Prior work has investigated methods to measure the relative importance sentences (Ko et al., 2002; Murata et al., 2000). In this work we adopt a particular view of sentence importance in the context of document classification. In particular, we assume that documents comprise sentences that directly support their categorization. We call such sentences *rationales*.

The notion of rationales was first introduced by Zaidan et al. (2007). To harness these for classification, they proposed modifying the Support Vector Machine (SVM) objective function to encode a preference for parameter values that result in instances containing manually annotated rationales being more confidently classified than 'pseudo'-instances from which these rationales had been stripped. This approach dramatically outperformed baseline SVM variants that do not exploit such rationales. Yessenalina et al. (2010) later developed an approach to *generate* rationales.

Another line of related work concerns models that capitalize on *dual supervision*, i.e., labels on individual *features*. This work has largely involved inserting constraints into the learning process that favor parameter values that align with *a priori* feature-label affinities or rankings (Druck et al., 2008; Mann and McCallum, 2010; Small et al., 2011; Settles, 2011). We do not discuss this line of work further here, as our focus is on exploiting provided rationales, rather than individual labeled features.

3 Preliminaries: CNNs for text classification

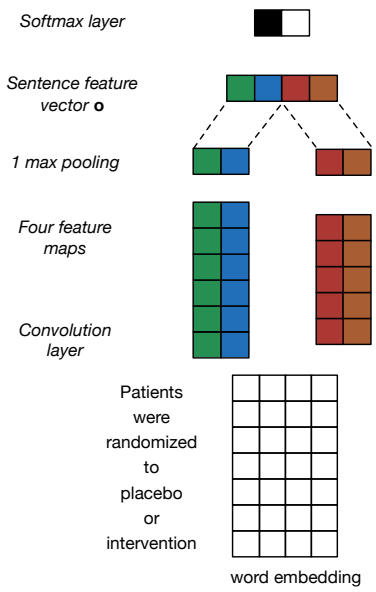


Figure 1: A toy example of a CNN for sentence classification. Here there are four filters, two with heights 2 and two with heights 3, resulting in feature maps with lengths 6 and 5 respectively.

We first review the simple one-layer CNN for sentence modeling proposed by Kim (2014). Given a sentence or document comprising n words $w_1, w_2, ..., w_n$, we replace each word with its d-dimensional pretrained embedding, and stack them row-wise, generating an instance matrix $A \in \mathbb{R}^{n \times d}$.

[1] https://github.com/yezhang-xiaofan/Rationale-CNN
[2] https://github.com/bwallace/rationale-CNN

We then apply convolution operations on this matrix using multiple linear filters, these will have the same width d but may vary in *height*. Each filter thus effectively considers distinct n-gram features, where n corresponds to the filter height. In practice, we introduce multiple, redundant features of each height; thus each filter height might have hundreds of corresponding instantiated filters. Applying filter i parameterized by $W_i \in \mathbb{R}^{h_i \cdot d}$ to the instance matrix induces a *feature map* $f_i \in \mathbb{R}^{n-h_i+1}$. This process is performed by sliding the filter from the top of the matrix (the start of the document or sentence) to the bottom. At each location, we apply element-wise multiplication between filter i and sub-matrix $A[j : j + h_i - 1]$, and then sum up the resultant matrix elements. In this way, we induce a vector (feature map) for each filter.

We next run the feature map through an element-wise non-linear transformation. Specifically, we use the *Rectified Linear Unit*, or ReLU (Krizhevsky et al., 2012). We extract the maximum value o_i from each feature map i (*1-max pooling*).

Finally, we concatenate all of the features o_i to form a vector representation $\mathbf{o} \in \mathbb{R}^{|F|}$ for this instance, where $|F|$ denotes the total number of filters. Classification is then performed on top of $\mathbf{o}$, via a softmax function. Dropout (Srivastava et al., 2014) is often applied at this layer as a means of regularization. We provide an illustrative schematic of the basic CNN architecture just described in Figure 1. For more details, see (Zhang and Wallace, 2015).

This model was originally proposed for sentence classification (Kim, 2014), but we can adapt it for document classification by simply treating the document as one long sentence. We will refer to this basic CNN variant as ***CNN*** in the rest of the paper. Below we consider extensions that account for document structure.

4 Rationale-Augmented CNN for Document Classification

We now move to the main contribution of this work: a rationale-augmented CNN for text classification. We first introduce a simple variant of the above CNN that models document structure (Section 4.1) and then introduce a means of incorporating rationale-level supervision into this model (Section 4.2). In Section 4.3 we discuss connections to attention mechanisms and describe a baseline equipped with one, inspired by Yang et al. (2016).

4.1 Modeling Document Structure

Recall that rationales are snippets of text marked as having supported document-level categorizations. We aim to develop a model that can exploit these annotations during training to improve classification. Here we achieve this by developing a hierarchical model that estimates the probabilities of individual sentences being rationales and uses these estimates to inform the document level classification.

As a first step, we extend the CNN model above to explicitly account for document structure. Specifically, we apply a CNN to each individual sentence in a document to obtain sentence vectors independently. We then sum the respective sentence vectors to create a document vector.[3] As before, we add a softmax layer on top of the document-level vector to perform classification. We perform regularization by applying dropout both on the individual sentence vectors and the final document vector. We will refer to this model as ***Doc-CNN***. Doc-CNN forms the basis for our novel approach, described below.

4.2 RA-CNN

In this section we present the Rationale-Augmented CNN (***RA-CNN***). Briefly, RA-CNN induces a document-level vector representation by taking a *weighted* sum of its constituent sentence vectors. Each sentence weight is set to reflect the estimated probability that it is a rationale in support of the most likely class. We provide a schematic of this model in Figure 2.

RA-CNN capitalizes on both sentence- and document-level supervision. There are thus two steps in the training phase: *sentence level* training and *document level* training. For the former, we apply a CNN to each sentence j in document i to obtain sentence vectors $\mathbf{x}_{\text{sen}}^{ij}$. We then add a softmax layer parametrized by $\mathbf{W}_{\text{sen}}$; this takes as input sentence vectors. We fit this model to maximize the probabilities of the observed rationales:

[3] We also experimented with taking the average of sentence vectors, but summing performed better in informal testing.

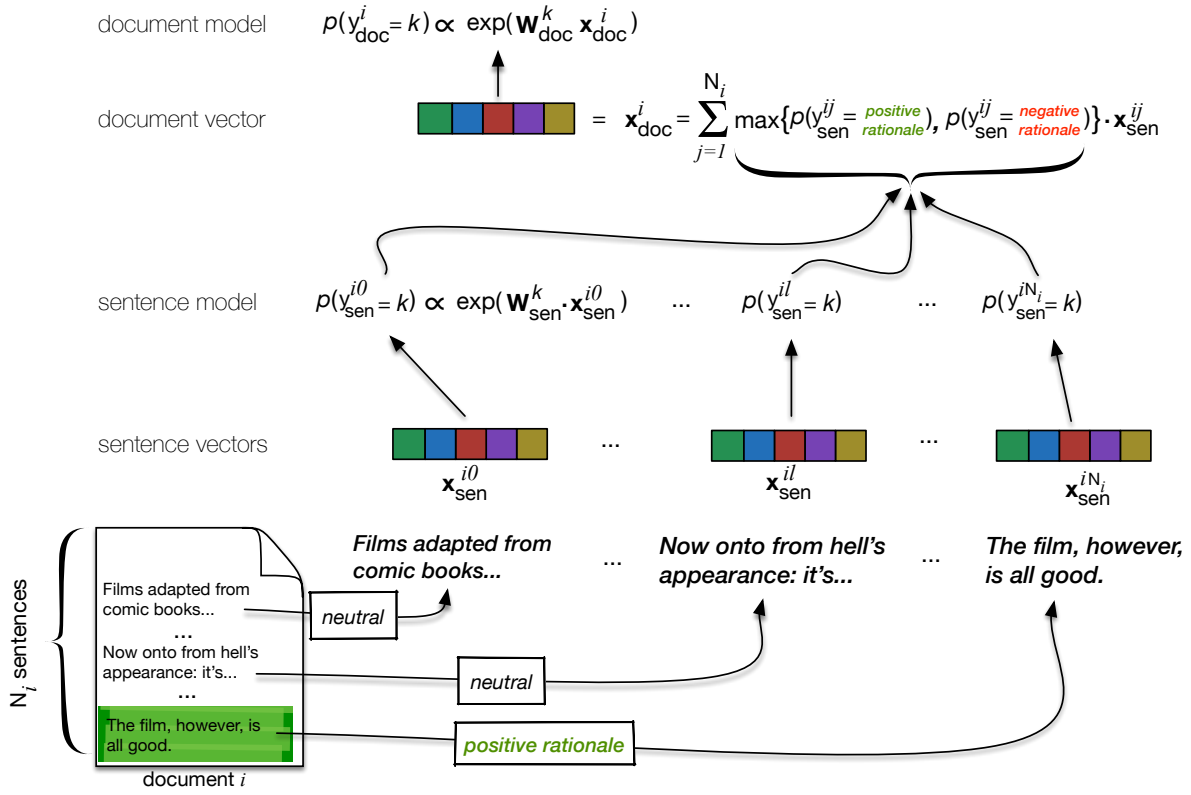


Figure 2: A schematic of our proposed Rationale-Augmented Convolution Neural Network (RA-CNN). The sentences comprising a text are passed through a sentence model that outputs probabilities encoding the likelihood that sentences are neutral or a (positive or negative) rationale. Sentences likely to be rationales are given higher weights in the global document vector, which is the input to the document model.

$$p(y^{ij}_{\text{sen}} = k; \mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{sen}}) = \frac{\exp(\mathbf{W}^{(k)T}_{\text{sen}} \mathbf{x}^{ij}_{\text{sen}})}{\sum_{k=1}^{K_{\text{sen}}} \exp(\mathbf{W}^{(k)T}_{\text{sen}} \mathbf{x}^{ij}_{\text{sen}})} \quad (1)$$

Where y^{ij}_{sen} denotes the rationale label for sentence j in document i, K_{sen} denotes the number of possible classes for sentences, $\mathbf{E}$ denotes the word embedding matrix, $\mathbf{C}$ denotes the convolution layer parameters, and $\mathbf{W}_{\text{sen}}$ is a matrix of weights (comprising one weight vector per sentence class).

In our setting, each sentence has three possible labels ($K_{\text{sen}} = 3$). When a rationale sentence appears in a positive document,[4] it is a *positive rationale*; when a rationale sentence appears in a negative document, it is a *negative rationale*. All other sentences belong to a third, neutral class: these are *non-rationales*. We also experimented with having only two sentence classes: *rationales* and *non-rationales*, but this did not perform as well as explicitly maintaining separate classes for rationales of different polarities.

We train an estimator using the provided rationale annotations, optimizing over $\{\mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{sen}}\}$ to minimize the categorical cross-entropy of sentence labels. Once trained, this sub-model can provide conditional probability estimates regarding whether a given sentence is a positive or a negative rationale, which we will denote by p_{pos} and p_{neg}, respectively.

We next train the document-level classification model. The inputs to this are vector representations of documents, induced by summing over constituent sentence vectors, as in Doc-CNN. However, in the RA-CNN model this is a weighted sum. Specifically, weights are set to the estimated probabilities

[4] All of the document classification tasks we consider here are binary, although extension of our model to multi-class scenarios is straight-forward.

that corresponding sentences are rationales in the most likely direction. More precisely:

$$\mathbf{x}^i_{\text{doc}} = \sum_{j=1}^{N_i} \mathbf{x}^{ij}_{\text{sen}} \cdot \max\{p^{ij}_{\text{pos}}, p^{ij}_{\text{neg}}\} \quad (2)$$

Where N_i is the number of sentences in the ith document. The intuition is that sentences likely to be rationales will have greater influence on the resultant document vector representation, while the contribution of neutral sentences (which are less relevant to the classification task) will be minimized.

The final classification is performed by a softmax layer parameterized by $\mathbf{W}_{\text{doc}}$; the inputs to this layer are the document vectors. The $\mathbf{W}_{\text{doc}}$ parameters are trained using the document-level labels, y^i_{doc}:

$$p(y^i_{\text{doc}} = k; \mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{doc}}) = \frac{\exp(\mathbf{W}^{(k)T}_{\text{doc}} \mathbf{x}^i_{\text{doc}})}{\sum_{k=1}^{K_{\text{doc}}} \exp(\mathbf{W}^{(k)T}_{\text{doc}} \mathbf{x}^i_{\text{doc}})} \quad (3)$$

where K_{doc} is the cardinality of the document label set. We optimize over parameters to minimize cross-entropy loss (w.r.t. the document labels).

We note that the sentence- and document-level models share word embeddings $\mathbf{E}$ and convolution layer parameters $\mathbf{C}$, but the document-level model has its own softmax parameters $\mathbf{W}_{\text{doc}}$. When training the document-level model, $\mathbf{E}$, $\mathbf{C}$ and $\mathbf{W}_{\text{doc}}$ are fit, but we hold $\mathbf{W}_{\text{sen}}$ fixed.

The above two-step strategy can be equivalently described as follows. We first estimate $\mathbf{E}$, $\mathbf{C}$ and $\mathbf{W}_{\text{sen}}$, which parameterize our model for identifying rationales in documents. We then move to fitting our document classification model. For this we initialize the word embedding and convolution parameters to the $\mathbf{E}$ and $\mathbf{C}$ estimates from the preceding step. We then directly minimize the document level classification objective, tuning $\mathbf{E}$ and $\mathbf{C}$ and simultaneously fitting $\mathbf{W}_{\text{doc}}$.

Note that this sequential training strategy differs from the *alternating* training approach commonly used in multi-task learning (Collobert and Weston, 2008). We found that the latter approach does not work well here, leading us to instead adopt the cascade-like feature learning approach (Collobert and Weston, 2008) just described.

One nice property of our model is that it naturally provides explanations for its predictions: the model identifies rationales and then categorizes documents informed by these. Thus if the model classifies a test instance as *positive*, then by construction the sentences associated with the highest p^{ij}_{pos} estimates are those that the model relied on most in coming to this disposition. These sentences can of course be output in conjunction with the prediction. We provide concrete examples of this in Section 7.2.

4.3 Rationales as 'Supervised Attention'

One may view RA-CNN as a supervised variant of a model equipped with an *attention mechanism* (Bahdanau et al., 2014). On this view, it is apparent that rather than capitalizing on rationales directly, we could attempt to let the model learn which sentences are important, using *only* the document labels. We therefore construct an additional baseline that does just this, thereby allowing us to assess the impact of learning directly from rationale-level supervision.

Following the recent work of Yang et al. (2016), we first posit for each sentence vector a hidden representation $\mathbf{u}^{ij}_{\text{sen}}$. We then define a sentence-level *context vector* $\mathbf{u}_s$, which we multiply with each $\mathbf{u}^{ij}_{\text{sen}}$ to induce a weight α_{ij}. Finally, the document vector is taken as a weighted sum over sentence vectors, where weights reflect α's. We have:

$$\mathbf{u}^{ij}_{\text{sen}} = \tanh(\mathbf{W}_s \mathbf{x}^{ij}_{\text{sen}} + b_s) \quad (4)$$

$$\alpha_{ij} = \frac{\exp(\mathbf{u}_s^T \mathbf{u}^{ij}_{\text{sen}})}{\sum_j^{N_i} \exp(\mathbf{u}_s^T \mathbf{u}^{ij}_{\text{sen}})} \quad (5)$$

$$\mathbf{x}^i_{\text{doc}} = \sum_j^{N_i} \alpha_{ij} \mathbf{x}^{ij}_{\text{sen}} \quad (6)$$

where $\mathbf{x}^i_{\text{doc}}$ again denotes the document vector fed into a softmax layer, and $\mathbf{W}_s$, $\mathbf{u}_s$ and b_s are learned during training. We will refer to this attention-based method as *AT-CNN*.

5 Datasets

We used five text classification datasets to evaluate our approach in total. Four of these are biomedical text classification datasets (5.1) and the last is a collection of movie reviews (5.2). These datasets share the property of having recorded rationales associated

with each document categorization. We summarize attributes of all datasets used in this work in Table 1.

5.1 Risk of Bias (RoB) Datasets

We used a collection *Risk of Bias* (RoB) text classification datasets, described at length elsewhere (Marshall et al., 2016). Briefly, the task concerns assessing the reliability of the evidence presented in full-text biomedical journal articles that describe the conduct and results of randomized controlled trials (RCTs). This involves, e.g., assessing whether or not patients were properly blinded as to whether they were receiving an active treatment or a comparator (such as a placebo). If such blinding is not done correctly, it compromises the study by introducing statistical bias into the treatment efficacy estimate(s) derived from the trial.

A formal system for making bias assessments is codified by the Cochrane Risk of Bias Tool (Higgins et al., 2011). This tool defines multiple *domains*; the risk of bias may be assessed in each of these. We consider four domains here. (1) Random sequence generation (RSG): were patients were assigned to treatments in a truly random fashion? (2) Allocation concealment (AC): were group assignments revealed to the person assigning patients to groups (so that she may have knowingly or unknowingly) influenced these assignments? (3) Blinding of Participants and Personnel (BPP): were all trial participants and individuals involved in running the trial blinded as to who was receiving which treatment? (4) Blinding of outcome assessment (BOA): were the parties who measured the outcome(s) of interest blinded to the intervention group assignments? These assessments are somewhat subjective. To increase transparency, researchers performing RoB assessment therefore record rationales (sentences from articles) supporting their assessments.

5.2 Movie Review Dataset

We also ran experiments on a movie review (MR) dataset with accompanying rationales. Pang and Lee (2004) developed and published the original version of this dataset, which comprises 1000 positive and 1000 negative movie reviews from the Internet Movie Database (IMDB).[5] Zaidan et al. (2007) then augmented this dataset by adding rationales corresponding to the binary classifications for 1800 documents, leaving the remaining 200 for testing. Because 200 documents is a modest test sample size, we ran 9-fold cross validation on the 1800 annotated documents (each fold comprising 200 documents). The rationales, as originally marked in this dataset, were sub-sentential snippets; for the purposes of our model, we considered the entire sentences containing the marked snippets as rationales.

[5] http://www.imdb.com/

	N	*#sen*	*#token*	*#rat*
RSG	8399	300	9.92	0.31
AC	11512	297	9.87	0.15
BPP	7997	296	9.95	0.21
BOA	2706	309	9.92	0.2
MR	1800	32.6	21.2	8.0

Table 1: Dataset characteristics. *N* is the number of instances, *#sen* is the average sentence count, *#token* is the average token per-sentence count and *#rat* is the average number of rationales per document.

6 Experimental Setup

6.1 Baselines

We compare against several baselines to assess the advantages of directly incorporating rationale-level supervision into the proposed CNN architecture. We describe these below.

SVMs. We evaluated a few variants of linear Support Vector Machines (SVMs). These rely on sparse representations of text. We consider variants that exploit uni- and bi-grams; we refer to these as *uni-SVM* and *bi-SVM*, respectively. We also re-implemented the rationale augmented SVM (*RA-SVM*) proposed by Zaidan et al. (2007), described in Section 2.

For the RoB dataset, we also compare to a recently proposed multi-task SVM (MT-SVM) model developed specifically for these RoB datasets (Marshall et al., 2015; Marshall et al., 2016). This model exploits the intuition that the risks of bias across the domains codified in the aforementioned Cochrane RoB tool will likely be correlated. That is, if we know that a study exhibits a high risk of bias for one domain, then it seems reasonable to assume it is at an elevated risk for the remaining domains. Furthermore, Marshall et al. (2016) include rationale-level supervision by first training a (multi-task) *sentence-level* model to identify sentences likely to support

RoB assessments in the respective domains. Special features extracted from these predicted rationales are then activated in the *document-level* model, informing the final classification. This model is the state-of-the-art on this task.

CNNs. We compare against several baseline CNN variants to demonstrate the advantages of our approach. We emphasize that our focus in this work is **not** to explore how to induce generally 'better' document vector representations – this question has been addressed at length elsewhere, e.g., (Le and Mikolov, 2014; Jozefowicz et al., 2015; Tang et al., 2015; Yang et al., 2016).

Rather, the main contribution here is an augmentation of CNNs for text classification to capitalize on rationale-level supervision, thus improving performance and enhancing interpretability. This informed our choice of baseline CNN variants: standard CNN (Kim, 2014), Doc-CNN (described above) and AT-CNN (also described above) that capitalizes on an (unsupervised) attention mechanism at the sentence level, described in Section 4.3.[6]

6.2 Implementation/Hyper-Parameter Details

Sentence splitting. To split the documents from all datasets into sentences for consumption by our Doc-CNN and RA-CNN models, we used the Natural Language Toolkit (NLTK)[7] sentence splitter.

SVM-based models. We kept the 50,000 most frequently occurring features in each dataset. For estimation we used SGD. We tuned the C hyper-parameter using nested development sets. For the RA-SVM, we additionally tuned the μ and $C_{contrast}$ parameters, as per Zaidan et al. (2007).

CNN-based models. For all models and datasets we initialized word embeddings to pre-trained vectors fit via Word2Vec. For the movie reviews dataset these were 300-dimensional and trained on Google News.[8] For the RoB datasets, these were 200-dimensional and trained on biomedical texts in PubMed/PubMed Central (Pyysalo et al., 2013).[9]

[6]We also experimented briefly with LSTM and GRU (Gated Recurrent Unit) models, but found that simple CNN performed better than these. Moreover, CNNs are relatively robust and less sensitive to hyper-parameter selection.

[7]http://www.nltk.org/api/nltk.tokenize.html

[8]https://code.google.com/archive/p/word2vec/

[9]http://bio.nlplab.org/

Training proceeded as follows. We first extracted all sentences from all documents in the training data. The distribution of sentence types is highly imbalanced (nearly all are neutral). Therefore, we downsampled sentences before each epoch, so that sentence classes were equally represented. After training on sentence-level supervision, we moved to document-level model fitting. For this we initialized embedding and convolution layer parameters to the estimates from the preceding sentence-level training step (though these were further tuned to optimize the document-level objective).

For RA-CNN, we tuned the dropout rate (range: 0-.9) applied at the sentence vector level on each training fold (using a subset of the training data as a validation set) during the document level training phase. Anecdotally, we found this has a greater effect than the other model hyperparameters, which we thus set after a small informal process of experimentation on a subset of the data. Specifically, we fixed the dropout rate at the document level to 0.5, and we used 3 different filter heights: 3, 4 and 5, following (Zhang and Wallace, 2015). For each filter height, we used 100 feature maps for the baseline CNN, and 20 for all the other CNN variants.

For parameter estimation we used ADADELTA (Zeiler, 2012), mini-batches of size 50, and an early stopping strategy (using a validation set).

7 Results and Discussion

7.1 Quantitative Results

For all CNN models, we replicated experiments 5 times, where each replication constituted 5-fold and 9-fold CV respectively the RoB and the movies datasets, respectively. We report the mean and observed ranges in accuracy across these 5 replications for these models, because attributes of the model (notably, dropout) and the estimation procedure render model fitting stochastic (Zhang and Wallace, 2015). We do not report ranges for SVM-based models because the variance inherent in the estimation procedure is much lower for these simpler, linear models.

Results on the RoB datasets and the movies dataset are shown in Tables 2 and Table 3, respectively. RA-CNN consistently outperforms all of the baseline models, across all five datasets. We also

Method	RSG	AC	BPP	BOA
Uni-SVM	72.16	72.81	72.80	65.85
Bi-SVM	74.82	73.62	75.13	67.29
RA-SVM	72.54	74.11	75.15	66.29
MT-SVM	76.15	74.03	76.33	67.50
CNN	72.50 (72.22, 72.65)	72.16 (71.49, 72.93)	75.03 (74.16, 75.44)	63.76 (63.12, 64.15)
Doc-CNN	72.60 (72.43, 72.90)	72.92 (72.19, 73.48)	74.24 (74.03, 74.38)	63.64 (63.23, 64.37)
AT-CNN	74.14 (73.40, 74.58)	73.66 (73.12, 73.92)	74.29 (74.09, 74.74)	63.34 (63.21, 63.49)
RA-CNN	**77.42 (77.33, 77.59)**	**76.14 (75.89, 76.29)**	**76.47 (76.15, 76.75)**	**69.67 (69.33, 69.93)**
Human	85.00	80.00	78.10	83.20

Table 2: Accuracies on the four RoB datasets. Uni-SVM: unigram SVM, Bi-SVM: Bigram SVM, RA-SVM: Rationale-augmented SVM (Zaidan et al., 2007), MT-SVM: a multi-task SVM model specifically designed for the RoB task, which also exploits the available sentence supervision (Marshall et al., 2016). We also report an estimate of human-level performance, as calculated using subsets of the data for each domain that were assessed by two experts (one was arbitrarily assumed to be correct). We report these numbers for reference; they are not directly comparable to the cross-fold estimates reported for the models.

observe that CNN/Doc-CNN do not necessarily improve over the results achieved by SVM-based models, which prove to be strong baselines for longer document classification. This differs from previous comparisons in the context of classifying shorter texts. In particular, in previous work (Zhang and Wallace, 2015) we observed that CNN outperforms SVM uniformly on sentence classification tasks (the average sentence-length in these datasets was about 10). In contrast, in the datasets we consider in the present paper, documents often comprise hundreds of sentences, each in turn containing multiple words. We believe that it is in these cases that explicitly modeling which sentences are most important will result in the greatest performance gains, and this aligns with our empirical results.

Another observation is that AT-CNN does often improve performance over vanilla variants of CNN (i.e., without attention), especially on the RoB datasets, probably because these comprise longer documents. However, as one might expect, RA-CNN clearly outperforms AT-CNN by exploiting rationale-level supervision directly. And by exploiting rationale information directly, RA-CNN is able to consistently perform better than baseline CNN and SVM model variants. Indeed, we find that RA-CNN outperformed MT-SVM on all of the RoB datasets, and this was accomplished without exploiting cross-domain correlations (i.e., without multi-task learning).

Method	Accuracy
Uni-SVM	86.44
Bi-SVM	86.94
RA-SVM	88.89
CNN	85.59 (85.27, 86.17)
Doc-CNN	87.14 (86.70, 87.60)
AT-CNN	86.69 (86.28, 87.17)
RA-CNN	**90.43 (90.11, 91.00)**

Table 3: Accuracies on the movie review dataset.

7.2 Qualitative Results: Illustrative Rationales

In addition to realizing superior classification performance, RA-CNN also provides *explainable* categorizations. The model can provide the highest scoring rationales (ranked by $\max\{p_{\text{pos}}, p_{\text{neg}}\}$) for any given target instance, which in turn – by construction – are those that most influenced the final document classification.

For example, a sample positive rationale supporting a correct designation of a study as being at low risk of bias with respect to blinding of outcomes assessment reads simply *The study was performed double blind*. An example rationale extracted for a study (correctly) deemed at high risk of bias, meanwhile, reads *as the present study is retrospective, there is a risk that the woman did not properly recall how and what they experienced ....*

Turning to the movie reviews dataset, an example rationale extracted from a glowing review of 'Goodfellas' (correctly classified as positive) reads *this cinematic gem deserves its rightful place among the best films of 1990s*. While a rationale extracted

from an unfavorable review of 'The English Patient' asserts that *the only redeeming qualities about this film are the fine acting of Fiennes and Dafoe and the beautiful desert cinematography*.

In each of these cases, the extracted rationales directly support the respective classifications. This provides direct, meaningful insight into the automated classifications, an important benefit for neural models, which are often seen as opaque.

8 Conclusions

We developed a new model (RA-CNN) for text classification that extends the CNN architecture to directly exploit *rationales* when available. We showed that this model outperforms several strong, relevant baselines across five datasets, including vanilla and hierarchical CNN variants, and a CNN model equipped with an attention mechanism. Moreover, RA-CNN automatically provides explanations for classifications made at test time, thus providing interpretability.

Moving forward, we plan to explore additional mechanisms for exploiting supervision at lower levels in neural architectures. Furthermore, we believe an alternative approach may be a hybrid of the AT-CNN and RA-CNN models, wherein an auxiliary loss might be incurred when the attention mechanism output disagrees with the available direct supervision on sentences.

Acknowledgments

Research reported in this article was supported by the National Library of Medicine (NLM) of the National Institutes of Health (NIH) under award number R01LM012086. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This work was also made possible by the support of the Texas Advanced Computer Center (TACC) at UT Austin.

References

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Julian PT Higgins, Douglas G Altman, Peter C Gøtzsche, Peter Jüni, David Moher, Andrew D Oxman, Jelena Savović, Kenneth F Schulz, Laura Weeks, and Jonathan AC Sterne. 2011. The cochrane collaborations tool for assessing risk of bias in randomised trials. *Bmj*, 343:d5928.

Thorsten Joachims. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems (NIPs)*, pages 919–927.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Youngjoong Ko, Jinwoo Park, and Jungyun Seo. 2002. Automatic text categorization using the importance of sentences. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Gideon S Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *The Journal of Machine Learning Research*, 11:955–984.

Iain J Marshall, Joël Kuiper, and Byron C Wallace. 2015. Automating risk of bias assessment for clinical trials. *Biomedical and Health Informatics, IEEE Journal of*, 19(4):1406–1412.

Iain J Marshall, Joël Kuiper, and Byron C Wallace. 2016. Robotreviewer: evaluation of a system for automatically assessing bias in clinical trials. *Journal of the American Medical Informatics Association*, 23(1):193–201.

Tyler McDonnell, Matthew Lease, Tamer Elsayad, and Mucahid Kutlu. 2016. Why Is That Relevant? Collecting Annotator Rationales for Relevance Judgments. In *Proceedings of the 4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*. 10 pages.

Masaki Murata, Qing Ma, Kiyotaka Uchimoto, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. 2000. Japanese probabilistic information retrieval using location and category information. In *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 81–88. ACM.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. *Proceedings of Languages in Biology and Medicine*.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478. Association for Computational Linguistics.

Kevin Small, Byron Wallace, Thomas Trikalinos, and Carla E Brodley. 2011. The constrained weight space svm: learning with ranked features. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 865–872.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ainur Yessenalina, Yejin Choi, and Claire Cardie. 2010. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 336–341. Association for Computational Linguistics.

Omar F Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 31–40. Association for Computational Linguistics.

Omar Zaidan, Jason Eisner, and Christine D Piatko. 2007. Using" annotator rationales" to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. Citeseer.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527, San Diego, California, June. Association for Computational Linguistics.

Transferring User Interests Across Websites with Unstructured Text for Cold-Start Recommendation

Yu-Yang Huang and **Shou-De Lin**
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
{r02922050,sdlin}@csie.ntu.edu.tw

Abstract

In this work, we investigate the possibility of cross-website transfer learning for tackling the cold-start problem. To address the cold-start issues commonly present in a collaborative filtering (CF) system, most existing cross-domain CF models require auxiliary rating data from another domain; nevertheless, under the cross-website scenario, such data is often unobtainable. Therefore, we propose the nearest-neighbor transfer matrix factorization (NT-MF) model, where a topic model is applied to the unstructured user-generated content in the source domain, and the similarity between users in the latent topic space is utilized to guide the target-domain CF model. Specifically, the latent factors of the nearest-neighbors are regarded as a set of pseudo observations, which can be used to estimate the unknown parameters in the model. Improvement over previous methods, especially for the cold-start users, is demonstrated with experiments on a real-world cross-website dataset.

1 Introduction

While collaborative filtering (CF) approaches are one of the most successful methods for building recommender systems, their performance deteriorates dramatically under *cold-start* situations. That is, low prediction accuracy is observed for users/items with very few ratings. Content-based recommender systems may also suffer from the cold-start problem. For instance, content-based nearest-neighbor models (Pazzani and Billsus, 2007) might not be as effective if some users contain too few information to generate a meaningful set of neighbors.

Two types of solutions have been proposed to address the cold-start problem. The first is to create hybrid recommendation models that impose a content-based model on a CF model to enrich the information for users/items with sparse rating profiles (Burke, 2002; Burke, 2007). The second is to transfer the information from auxiliary domains as a remedy to the cold-start individuals (Deng et al., 2015). This paper aims at bringing a marriage between these two types of strategies.

Although transfer learning gradually gains popularity in handling the cold-start issue (Roy et al., 2012), most of them assume a homogeneous model where observations in both domains are of the same type. That is, to transfer knowledge to a rating-based/text-based recommender system, the source system must also be rating-based/text-based. Some earlier works even require the ratings from both domains to be in the same format (Li et al., 2009), or assume specific structured text, such as user-provided tags (Shi et al., 2011; Deng et al., 2015). In this work, by contrast, no source-domain ratings are available and unstructured user-generated content is treated as the auxiliary data. We propose a *heterogeneous transfer learning* framework to utilize unstructured auxiliary text for a better target-domain CF model.

As there is no single service satisfying all social needs, users nowadays hold multiple accounts across many websites. Furthermore, the account linking mechanism is often available on these websites. This allows a precise mapping between the accounts of the same user to be built. One major application of our approach is to improve the recom-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 805–814,
Austin, Texas, November 1-5, 2016.

mendation quality in the target service using auxiliary data obtained from another seemingly irrelevant service.

For instance, consider a new user on YouTube. The initial recommended videos for this user is likely to be irrelevant as there is very few information available. However, with the account linking mechanism, YouTube accounts can be linked to Twitter accounts with a simple click. Our goal is to utilize the content generated by this user on Twitter, despite the possibility that the content is irrelevant to their preference on video browsing, to produce a better video recommendation list on YouTube.

Seemingly intuitive, there exist some difficulties in such *cross-website transfer learning* approach. The biggest challenge lies in the fact that most users do not use multiple services (e.g. social media sites) for the same purpose. Usually a user registers for multiple services because each of them serves its own purpose. As a result, we cannot assume the existence of direct mentions about target-domain items in the source-domain text data. For example, a regular YouTube viewer does not necessarily tweet about the videos he/she has viewed. Thus simple methods such as keyword matching are likely to fail. The same reasoning also implies that, when transferring knowledge across websites or services, the assumption of a shared rating format or structured text is overly optimistic. Even websites aiming for the same purpose often violate this assumption, let alone websites of different types. Therefore, we expect that the source and target services contain heterogeneous information (e.g. content vs. rating). In our model, we make a less strong assumption: regardless of the type of information available in each domain, the users that are similar in one domain should have similar taste in the other domain. Thus, instead of directly transfer the content material from source to target domain, we transfer the *similarity* between users, and use it as a guide to improve the CF model in the target domain.

2 LDA-MF Model

We first introduce an intuitive model to realize the above-mentioned idea, and point out several intrinsic weaknesses making it unsuitable for cross-website transfer learning.

Here we rely on the probabilistic matrix factorization (PMF) model as our target-domain CF model. In the PMF model, each user latent factor is modeled (a priori) by a zero-mean Gaussian. To incorporate source-domain information into the target-domain PMF model, for each user i, a topic vector θ_i is extracted from source-domain text content and assigned as the prior mean of this user's PMF latent factor, that is,

$$u_i \sim \mathcal{N}(\theta_i, \lambda_U^{-1} I), \tag{1}$$

where λ_U is the precision parameter and I is the identity matrix. Different from the original PMF model, prior distributions of different user latent factors are no longer identical. For users having similar source-domain topic vectors, their latent factors are expected to be close in the target-domain latent space. Such property allows the similarity between users to be transferred from source domain to the target domain.

With the latent Dirichlet allocation (LDA) (Blei et al., 2003) topic model being used, the graphical model is depicted in Figure 1. This model is similar in structure to the recently proposed collaborative topic regression (CTR) (Wang and Blei, 2011) model. The main difference is that, instead of modeling description about items, now user-generated content *from the source domain* is modeled in our problem. We call this model the LDA-MF model.

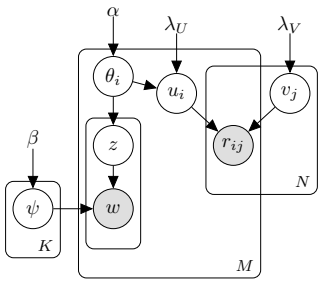


Figure 1: The LDA-MF model.

Although LDA-MF indeed incorporates knowledge from the source domain, it has certain weaknesses which need to be addressed. The most significant drawback is that the dimensionalities of the LDA topic vector θ_i and the PMF user latent factor u_i are required to be equal. These two variables are of very different nature. One is extracted from text data in the source domain to model the topics

of the user-generated content, and the other is generated from the rating data in the target domain to model the latent interests of users. It is an overly strong assumption to assume the optimal dimensionalities for LDA and PMF are equal. In practice, if we choose the dimensionality to optimize the predictive power of PMF (e.g. by cross-validation on the rating data), the LDA model is likely to yield sub-optimal results and vice versa. The experiments that will be shown later confirm this concern. Furthermore, since the two variables are modeling different types of observations coming from different websites, the underlying meanings of the latent dimensions are unlikely to be identical. By treating the LDA topic vector as the prior mean of the PMF user latent factor, the latent dimensions are forced to be one-to-one aligned, which is again a strong assumption. Finally, the topic vectors are drawn from the Dirichlet distribution which has a bounded (and positive) support S, while the latent factors in PMF are unbounded Gaussian random vectors. If the optimal solution of u_i is far from S, the performance of the model could be affected, particularly in the cold-start situation where data is sparse and the prior plays an important role.

3 Nearest-Neighbor Transfer MF Model

To alleviate the drawbacks of the LDA-MF model, here we propose *nearest-neighbor transfer matrix factorization* (NT-MF) model to transfer user interests across websites. The entire framework is depicted in Figure 2.

We begin by describing the scenario in which our model operates. First, there is a rating-based recommender system (i.e. PMF) in the target domain, which suffers from the cold-start problem. The target domain might or might not contain content information. For example, in the video recommendation task, we can use the titles of all rated videos of a user to generate content information in the target domain. Such information is not available for the cold-start users since they have not rated any videos. However, in the source domain there are some content information available for these users. This can be, say, the content of a user's tweets. As previously mentioned, this type of auxiliary text data is immediately available when a user connects the accounts

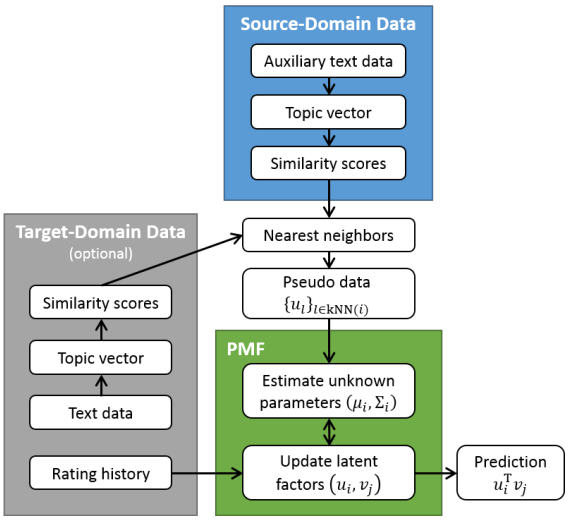


Figure 2: The entire system.

from two domains. Therefore, we assume this auxiliary text data is available for all users.

3.1 Model Outline

Next, we describe the high level concept of our model. As described previously, we have observed that the hypotheses encoded by the LDA-MF model is too strong as the PMF latent factor is enforced to inherit certain mathematical properties from the LDA topic vector. Here we loosen the constraint to only enforce that users should have similar distributions over the target-domain PMF latent factors if there is a high similarity between their source-domain topic vectors.

It is a reasonable hypothesis since our objective is to make the target-domain rating matrix factorize in a way that is consistent with the knowledge extracted from source-domain text. After all, the factorization of incomplete matrix is not unique, and there is no reason that the latent factor should match the topic factor of the user. In fact, our hypothesis implies a different distribution over the PMF latent factor for each user, i.e. $u_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where (μ_i, Σ_i) are unknown parameters, and are (possibly) different for each user.

To estimate the unknown parameters in a distribution, normally we need a set of observations, $(u_i^{(1)}, u_i^{(2)}, \dots)$. However, the parameters now belong to a distribution over a latent variable, which is non-trivial to estimate since we have no observations about the user latent factor. An exhaustive search

over the parameter space is obviously intractable. Even if we treat the entire model as a hierarchical model and learn the parameters indirectly from rating data, the cold-start problem immediately comes in and forbids us from learning a representative distribution for users.

We propose the idea of using the latent factors of the nearest-neighbors to estimate the unknown parameters in the distribution for a user. That is, the latent factors of the nearest-neighbors, $\{u_l\}_{l \in k\text{NN}(i)}$, are regarded as a set of *pseudo observations* to replace the unavailable data, $(u_i^{(1)}, u_i^{(2)}, \dots)$. However, the definition of "closeness" is not based on target-domain rating data, but computed by the topic vectors obtained from the content in the source domain (and the target domain, if available). Our model is thus not hampered by the cold-start problem.

Note that, in addition to the set of k-nearest-neighbors $k\text{NN}(i)$, we also have the corresponding similarity scores $sim(i, l)$ between each neighbor l and user i. The similarity scores along with the list of nearest-neighbors are transferred to the target domain to form a set of weighted samples, $\mathcal{D}$, which can be used to estimate the unknown parameters (μ_i, Σ_i), i.e.,

$$\mathcal{D} \equiv \begin{cases} \{u_l\}_{l \in k\text{NN}(i)} \\ w_l = sim(i, l). \end{cases} \tag{2}$$

The main purpose of assigning a sample weight w_l to each of the pseudo observations u_l is that by doing so, users with a higher source-domain similarity to user i will have a larger impact on the estimation of the target-domain parameters (μ_i, Σ_i). In other words, with this model specification, the *similarity between users* is transferred across domains.

3.2 Inference in NT-MF Model

To perform inference in our model, we adopt the maximum a posteriori (MAP) strategy and alternately update the user and item latent factors (i.e. by block coordinate ascent), similar to some previous solutions (Salakhutdinov and Mnih, 2007; Wang and Blei, 2011).

To solve for the optimal user latent factor u_i, we need to first estimate the unknown parameters (μ_i, Σ_i). Therefore, in our coordinate ascent algorithm, different from the original PMF model, we update the user latent factors one by one. That is, all user latent factors are regarded as fixed constants except for the one, u_i, to be updated. By doing so, for each user i, a set of *pseudo observations* about u_i (Eq. 2) is available. Using these pseudo observations, the unknown parameters (μ_i, Σ_i) can then be estimated with standard techniques such as maximum likelihood estimation (MLE). After an estimator of (μ_i, Σ_i) is obtained, we can analytically solve for the MAP solution of the user latent factor u_i. Then, we move on to the next user, and the coordinate ascent procedure continues. These two steps, namely the estimation of unknown parameters and the updating of the latent factors, are repeated until convergence.

One advantage of this procedure is that the list of nearest-neighbors and the similarities in Eq. 2 need not be recomputed during inference, avoiding expensive recomputation of pairwise similarities. It is also noticeable that, different from other transfer-based approaches, rating information and structured text from the source domain are not required in this procedure of model optimization. This further adds a level of flexibility to our framework for transferring user interests across websites.

3.3 Case Study: Inferring Unknown Mean

To clarify the previous discussions, we present a simple but detailed case-study on how an NT-MF model and its optimization procedure can be derived. The latent factor u_i for each user is assumed to be generated from a multivariate normal distribution with unknown mean μ_i and a known precision parameter λ_U, which is shared among the users.

The generative process proceeds as follows:

1. For each user i, draw user latent factor $u_i \sim \mathcal{N}(\mu_i, \lambda_U^{-1} I)$.
2. For each item j, draw item latent factor $v_j \sim \mathcal{N}(0, \lambda_V^{-1} I)$.
3. For each observed user-to-item pair (i, j), draw the rating $r_{ij} \sim \mathcal{N}(u_i^T v_j, \lambda_0^{-1})$,

where λ_0 is the precision parameter of the ratings, and λ_U, λ_V are the precision parameter of the

users and items, respectively. We use the notation $\mathcal{N}(x|\mu, \Sigma)$ to denote the Gaussian pdf with mean μ and covariance Σ.

The model is optimized by maximizing the posterior likelihood of the latent variables (an additive term is omitted),

$$\mathcal{L} = -\frac{\lambda_0}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}\gamma_{ij}\left(r_{ij} - u_i^T v_j\right)^2 - \frac{\lambda_U}{2}\sum_{i=1}^{M}\left(u_i - \mu_i\right)^T\left(u_i - \mu_i\right) - \frac{\lambda_V}{2}\sum_{j=1}^{N} v_j^T v_j, \tag{3}$$

where γ_{ij} is an indicator variable which is equal to 1 if item j is rated by user i, and 0 otherwise.

To solve the MAP problem, we need to first estimate the unknown parameters in the distribution, which in this case is the mean vector μ_i. The likelihood function over the pseudo observations, $\{u_l\}_{l\in k\text{NN}(i)}$, is defined as,

$$p(\mathcal{D}|\mu_i, \lambda_U) = \prod_{l\in k\text{NN}(i)} \mathcal{N}(u_l|\mu_i, \lambda_U^{-1} I). \tag{4}$$

By taking derivative of Eq. 4 with respect to μ_i and set it to zero, we obtain,

$$\sum_{l\in k\text{NN}(i)} (u_l - \mu_i) = 0, \tag{5}$$

which implies that the MLE of μ_i is the sample mean. However, since we are dealing with a set of *weighted samples*, the sample mean is replaced by the weighted average (the weights w_l are assumed to add up to one):

$$\mu_i = \sum_{l\in k\text{NN}(i)} w_l u_l. \tag{6}$$

Our model yields an intuitive result: to estimate the mean vector μ_i of u_i, we can simply take the weighted average of the latent factors u_l from the nearest-neighbors as an estimator, where the weights are the similarity scores between the textual profiles of user i and its neighbors.

Given μ_i, we can now maximize Eq. 3 with respect to u_i and v_j. By taking derivative of Eq. 3 with respect to u_i and v_j and set it to zero, we obtain the update equations,

$$\left(\sum_{j=1}^{N}\gamma_{ij} v_j v_j^T + \frac{\lambda_U}{\lambda_0} I\right) u_i = \sum_{j=1}^{N}\gamma_{ij} r_{ij} v_j + \frac{\lambda_U}{\lambda_0}\mu_i \tag{7}$$

$$\left(\sum_{i=1}^{M}\gamma_{ij} u_i u_i^T + \frac{\lambda_V}{\lambda_0} I\right) v_j = \sum_{i=1}^{M}\gamma_{ij} r_{ij} u_i. \tag{8}$$

Now with Eq. 6 to Eq. 8 at hand, we can iteratively solve for μ_i, u_i and v_j for all users and items until the model converges.

It can be seen from this case-study that NT-MF eliminates the three major drawbacks of the previously mentioned LDA-MF model. First, the topic vectors and the user latent factors are not required to have equal dimensionalities, which allows for the optimal dimensionality to be chosen in both models. Second, the mean vector, that is, the kNN weighted average in Eq. 6, is a linear combination of a set of user latent factors; as a result, the latent dimensions of u_i and μ_i are naturally aligned. Third, the mean vector μ_i has the same support as the user latent factor u_i, avoiding the risk of prior misspecification in cold-start situations.

4 Experiment

We use YouTube video recommendation to test the usefulness of NT-MF under the cold-start scenario. The NT-MF model used in this section follows the optimization procedure derived in Section 3.3.

4.1 Dataset and Statistics

To construct a dataset containing both the users' rating history and textual information, we begin with the user profile pages on Google+. A large proportion of Google+ users provide links to their profile pages from other social network services (e.g. Twitter). More importantly, if a user owns a YouTube account, a link to the user's YouTube channel will be automatically added to his Google+ profile. This makes a fully aligned dataset available. Users' Twitter accounts are obtained via their Google+ profile page, and the concatenation of tweets is regarded as the auxiliary text data. It has been shown that by concatenating the tweets, more representative user

topic vectors can be obtained (Hong and Davison, 2010). We refer to this text data as the *Twitter corpus*.

Videos in a user's "liked" or "favorite" playlists are considered to have a rating $r_{ij} = 1$. Other videos are assigned $r_{ij} = 0$. In other words, we are dealing with a one-class collaborative filtering (OCCF) problem (Pan et al., 2008). We adopt the same strategy as in (Wang and Blei, 2011) to deal with OCCF. First, all ratings are assumed to be observed, i.e. $\gamma_{ij} = 1$ for all user-item pairs. Next, a confidence parameter c_{ij} is introduced to reduce the influence of the huge number of zeroes during model optimization. The confidence parameter takes place of the original rating precision parameter λ_0 and is defined in (Wang and Blei, 2011) as $c_{ij} = a$ if $r_{ij} = 1$ and $c_{ij} = b$ otherwise $(a > b > 0)$. All the derivations in the previous sections follow intuitively.

The titles of the liked videos are concatenated and treated as the text data in the target domain (which we refer to as the *YouTube corpus*). As for the vocabulary, stopwords are first removed, and then 5000 words are selected from the YouTube corpus based on their TF-IDF scores (Blei and Lafferty, 2009). On average, each user's Twitter text data contains 5149 words and 1193 distinct terms, and each user's YouTube text data contains 158 words and 116 distinct terms. These statistics are in accordance with our assumption that text data in the source domain is abundant comparing to that in the target domain.

To validate the prediction result, each user has at least 10 liked videos. Videos with less than 5 likes are removed from the dataset. After data cleansing, there are 7328 users and 18691 videos in the dataset. The maximum number of likes received by a video is 98, and the average is 19.1. Among all videos, 92% of them are liked by less than 40 users. The maximum number of likes given by a user is 908, and the average is 48.8. Among all users, 89% of them have liked less than 100 videos. The sparsity (ratio of zeroes to the total number of entries) of the rating matrix is 99.74%, which illustrates the difficulty of this recommendation task.

4.2 Evaluation and Scenario

We choose the area under ROC curve (AUC) as the evaluation metric. AUC is often used to compare models when there is severe class imbalance, which is the case in our OCCF problem since we regard all zeroes as observed. All reported results are the average of 5 random data splits.

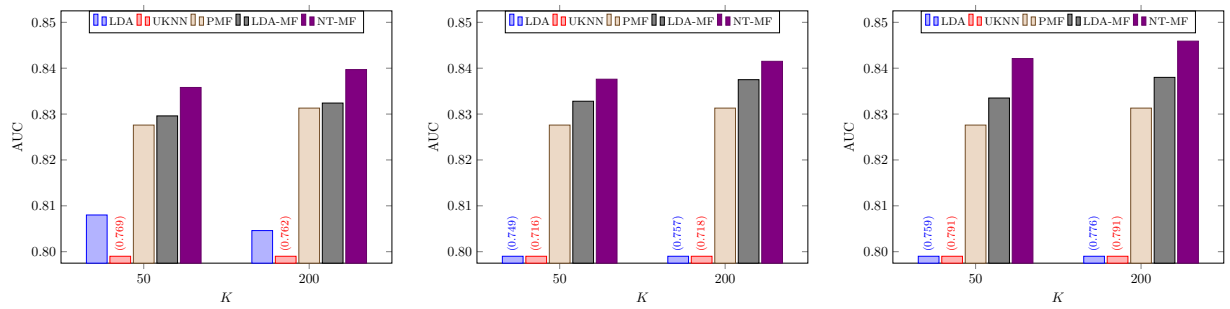
Similar to the experiments performed in (Wang and Blei, 2011), we test the performance of each model under two different scenarios. The first one is the task of *in-matrix* prediction. In this task, the likes received by each video are partitioned into three sets, namely the training, validation and testing sets. The ratio of data partition is 3:1:1. There are no cold-start users for the in-matrix prediction.

The second task is the *out-of-matrix* prediction, where the *users* are partitioned into three sets with the same 3:1:1 ratio. To make the two tasks comparable, we randomly split the data until the number of observations in each of the three sets is closed to that of the in-matrix task. Users in the testing set are all cold-start users. The only data we have when making prediction on the cold-start users is the auxiliary text data.

4.3 Baseline Methods

- **LDA:** We run linear regression on the LDA features to predict the ratings. This model serves as a content-based baseline.
- **UKNN:** The user-kNN algorithm (Herlocker et al., 1999) based on LDA features is implemented. This model serves as a neighborhood-based baseline.
- **PMF:** PMF (Salakhutdinov and Mnih, 2007) is a classic and widely-used CF model. It uses only the rating information, and thus is not capable of performing the out-of-matrix task.
- **LDA-MF:** This model is implemented as has been described in Section 2. It is similar to CTR (Wang and Blei, 2011) in structure. Since the optimization of the full model converges badly, we pre-train the LDA part of the model, and fix the topic vector when optimizing the PMF part.

All hyperparameters are tuned on the validation set. Due to efficiency and storage considerations, for UKNN and NT-MF, the k-nearest-neighbors are computed approximately with the FLANN library



(a) YouTube corpus (b) Twitter corpus (c) YouTube + Twitter corpus

Figure 3: In-matrix AUC using different corpus. For methods significantly worse than others, we cut off the plot and put the AUC values on top of the bars. NT-MF is significantly better than the baselines in all plots, according to a paired t-test ($p < 0.05$).

(Muja and Lowe, 2014). The symmetric Kullback-Leibler divergence is chosen to be the distance metric between topic vectors. For all baseline methods, we use K to denote the dimensionality of the latent variables. However, when discussing about NT-MF, since the number of topics can be different from the number of user latent factors, we use T to denote the former and K to denote the latter to avoid confusion.

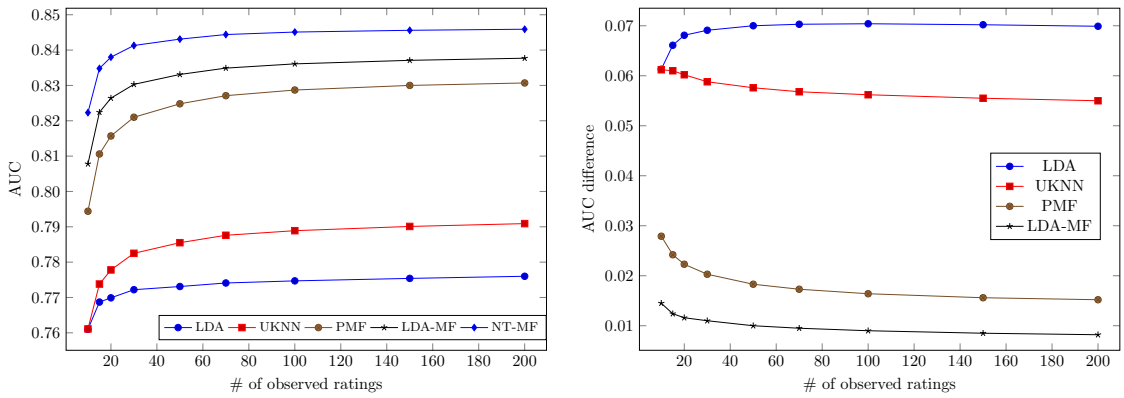
4.4 In-Matrix Prediction

In this section, the in-matrix prediction is discussed. First, we test the model's general performance on different corpora. Normally, the optimal number of topics will not be the same for different corpora. Since the LDA model performs the best with $K = 50$ on the YouTube corpus and $K = 200$ on the Twitter corpus, we report the results when K is set to these two numbers.

Figure 3(a) shows the results when no source-domain information is available and thus no transfer learning is performed. That is, all models are provided only with the YouTube ratings and the YouTube corpus. Because the YouTube corpus is scarce, the LDA model results in lower AUC when more topics are used, signifying overfitting. The same reason also leads to limited improvement of LDA-MF over PMF. Using neighborhood information alone, UKNN performs poorly. On the other hand, as a model bringing neighborhood information into PMF, NT-MF outperforms all baselines significantly. The above analysis shows that, although using either content (LDA) or neighborhood (UKNN) information alone is insufficient to generate good predictions, they can effectively improve the factorization of the rating matrix if used correctly.

To demonstrate the advantage of transfer learning, we study the scenario where only source-domain text and target-domain ratings are available. That is, the YouTube corpus in the previous analysis is replaced with the Twitter corpus. The result is shown in Figure 3(b). Comparing to Figure 3(a), we can see that although the Twitter corpus is larger than the YouTube corpus, it leads to a worse performance for LDA and UKNN. Content information from the noisy Twitter corpus alone is not sufficient to capture the rating behavior of users. However, by integrating the content information and rating history, both LDA-MF and NT-MF benefit from a larger corpus.

In the following analyses, we use data from both websites. For LDA, PMF and LDA-MF, we merge the two corpora by summing up the word counts. For UKNN and NT-MF, however, there is a more elegant way to combine the knowledge from different websites. First, we compute user similarity separately from the two corpora. Then, the two sets of similarity scores are weighted and averaged. Finally, the nearest-neighbors are computed based on this set of newly generated similarity scores. By applying this strategy to NT-MF, not only can θ_i and u_i differ in dimensionality, but also the optimal number of topics can be used for different corpora. Regardless of K, we use $T = 50$ for YouTube and $T = 200$ for Twitter in our NT-MF model. The result is shown in Figure 3(c). By comparing it with Figure 3(b), we can see that the AUC of NT-MF increases while that of LDA-MF remains unchanged. UKNN also benefits from this strategy. These facts show that, instead of merging the two corpora directly, our strategy of averaging the similarities is more advantageous.



(a) (b)

Figure 4: (a) Cumulative in-matrix AUC. Each point (x, y) in the figure means that the model gives an averaged AUC of y among all users who have less than or equal to x observed ratings. (b) Difference in cumulative in-matrix AUC between NT-MF and baseline methods.

Next, as a preliminary investigation of the performance on cold-start users, in Figure 4(a), we plot the *cumulative* AUC with respect to the total number of observed ratings. NT-MF outperforms other methods in terms of cumulative AUC regardless of the number of observed ratings. The advantage of NT-MF over the baseline methods is even greater as the number of observed ratings decreases (except for LDA). To make it clear, we plot the difference in AUC between NT-MF and the baseline methods in Figure 4(b). This phenomenon sheds light on the advantage of NT-MF under cold-start scenario.

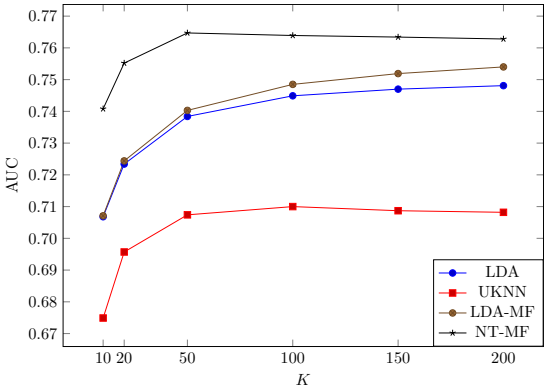


Figure 5: Out-of-matrix AUC. NT-MF is significantly better than the baselines, according to a paired t-test ($p < 0.05$).

4.5 Out-of-Matrix Prediction

In this section, we discuss the *out-of-matrix* prediction. Users in the testing set are all completely cold-start users. That is, we are only provided the Twitter corpus when making prediction for these users. Therefore, our previous strategy of averaging the similarities only applies to users in the training set. For this study we adopt the strategy of merging the two corpus instead of averaging the similarities. The number of topics $T = 150$ is chosen for NT-MF with respect to the validation AUC.

The result is presented in Figure 5. We plot the AUC against the dimensionality of the latent variables K. It can be observed that NT-MF beats all baseline methods regardless of K. Comparing to Figure 3, the out-of-matrix AUC is much lower, signifying the difficulty of cold-start recommendation.

Under the cold-start scenario, the latent factor used in the prediction phase is taken to be the prior mean for the MF-based models. For LDA-MF the prior mean is the topic vector θ_i, while for NT-MF it is the weighted average μ_i given by Eq. 6.

Since θ_i is used in place of u_i in the LDA-MF model when generating predictions, the curves of LDA and LDA-MF look very similar. A paired t-test ($p < 0.05$) shows no statistically significant difference between these two methods when $K = 10$ ($p = 0.48$) and $K = 20$ ($p = 0.09$). Despite the fact that $u_i = \theta_i$ is fixed for the cold-start users in the LDA-MF model, as K becomes larger, the item latent factors can carry more information in the rating data, which results in a higher AUC than LDA. However, since the dimensionalities of the LDA part and PMF part must match, the inference procedure of LDA-MF becomes very slow when K is large. To

make a better use of the available data, the computational efficiency must be sacrificed.

On the other hand, note that NT-MF achieves the highest AUC when $K = 50$. In fact, not only does NT-MF beat all baseline methods under different K values, it also outperforms the best LDA-MF model ($K = 200$) with fewer latent factors ($K = 20$). Unlike LDA-MF, the latent factors of the cold-start users are not fixed in NT-MF. Therefore, NT-MF can represent the information in a more concise way. In this case, NT-MF is better than LDA-MF in terms of both execution speed and predictive power.

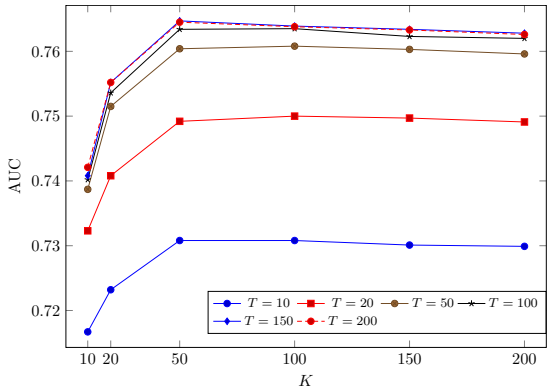


Figure 6: Performance of NT-MF based on out-of-matrix AUC for different values of K and T.

In Figure 6 we investigate the effect of different values of K and T. For each curve, we can see that the performance is about the same for $K \geq 50$. This is in accordance with the observation that NT-MF does not need as many latent factors as LDA-MF to achieve the same level of performance. Also, while increasing the number of topics T improves the performance in general, increasing T from 150 to 200 gives no significant improvement. The most important observation is that the highest AUC is achieved when $K = 50$ and $T = 150$. In other words, the optimal number of topics is different from that of user latent factors. This further justifies the advantage of NT-MF against previous methods.

5 Related Work

Although not directly aiming to solve the problem we have proposed, there exists some models of similar structure or adopt similar ideas.

As previously mentioned, LDA-MF is similar in structure to CTR. Collaborative topic Poisson factorization (CTPF) (Gopalan et al., 2014) combines the ideas of CTR and Poisson factorization (Gopalan et al., 2013) for a better performance. We have also tried CTPF on our dataset; nevertheless, there is no significant improvement over LDA-MF.

Recently, the neighborhood-aware probabilistic matrix factorization (NHPMF) model is proposed (Wu et al., 2012) as a method to combine kNN and PMF. It is originally proposed to leverage tagging data for improving PMF. This model can also be applied to our problem if we use the Twitter corpus in place of the unavailable tagging data. However, in the NHPMF model, the mean parameters are not treated as constants when the user latent factors are updated. As a result, an extra term appears in the gradient formula, which leads to an $O(k^2)$ time complexity, with k being the number of nearest-neighbors considered. On the other hand, the computation of the weighted average (i.e. Eq. 6) takes $O(k)$ time complexity. We have implemented NHPMF for comparison. As we increase k, NHPMF becomes significantly slower than NT-MF, while its performance is no better than NT-MF on our dataset.

6 Conclusion

In this work, we propose NT-MF, a cross-website transfer learning model which integrates content, neighborhood and rating information to alleviate the cold-start problem. A significant improvement over previous methods is demonstrated on a real-world cross-website dataset. The improvement is even more significant under the cold-start scenario.

So far we use the LDA topic vector to represent a user. As future work, different aspects of text can be taken into account to generate a more comprehensive user model. For example, writing styles or opinion mining may provide different insights on user behavior. Another possible extension is to apply our idea to more realistic settings such as large-scale and online recommender systems.

Acknowledgments

This material is based upon work supported by Microsoft Research Asia (MSRA) under award number FY16-RES-THEME-013 and by Taiwan Ministry of Science and Technology (MOST) under grant number 103-2221-E-002-104-MY2.

References

David M. Blei and John D. Lafferty. 2009. Topic models. In *Text Mining: Theory and Applications*. Taylor and Francis.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November.

Robin Burke. 2007. The adaptive web. chapter Hybrid Web Recommender Systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg.

Zhengyu Deng, Ming Yan, Jitao Sang, and Changsheng Xu. 2015. Twitter is faster: Personalized time-aware video recommendation from twitter to youtube. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(2):31:1–31:23, January.

Prem Gopalan, Jake M. Hofman, and David M. Blei. 2013. Scalable recommendation with poisson factorization. *CoRR*, abs/1311.1704.

Prem Gopalan, Laurent Charlin, and David M. Blei. 2014. Content-based recommendations with poisson factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3176–3184. Curran Associates, Inc.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA. ACM.

Bin Li, Qiang Yang, and Xiangyang Xue. 2009. Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 2052–2057, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marius Muja and David G. Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 502–511.

Michael J. Pazzani and Daniel Billsus. 2007. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg.

Suman Deb Roy, Tao Mei, Wenjun Zeng, and Shipeng Li. 2012. Socialtransfer: Cross-domain transfer learning from social streams for media applications. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 649–658, New York, NY, USA. ACM.

Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264.

Yue Shi, Martha Larson, and Alan Hanjalic. 2011. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization*, UMAP'11, pages 305–316, Berlin, Heidelberg. Springer-Verlag.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA. ACM.

Le Wu, Enhong Chen, Qi Liu, Linli Xu, Tengfei Bao, and Lei Zhang. 2012. Leveraging tagging for neighborhood-aware probabilistic matrix factorization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1854–1858, New York, NY, USA. ACM.

Speculation and Negation Scope Detection via Convolutional Neural Networks

Zhong Qian[1], Peifeng Li[1], Qiaoming Zhu[1], Guodong Zhou[1], Zhunchen Luo[2] and Wei Luo[2]

School of Computer Science and Technology, Soochow University, Suzhou, 215006, China[1]

China Defense Science and Technology Information Center, Beijing, 100142, China[2]

qianzhongqz@163.com, {pfli, qmzhu, gdzhou}@suda.edu.cn,

zhunchenluo@gmail.com, htqxjj@126.com

Abstract

Speculation and negation are important information to identify text factuality. In this paper, we propose a Convolutional Neural Network (CNN)-based model with probabilistic weighted average pooling to address speculation and negation scope detection. In particular, our CNN-based model extracts those meaningful features from various syntactic paths between the cues and the candidate tokens in both constituency and dependency parse trees. Evaluation on BioScope shows that our CNN-based model significantly outperforms the state-of-the-art systems on Abstracts, a sub-corpus in BioScope, and achieves comparable performances on Clinical Records, another sub-corpus in BioScope.

1 Introduction

Factual information is critical to understand a sentence or a document in most typical NLP applications. Speculation and negation extraction has been drawing more and more attentions in recent years due to its importance in distinguishing counterfactual or uncertain information from the facts. Generally speaking, speculation is a type of uncertain expression between certainty and negation, while negation is a grammatical category which reverses the truth value of a proposition.

Commonly, speculation and negation extraction involves two typical subtasks: cue identification and scope detection. Here, a cue is a word or phrase that has speculative or negative meaning (e.g., suspect, guess, deny, not), while a scope is a text fragment governed by the corresponding cue in a sentence. Consider the following two sentences for examples:

(S1) *The doctors warn that smoking [**may** harm our lungs]*.

(S2) *He does [**not** like playing football] but likes swimming*.[1]

In sentence S1, the speculative cue "***may***" governs the scope "***may** harm our lungs*", while the negative cue "***not***" governs the scope "***not** like playing football*" in sentence S2.

Previous work have achieved quite success on cue identification (e.g., with F1-score of 86.79 for speculative cue detection in Tang et al. (2010)). In comparison, speculation and negation scope detection is still a challenge due to its inherent difficulties and those upstream errors. In this paper, we focus on scope detection. Previous work on scope detection can be classified into heuristic rules based methods (e.g., Özgür et al., 2009; Øvrelid et al., 2010), machine learning based methods (e.g., Tang et al., 2010; Zou et al., 2013), and hybrid approaches which integrate empirical models with manual rules (Velldal et al., 2012).

Different from those previous studies, this paper presents a Convolutional Neural Network (CNN)-based approach for scope detection. CNN models, firstly invented to capture more abstract features for computer vision (LeCun et al., 1989), have achieved certain success on various NLP tasks in

[1] In this paper, cues are in ***bold face***, and scopes are in *[brackets]* in the example sentences.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 815–825,
Austin, Texas, November 1-5, 2016.

recent years, such as semantic role labeling (Collobert et al., 2011), machine translation (Meng et al., 2015; Hu et al., 2015), event extraction (Chen et al., 2015; Nguyen et al., 2015), etc. These studies have proved the ability of CNN models in learning meaningful features.

In particular, our CNN-based model extracts various kinds of meaningful features from the syntactic paths between the cue and the candidate token in both constituency and dependency parse trees. The importance of syntactic information in scope detection has been justified in previous work (Velldal et al., 2012; Lapponi et al., 2012; Zou et al., 2013, etc). Our model can also benefit from the ability of neural networks in extracting useful information from syntactic paths (Xu et al., 2015a; Xu et al., 2015b) or more complex syntactic trees (Ma et al., 2015; Tai et al., 2015). Moreover, instead of traditional average pooling, our CNN-based model utilizes probabilistic weighted average pooling to alleviate the overfitting problem (Zeiler et al., 2013). Experimental results on BioScope prove the effectiveness of our CNN-based model.

The reminder of this paper is organized as follows: Section 2 gives an overview of the related work. Section 3 describes our CNN-based model with probabilistic weighted average pooling for scope detection. Section 4 illustrates the experimental settings, and reports the experimental results and analysis. Finally, Section 5 draws the conclusion.

2 Related Work

In this section, we give an overview of previous work on both scope detection and utilization of CNNs in NLP applications.

2.1 Scope Detection

Earlier studies on speculation and negation scope detection focused on developing various heuristic rules manually to detect scopes.

Chapman et al. (2001) developed various regular expressions for negation scope detection. Subsequently, various kinds of heuristic rules began to emerge. Özgür et al. (2009) resorted to the part-of-speech of the speculative cues and the syntactic structures of the current sentences for identifying scopes, and developed heuristic rules according to the syntactic trees. Øvrelid et al. (2010) constructed a set of heuristic rules on dependency structures and obtained the accuracy of 66.73% on the CoNLL evaluation data. The approaches based on heuristic rules were effective because the sentence structures in BioScope satisfy some grammatical rules to a certain extent.

With the release of the BioScope corpus (Szarvas et al., 2008), machine learning based methods began to dominate the research of speculation and negation scope detection.

Morante et al. (2008) regarded negation scope detection as a chunk classification task utilizing lexical and syntactic features. Morante et al. (2009a) further implemented a scope detection system combining three classifiers, i.e., TiMBL, SVM and CRF, based on shallow syntactic features, and achieved the performance of 77.13% and 73.36% in Percentage of Correct Scopes (PCS) on speculation and negation scope detection on Abstracts, a sub-corpus of BioScope. Velldal et al. (2012) explored a hybrid method, adopting manually crafted rules over dependency parse trees and a discriminative ranking function over nodes in constituent parse trees. Zou et al. (2013) proposed a tree kernel based approach on the syntactic parse trees to detect speculation and negation scopes.

Alternative studies treated scope detection as a sequential labeling task. Tang et al. (2010) proposed a CRF model with POS, chunks, NERs, dependency relations as features. Similarly, Lapponi et al. (2012) employed a CRF model with lexical and dependency features for negation scope and event resolution on the Conan Doyle corpus. These machine learning methods manifest the effectiveness of syntactic features.

2.2 CNN based NLP Applications

Currently, CNNs have obtained certain success on various NLP tasks, e.g., part-of-speech tagging, chunking, named entity recognition (Collobert et al., 2011). Specifically, CNNs have been proven effective in extracting sentence-level features. For instance, Zeng et al. (2014) utilized a CNN-based model to extract sentence-level features for relation classification. Zhang et al. (2015) proposed a shallow CNN-based model for implicit discourse relation recognition. Chen et al. (2015) presented a CNN-based model with dynamic multi-pooling on event extraction.

More recently, researchers tend to learn features from complex syntactic trees. Ma et al. (2015) use

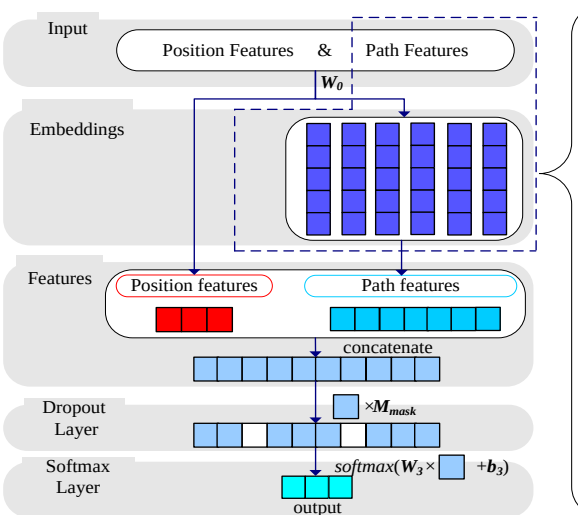


Figure 1: The framework of CNN for scope detection.

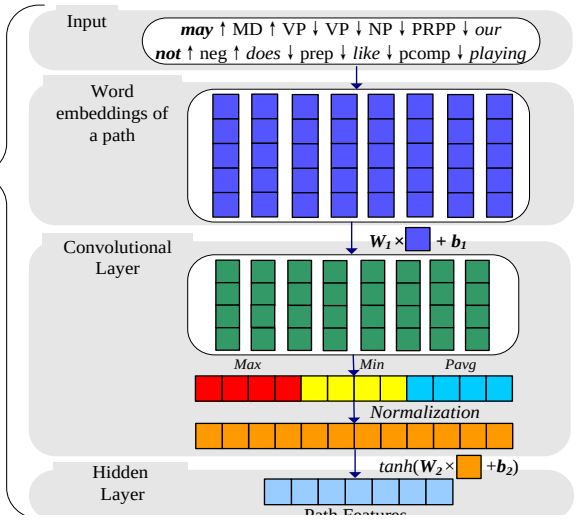


Figure 2: The architecture of CNN-based model to extract path features.

a CNN-based model for sentence embedding, utilizing dependency tree-based n-grams. Xu et al. (2015a) exploited a CNN-based model to learn features from the shortest dependency path between the subject and the object for semantic relation classification.

3 CNN-based Modeling with Probabilistic Weighted Average Pooling

This section describes our CNN-based model for speculation and negation scope detection, which is recast as a classification task to determine whether each token in a sentence belongs to the scope of the corresponding cue or not. Principally, our CNN-based model first extracts path features from syntactic trees with a convolutional layer and concatenates them with their relative positions into one feature vector, which is then fed into a softmax layer to compute the confidence scores of its location labels, described in subsection 3.1.

3.1 Token Labeling

We employ following labeling scheme for each candidate token:

- A token is labeled as ***O*** if it is NOT an element of a speculation or negation scope;
- A token is labeled as ***B*** if it is inside a scope and occurs before the cue, i.e., $P_{token} < P_{cue}$, where P_{token} and P_{cue} are the positions of the token and the cue in a sentence, respectively;
- A token is labeled as ***A*** if it is inside a scope and occurs after the cue (inclusive), i.e., $P_{token} \geqslant P_{cue}$.

Under this scheme, each token in a sentence is classified into ***B***, ***A*** or ***O***. For example, the labels of all the tokens in sentence S3 are shown in sentence S4.

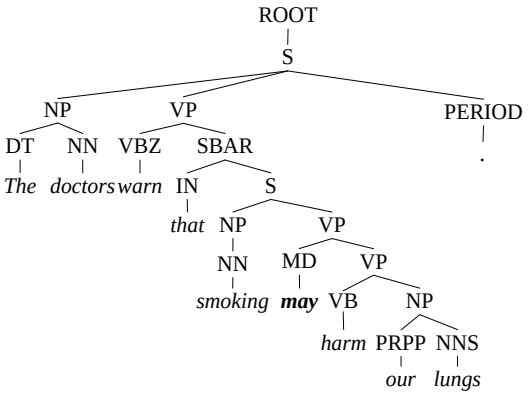
(S3) *They think that [those bacteria **may** be killed by white blood cells] , but other researchers do not think so.*

(S4) *They/O think/O that/O [those/B bacteria/B **may/A** be/A killed/A by/A white/A blood/A cells/A] ,/O but/O other/O researchers/O do/O not/O think/O so/O ./O*

The advantage of our scheme is that it can describe the location relationship among the tokens, cues and scopes more precisely than some previous studies, which regarded scope detection as a binary classification task (Øvrelid et al., 2010; Zou et al., 2013). Compared to other schemes with more than two labels (Morante et al., 2009a; Tang et al., 2010; Lapponi et al., 2012), our scheme can much alleviate the imbalance of labels, because the tokens occurring at the first or last positions of the scopes are much fewer than other tokens.

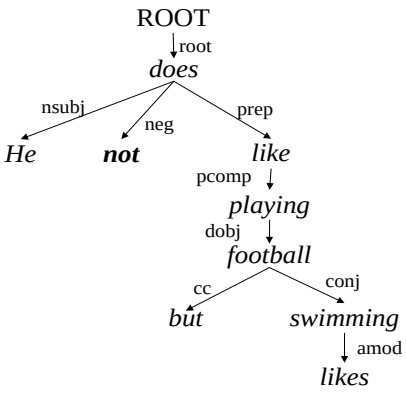
3.2 Input Representation

Figure 1 shows the framework of our model based on neural network. We concentrate on **Path Feature** and **Position Feature**. They are concatenated into one feature vector, which is finally fed into the softmax layer to obtain the output vector.



Cue: ***may***
Current candidate token: *our*
Constituency path:
may ↑ MD ↑ VP ↓ VP ↓ NP ↓ PRPP ↓ *our*

Figure 3: An example for the constituency parse tree of sentence S1 and the path from the cue to the candidate token.



Cue: ***not***
Current candidate token: *playing*
Dependency path:
not ↑ neg ↑ *does* ↓ prep ↓ *like* ↓ pcomp ↓ *playing*

Figure 4: An example for the dependency parse tree of sentence S2 and the path from the cue to the candidate token.

Relative Position has been proven useful in previous studies (Zeng et al., 2014; Chen et al., 2015). In this paper, relative position is defined as the relative distance of the cue to the candidate token. For instance, in sentence S1, the relative distances of the cue "***may"*** to the candidate tokens "*warn"* and "*our"* are 3 and -2, respectively. The values of position features are mapped into a vector $\boldsymbol{P}$ of dimension d_p, with $\boldsymbol{P}$ initialized randomly.

Instead of the word sequence (e.g., Zeng et al., 2014; Zhang et al. 2015; Chen et al., 2015), we argue that the **Shortest Syntactic Path** from the cue to the candidate token can offer effective features to determine whether a token belongs to the scope. It is remarkable that the lowest common ancestor node of the cue and the token is the highest tree node in the path.

Figure 2 illustrates the architecture of our CNN-based model to extract path features. Here, convolutional features are first extracted from the matrix of embeddings of the path, and then fed into the hidden layer to produce more complicated features.

In this paper, the syntactic paths between the cues and the candidate tokens in constituency and dependency parse trees are both considered. Figure 3 presents the constituency parse tree of sentence S1 and the constituency path from the cue "***may***" to the candidate token "*our*". It shows that the tokens are at both the beginning and the end of the path with the arrows indicating the directions. Meanwhile, Figure 4 displays the dependency parse tree of sentence S2 and the dependency path from the cue "***not***" to the token "*playing*".

As the input of our CNN-based model, both the constituency path and the dependency path between the cue and the token can be regarded as the special "sentences" $S=(t_1, t_2, \ldots, t_n)$, whose "words" can be tokens of sentences, syntactic categories, dependency relations, and arrows.

Similar to other CNN-based models, we also consider a fixed size window of tokens around the current token to capture its local features in the path. Here, the window size is set as an odd number w, indicating that there are (w-1)/2 tokens before and after the candidate token, respectively. In this case, path S is transferred into matrix $\boldsymbol{X}_0 \in \mathbf{R}^{w d_0 \times n}$ according to embedding table $\boldsymbol{W}_0 \in \mathbf{R}^{d_0 \times |T_0|}$, where d_0 is the dimension of the embeddings and $|T_0|$ is the size of the table.

3.3 Convolutional Neural Networking

After fed into the convolutional layer, the matrix of the syntactic path $\boldsymbol{X_0}$ is processed with a linear operation:

$$\boldsymbol{Y}_1 = \boldsymbol{W}_1 \boldsymbol{X}_0 + \boldsymbol{b}_1 \quad (1)$$

where $\boldsymbol{W}_1 \in \mathbf{R}^{n_1 \times w d_0}$ is the parameter matrix, and $\boldsymbol{b}_1 \in \mathbf{R}^{n_1}$ is the bias term. To extract the most active convolutional features from $\boldsymbol{Y}_1 \in \mathbf{R}^{n_1 \times n}$, we consider two features ***Cmax*** and ***Cmin*** whose elements are maximum, minimum values of rows in $\boldsymbol{Y_1}$, respectively:

$$\boldsymbol{Cmax}(r) = max[\boldsymbol{Y_1}(r,0), \boldsymbol{Y_1}(r,1), ..., \boldsymbol{Y_1}(r,n-1)] \quad (2)$$

$$\boldsymbol{Cmin}(r) = min[\boldsymbol{Y_1}(r,0), \boldsymbol{Y_1}(r,1), ..., \boldsymbol{Y_1}(r,n-1)] \quad (3)$$

where $0 \le r \le n_1 - 1$. Moreover, we extract a convolutional feature ***Cpavg***, whose elements are probabilistic weighted average values of rows in $\boldsymbol{Y_1}$. Formally, ***Cpavg*** can be written as:

$$\boldsymbol{Cpavg}(r) = \sum_{i=0}^{n-1} p_i \cdot \boldsymbol{Y_1}(r,i) \quad (4)$$

In Equation (4), p_i is the probability of the element $\boldsymbol{Y_1}(r,i)$ in the vector $\boldsymbol{Y_1}(r,\cdot)$:

$$p_i = \frac{\left|\boldsymbol{Y_1}(r,i)\right|}{\sum_{j=0}^{n-1} \left|\boldsymbol{Y_1}(r,j)\right|} \quad (5)$$

Cpavg is a variant probabilistic weighted average pooling used by Zeiler et al. (2013). Compared to the standard average pooling, each element in ***Cpavg*** has a weight depending on its value. That is, during computing ***Cpavg***, the most active elements with the largest absolute values (i.e., the maximum and minimum values) play the leading roles, while those less active elements with smaller absolute values have less effect. In this way, we can reduce the influence of less active elements, and can capture more active information in $\boldsymbol{Y_1}(r,\cdot)$. From this respect, ***Cpavg*** can be regarded as a meaningful convolutional feature.

The extracted convolutional features above are first concatenated into $\boldsymbol{C} \in \mathbf{R}^{3n_1}$, as the output of the convolutional layer:

$$\boldsymbol{C} = [\boldsymbol{Cmax}, \boldsymbol{Cmin}, \boldsymbol{Cpavg}] \quad (6)$$

Then, ***C*** is fed into the hidden layer to learn more complex and meaningful features. Here, we process ***C*** with a linear operation just like in the convolutional layer, and choose hyperbolic *tanh* as the activation function to get $\boldsymbol{Y_2} \in \mathbf{R}^{n_2}$:

$$\boldsymbol{Y_2} = tanh(\boldsymbol{W_2 C} + \boldsymbol{b_2}) \quad (7)$$

where $\boldsymbol{W_2} \in \mathbf{R}^{n_2 \times 3n_1}$ is the parameter matrix, and $\boldsymbol{b_2} \in \mathbf{R}^{n_2}$ is the bias term. To produce the output of the hidden layer, a normalization operation is applied to eliminate the manifold differences among various features:

$$\boldsymbol{H} = \boldsymbol{Y_2} / \left\|\boldsymbol{Y_2}\right\| \quad (8)$$

In this way we obtain the path feature $\boldsymbol{H} \in \mathbf{R}^{n_2}$ for each candidate token and then concatenate it with the position feature ***P*** into one vector $\boldsymbol{F_0}$:

$$\boldsymbol{F_0} = [\boldsymbol{P}^T, \boldsymbol{H}^T]^T \quad (9)$$

where $\boldsymbol{F_0} \in \mathbf{R}^{n_f}$ is the feature vector of a candidate token with the dimension equaling the sum of n_2 and the dimension of ***P***. Besides, we also consider the dropout operation for regularization to prevent the co-adaptation of hidden units on the penultimate layer:

$$\boldsymbol{F_1} = \boldsymbol{F_0} \circ \boldsymbol{M} \quad (10)$$

where $\circ$ is an element-wise multiplication and ***M*** is a mask vector whose elements follow the Bernoulli distribution with the probability p of being 1. We determine whether the candidate token is in the scope of the current cue according to its $\boldsymbol{F_1}$.

3.4 Output

Finally, $\boldsymbol{F_1}$ is fed into the softmax layer:

$$\boldsymbol{O} = softmax(\boldsymbol{W_3 F_1} + \boldsymbol{b_3}) \quad (11)$$

where $\boldsymbol{W_3} \in \mathbf{R}^{n_3 \times n_f}$ is the parameter matrix, and $\boldsymbol{b_3} \in \mathbf{R}^{n_3}$ is the bias term. The dimension of ***O*** is n_3=3, which is equal to the number of labels representing whether the token is an element of the scope, just as described in subsection 3.1, and the elements of ***O*** can be interpreted as the confidence scores of the three labels, i.e., ***B***, ***A*** and ***O***.

To learn the parameters of the network, we supervise the predicted labels of ***O*** with the gold labels in the training set, and utilize the following training objection function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \log\, p(y_i \mid x_i, \theta) + \frac{\lambda}{2}\left\|\theta\right\|^2 \quad (12)$$

where $p(y_i \mid x_i, \theta)$ is the confidence score of the golden label y_i (***B***, ***A***, ***O***) of the training instance x_i, m is the number of the training instances, λ is the regularization coefficient and θ={$\boldsymbol{W_0}$, $\boldsymbol{W_1}$, $\boldsymbol{b_1}$, $\boldsymbol{W_2}$, $\boldsymbol{b_2}$, $\boldsymbol{W_3}$, $\boldsymbol{b_3}$} is the set of parameters. To train the CNN-based model, the Stochastic Gradient Descent algorithm is applied to fine-tune θ.

4 Experimentation

In this section, we first introduce the evaluation data, and then describe the experimental settings. Finally, we report the experimental results and analysis.

4.1 Corpus

We evaluate our CNN-based model on BioScope (Szarvas et al., 2008; Vincze et al., 2008), a widely used and freely available resource consisting of sentences annotated with speculative and negative cues and their scopes in biomedical domain.

		Abs	Papers	Cli
Total	#Documents	1273	9	1954
	#Sentences	11871	2670	6383
	Ave. Len Sentences	25.47	24.54	7.71
Spe	#Sentences	2101	519	855
	#Scopes	2659	672	1112
	Ave. Len Sentences	29.77	30.76	11.96
	Ave. Len Scopes	15.10	13.38	4.92
Neg	#Sentences	1597	339	865
	#Scopes	1719	376	870
	Ave. Len Sentences	29.28	30.55	8.53
	Ave. Len Scopes	7.60	7.35	3.87

(Notes: "Ave. Len" denotes average length; "Abs", "Papers" and "Cli" denote Abstracts, Full Papers and Clinical Records, respectively; "Spe" and "Neg" denote speculation and negation, respectively.)

Table 1: Statistics on the BioScope corpus.

BioScope includes 3 different sub-corpora: Abstracts of biological papers from the GENIA corpus (Collier et al., 1999), Full scientific Papers from Flybase and BMC Bioinformatics website, and Clinical radiology Records corpus. These texts in three sub-corpora ensure that BioScope can capture the heterogeneity of language use in biomedical domain. While Abstracts and Full Papers share the same genre, Clinical Records consists of shorter sentences. Previous studies regarded Abstracts as the main resource for text mining applications due to its public accessibility (e.g. through PubMed).

Table 1 shows the statistics of the BioScope corpus. While in both Abstracts and Full Papers, the average lengths of speculation and negation sentences are comparable (Abstracts: 29.77 vs 29.28; Full Papers: 30.76 vs 30.55). However, their average lengths of the negation scopes are shorter than those of speculation ones (Abstracts: 7.60 vs 15.10; Full Papers: 7.35 vs 13.38). Moreover, both the average lengths of sentences and scopes in Clinical Records are shorter than those of other two sub-corpora (Average length: 11.96 (speculation sentence), 8.53 (negation sentence), 4.92 (speculation scope) and 3.87 (negation scope)).

4.2 Experimental Settings

Following the previous work (e.g., Özgür et al., 2009; Morante et al., 2009a, 2009b; Zou et al., 2013), we divide the Abstracts sub-corpus into 10 folds to perform 10-fold cross-validation. Moreover, to examine the robustness of our CNN-based model towards different text types within biomedical domain, all the models are trained on the same Abstracts sub-corpus. Therefore, the results on Abstracts can be regarded as in-domain evaluation while the results on Clinical Records and Full Papers can be regarded as cross-domain evaluation.

For the measurement, traditional Precision, Recall, and F1-score are used to report the token-based performance in scope detection, while the Percentage of Correct Scopes (PCS) is adopted to report the scope-based performance, which considers a scope correct if all the tokens in the sentence have been assigned the correct scope classes for a specific cue. Obviously, PCS can better describe the overall performance in scope detection. Besides, Percentage of Correct Left Boundaries (PCLB) and Percentage of Correct Right Boundaries (PCRB) are reported as partial measurements.

In all our experiments, both the constituency and dependency parse trees are produced by Stanford Parser[2]. Specially, we train the parser on the GENIA Treebank 1.0[3] (Tateisi et al., 2005), which contains Penn Treebank-style syntactic (phrase structure) annotation for the GENIA corpus. The parser achieves the performance of 87.12% in F1-score in terms of 10-fold cross-validation on GENIA TreeBank 1.0.

For the hyper-parameters in our CNN-based model, we set d_0=100, d_p=10, w=3, n_1=200, n_2=500, λ=10^{-4}, p=0.8. The embeddings of the tokens in ordinary sentences (as word sequences) are initialized by Word2Vec[4] (Mikolov et al., 2013).

For the baseline, we utilize the classifier-based baseline developed by Zou et al. (2013). Besides those typical features, constituency and dependen-

[2] http://nlp.stanford.edu/software/lex-parser.shtml
[3] http://www.geniaproject.org/genia-corpus/treebank
[4] https://code.google.com/archive/p/word2vec/

	Systems	P(%)	R(%)	F1	PCLB(%)	PCRB(%)	PCS(%)
Speculation	Baseline	94.71	90.54	92.56	84.81	85.11	72.47
	CNN_C	**95.95**	**95.19**	**95.56**	**93.16**	**91.50**	**85.75**
	CNN_D	92.25	94.98	93.55	86.39	84.50	74.43
Negation	Baseline	85.46	72.95	78.63	84.00	58.29	46.42
	CNN_C	85.10	**92.74**	89.64	81.04	**87.73**	70.86
	CNN_D	**89.49**	90.54	**89.91**	**91.91**	83.54	**77.14**

Table 2: The performances on the Abstracts sub-corpus.

cy syntactic features are also included. Furthermore, Mallet[5] is selected as the classifier.

In addition, since our CNN-based model may result in discontinuous blocks, we utilize a post-processing algorithm (Morante et al., 2008) to ensure the continuity of scopes. Meanwhile, the cue must be in its scope as defined in Bioscope.

4.3 Experimental Results on Abstracts

Table 2 summarizes the performances of scope detection on Abstracts. In Table 2, CNN_C and CNN_D refer the CNN-based model with constituency paths and dependency paths, respectively (the same below). It shows that our CNN-based models (both CNN_C and CNN_D) can achieve better performances than the baseline in most measurements. This indicates that our CNN-based models can better extract and model effective features. Besides, compared to the baseline, our CNN-based models consider fewer features and need less human intervention. It also manifests that our CNN-based models improve significantly more on negation scope detection than on speculation scope detection. Much of this is due to the better ability of our CNN-based models in identifying the right boundaries of scopes than the left ones on negation scope detection, with the huge gains of 29.44% and 25.25% on PCRB using CNN_C and CNN_D, respectively.

Table 2 illustrates that the performance of speculation scope detection is higher than that of negation (Best PCS: 85.75% vs 77.14%). It is mainly attributed to the shorter scopes of negation cues. Under the circumstances that the average length of negation sentences is almost as long as that of speculation ones (29.28 vs 29.77), shorter negation scopes mean that more tokens do not belong to the scopes, indicating more negative instances. The imbalance between positive and negative instances has negative effects on both the baseline and the CNN-based models for negation scope detection.

Table 2 also shows that our CNN_D outperforms CNN_C in negation scope detection (PCS: 77.14% vs 70.86%), while our CNN_C performs better than CNN_D in speculation scope detection (PCS: 85.75% vs 74.43%). To explore the results of our CNN-based models in details, we present the analysis of top 10 speculative and negative cues below on CNN_C and CNN_D, respectively.

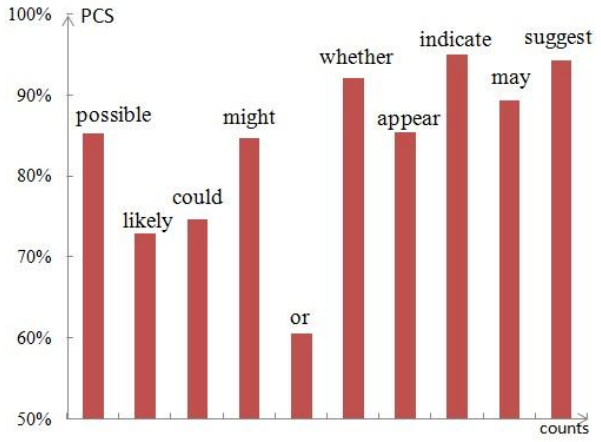


Figure 5: PCSs of top 10 speculative cues for scope detection in Abstracts sub-corpus.

Figure 5 illustrates the PCSs of the most frequent 10 speculative cues using CNN_C. The cues in the horizontal axis are in the order of lowest to highest in frequency. Among those cues, "*suggest"*, *"may"*, *"indicate"*, and "*appear"* are commonly used to express opinions of certain individuals. The scopes of these cues are integrated semantic fragments (probably clauses) governed by corresponding cues in grammatical sense, and the tokens in the scope tend to share the same chunk with the cue in the constituency parse tree. Hence, constituency paths are more useful for speculation scope detection. Figure 5 also shows that the PCSs of all the top 10 speculative cues are higher than 70% except "*or"* (PCS: 60.44%), mainly due to the flexible usage of "*or"*, which

[5] http://mallet.cs.umass.edu/

	Systems		P(%)	R(%)	F1	PCLB(%)	PCRB(%)	PCS(%)
Speculation	Clinical Records	CNN_C	86.85	**93.84**	**90.21**	**84.35**	**86.87**	**73.92**
		CNN_D	**89.02**	85.41	87.18	82.91	76.17	64.39
	Full Papers	CNN_C	**86.78**	**86.59**	**86.69**	**86.01**	**68.60**	**59.82**
		CNN_D	86.13	85.09	85.61	80.95	64.14	52.98
Negation	Clinical Records	CNN_C	88.29	97.00	92.44	**95.98**	**93.45**	**89.66**
		CNN_D	**91.97**	**97.03**	**94.43**	**95.98**	90.57	87.82
	Full Papers	CNN_C	80.92	82.26	81.58	82.71	**67.29**	**55.32**
		CNN_D	**82.08**	**84.90**	**83.46**	**84.04**	64.89	53.99

Table 3: The performances of our CNN-based models on Clinical Records and Full Papers.

can connect two words, two professional terms, or even two clauses.

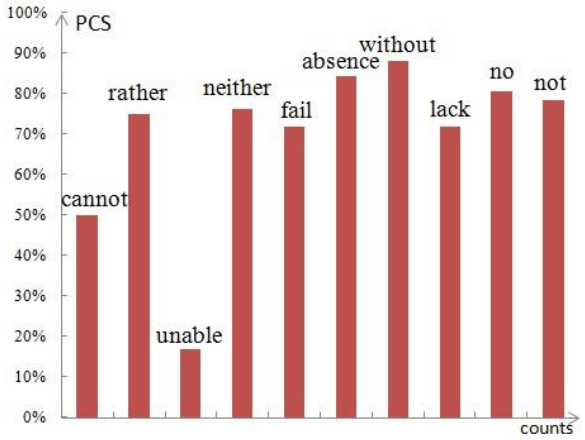


Figure 6: PCSs of top 10 negative cues for scope detection in Abstracts sub-corpus.

Figure 6 illustrates the performances of the most frequent 10 negative cues using CNN_D. In those negative cues, "*not"* is in the absolute majority, and "*not"* and "*no"* cover over 70%. We have noticed that most negative cues (e.g., "*not", "no", "without", "fail"*) are often applied to negate phrases, and the tokens in negation scope tend to have the tight dependency relationship with them. Therefore, our model can achieve better results using dependency paths for negation scope.

In Figure 6, most negative cues have good PCSs (higher than 70%). However, "*unable"* has poor PCS of 16.67%. This is due to the fact that "*unable"* usually occurs in the phrase structure "*be unable to"*, which often follows a subject. It is notable that a cue is always in its scope and most cues in BioScope are much closer to the left boundaries than to the right ones. Hence, the tokens labeled as ***B*** (i.e., inside the scope and before the cue) are much fewer than the ones labeled as ***A*** or ***O***. Such imbalance makes it hard to judge whether the tokens before "*unable"* are in of its scope or not.

4.4 Experimental Results on Clinical Records and Full Papers

The performances of our CNN-based models on the other two sub-corpora, i.e., Clinical Records and Full Papers, are presented in Table 3. Although Abstracts and Clinical Records have different genres, our CNN-based models can obtain satisfactory results on Clinical Records using both constituency paths and dependency paths, proving the portability of our models.

Table 3 also shows that the results of negation scope are better than those of speculation scope on Clinical Records (PCS: 89.66% vs 73.92%). We argue the reason is that both the lengths of negation sentences and scopes (8.53 and 3.87, respectively) in Clinical Records are much shorter, indicating that the structures of negation sentences are simpler than those of speculation ones. After error analysis of speculation scopes, we find that 54.83% of our error scopes contain the annotated scopes, just like sentence S5:

(S5) *This does not [**appear** to represent a stone] and is not mobile.*

The annotated scope of the cue "***appear***" is "***appear*** *to represent a stone*". However, our CNN-based model identifies the whole sentence as the scope. These errors indicate that some words may be wrongly identified as the components of scopes because the scopes in Clinical Records are short and their structures are simple.

Compared with Abstracts and Clinical Records, the results on Full Papers are much lower. This is mainly due to the poor PCRBs, indicating a considerable quantity of right boundaries of scopes cannot be identified correctly. We should note that the average lengths of both speculation and negation sentences (30.76 and 30.55, respectively) in Full Papers are longer than those in Abstracts and

Clinical Records. Normally, longer sentences mean more complicated syntactic structures.

Besides the results trained on Abstracts, we also consider the 10-fold cross-validation on Clinical Records and Full Papers. The PCSs of speculation and negation scope detection are 74.73% (CNN_C) and 91.03% (CNN_C) on Clinical Records, which are both higher than the ones trained on Abstracts. Remember that Abstracts and Clinical Records come from the different genres. However, we get lower PCSs on Full Papers (49.54% for speculation scope detection using CNN_C, and 44.67% for negation scope detection using CNN_C). In addition to the complex structures of long sentences, another reason is that the smaller size of the Full Papers sub-corpus compared to the other two sub-corpora. Fewer sentences and scopes (only 672 speculation scopes in 519 sentences and 376 negation scopes in 339 sentences) mean that we cannot get an excellent model.

4.5 Comparison with the State-of-the-Art

Table 4 compares our CNN-based models with the state-of-the-art systems. It shows that our CNN-based models can achieve higher PCSs (+1.54%) than those of the state-of-the-art systems for speculation scope detection and the second highest PCS for negation scope detection on Abstracts, and can get comparable PCSs on Clinical Records (73.92% vs 78.69% for speculation scopes, 89.66% vs 90.74% for negation scopes). It is worth noting that Abstracts and Clinical Records come from different genres.

It also displays that our CNN-based models perform worse than the state-of-the-art on Full Papers due to the complex syntactic structures of the sentences and the cross-domain nature of our evaluation. Although our evaluation on Clinical Records is cross-domain, the sentences in Clinical Records are much simpler and the results on Clinical Records are satisfactory. Remind that our CNN-based models are all trained on Abstracts. Another reason is that those state-of-the-art systems on Full Papers (e.g., Li et al., 2010; Velldal et al., 2012) are tree-based, instead of token-based. Li et al. (2010) proposed a semantic parsing framework and focused on determining whether a constituent, rather than a word, is in the scope of a negative cue. Velldal et al. (2012) presented a hybrid framework, combining a rule-based approach using dependency structures and a data-driven approach for selecting appropriate subtrees in constituent structures. Normally, tree-based models can better capture long-distance syntactic dependency than token-based ones. Compared to those tree-based models, however, our CNN-based model needs less manual intervention. To improve the performances of scope detection task, we will explore this alternative in our future work.

	System	Abs	Cli	Papers
Spe	Morante (2009a)	77.13	60.59	47.94
	Özgür (2009)	79.89	N/A	61.13
	Velldal (2012)	79.56	**78.69**	**75.15**
	Zou (2013)	84.21	72.92	67.24
	Ours	**85.75**	73.92	59.82
Neg	Morante (2008)	57.33	N/A	N/A
	Morante (2009b)	73.36	87.27	50.26
	Li (2010)	**81.84**	89.79	64.02
	Velldal (2012)	74.35	**90.74**	**70.21**
	Zou (2013)	76.90	85.31	61.19
	Ours	77.14	89.66	55.32

Table 4: Comparison of our CNN-based model with the state-of-the-art in PCS.

5 Conclusion

This paper proposes a CNN-based model for speculation and negation scope detection. Compared with various lexical and syntactic features adopted in previous studies (e.g., Lapponi et al., 2012; Zou et al., 2013), our CNN-based model only considers the position feature and syntactic path feature.

Experimental results on the BioScope corpus show that our CNN-based model can get the best performances for speculation scopes and the second highest performances for negation scopes on Abstracts in in-domain evaluation. In cross-domain evaluations, we can achieve comparable results on Clinical Records, but our CNN-based model performs worse on Full Papers. This suggests our future direction to extend the model from token level to parse tree level in better capturing long-distance syntactic dependency and to address the cross-domain adaptation issue.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant Nos. 61472265, 61402314 and 61331011), and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. In addition, thanks to the three anonymous reviewers for their valuable comments.

References

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. Evaluation of Negation Phrases in Narrative Clinical Reports. In *Proceedings of American Medical Informatics Association Symposium*, 2001, pages 105-109.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL2015)*, Beijing, China, July 26-31, 2015, pages 167-176.

Nigel Collier, Hyun Seok Park, Norihiro Ogata, et al. 1999. The GENIA Project: Corpus-based Knowledge Acquisition and Information Extraction from Genome Research Papers. In *Proceedings of the European Chapter of the ACL 1999 (EACL 1999)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 2011, 12(1): 2493-2537.

Baotian Hu, Zhaopeng Tu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2015. Context-Dependent Translation Selection Using Convolutional Neural Network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL2015)*, Beijing, China, July 26-31, 2015, pages 536-541.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. UiO_2: Sequence-Labeling Negation Using Dependency Features. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*, Montreal, Canada, June 7-8, 2012, pages 319–327.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1989, 1(4): 541-551.

Junhui Li, Guodong Zhou, Hongling Wang, and Qiaoming Zhu. 2010. Learning the Scope of Negation via Shallow Semantic Parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, Beijing, 2010, pages 671-679.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based Convolutional Neural Networks for Sentence Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers) (ACL2015)*, Beijing, China, 2015, pages 174-179.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding Source Language with Convolutional Neural Network for Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL2015)*, Beijing, China, July 26-31, 2015, pages 20-30.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 2013(26): 3111-3119.

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the Scope of Negation in Biomedical Texts. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, Honolulu, 2008, pages 715-724.

Roser Morante and Walter Daelemans. 2009a. Learning the Scope of Hedge Cues in Biomedical Texts. In *Proceedings of the Workshop on BioNLP*. Boulder, Colorado, 2009, pages 28-36.

Roser Morante and Walter Daelemans. 2009b. A Metalearning Approach to Processing the Scope of Negation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009)*, Boulder, Colorado, June 2009, pages 21-29.

Thien Huu Nguyen and Ralph Grishman. 2015. Event Detection and Domain Adaptation with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL2015)*, Beijing, China, July 26-31,2015, pages 365-371.

Arzucan Özgür and Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, Singapore, 2009, pages 1398-1407.

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010. Syntactic Scope Resolution in Uncertainty Analysis.

In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, Beijing, August 2010, pages 1379-1387.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: Annotation for Negation, Uncertainty and their Scope in Biomedical Texts. In *Proceedings of BioNLP 2008: Current Trends in Biomedical Natural Language Processing*, Columbus, Ohio, USA, 2008, pages 38-45.

Kai Sheng Tai, Richard Socher and Christopher D. Manning. 2015. Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, Beijing, China, July 26-31, 2015, pages 1556-1566.

Buzhou Tang, Xiaolong Wang, XuanWang, Bo Yuan, and Shixi Fan. 2010. A Cascade Method for Detecting Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL 2010): Shared Task*, Uppsala, Sweden, 15-16 July 2010, pages 13-17.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax annotation for the GENIA corpus. In *Proceedings of IJCNLP 2005*, Jeju Island, Korea, October 2005, pages 222-227.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and Negation: Rules, Rankers, and the Role of Syntax. *Computational Linguistics*, 2012, 38(2): 369-410.

Andreas Vlachos and Mark Craven. 2010. Detecting Speculative Language Using Syntactic Dependencies and Logistic Regression. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL 2010): Shared Task*, Uppsala, Sweden, 15-16 July 2010, pages 18-25.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal, 2015, pages 536-540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, Zhi Jin. 2015b. Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal, 2015, pages 1785-1794.

Matthew D. Zeiler and Rob Fergus. 2013. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *arXiv preprint arXiv: 1301.3557v1*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation Classification via Convolutional Deep Neural Network. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2014)*, Dublin, Ireland, August 23-29, 2014, pages 2335-2344.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal, 17-21 September 2015, pages 2230-2235.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2013. Tree Kernel-based Negation and Speculation Scope Detection with Structured Syntactic Parse Features. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, Seattle, Washington, USA, 2013, pages 968-976.

Analyzing Linguistic Knowledge in Sequential Model of Sentence

Peng Qian Xipeng Qiu* Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pqian11, xpqiu, xjhuang}@fudan.edu.cn

Abstract

Sentence modelling is a fundamental topic in computational linguistics. Recently, deep learning-based sequential models of sentence, such as recurrent neural network, have proved to be effective in dealing with the non-sequential properties of human language. However, little is known about how a recurrent neural network captures linguistic knowledge. Here we propose to correlate the neuron activation pattern of a LSTM language model with rich language features at sequential, lexical and compositional level. Qualitative visualization as well as quantitative analysis under multilingual perspective reveals the effectiveness of gate neurons and indicates that LSTM learns to allow different neurons selectively respond to linguistic knowledge at different levels. Cross-language evidence shows that the model captures different aspects of linguistic properties for different languages due to the variance of syntactic complexity. Additionally, we analyze the influence of modelling strategy on linguistic knowledge encoded implicitly in different sequential models.

1 Introduction

Sentence modelling is a central and fundamental topic in the study of language generation and comprehension. With the application of popular deep learning methods, researchers have found that recurrent neural network can successfully model the non-sequential linguistic properties with sequential data input (Vinyals et al., 2015; Zhou and Xu, 2015; Rocktäschel et al., 2015). However, due to the complexity of the neural networks and the lack of effective analytic methodology, little is known about how a sequential model of sentence, such as recurrent neural network, captures linguistic knowledge. This makes it hard to understand the underlying mechanism as well as the model's strength and weakness. Previous work (Li et al., 2016) has attempted to visualize neural models in NLP, but only focus on analyzing the hidden layer and sentiment representation rather than grammar knowledge.

Currently, there have been a few attempts (Yogatama et al., 2014; Köhn, 2015; Faruqui and Dyer, 2015) at understanding what is embedded in the word vectors or building linguistically interpretable embeddings. Few works focus on investigating the linguistic knowledge encoded in a sequential neural network model of a sentence, not to mention the comparison of model behaviours from a cross-language perspective. Our work, therefore, aims to shedding new insights into the following topics:

a) How well does a sequential neural model (e.g. language model) encodes linguistic knowledge of different levels?
b) How does modelling strategy (e.g. the optimization objective) influence the neuron's ability of capturing linguistic knowledge?
c) Does the sequential model behave similarly towards typologically diverse languages?

To tackle the questions above, we propose to visualize and analyze the neuron activation pattern

*Corresponding author.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 826–835,
Austin, Texas, November 1-5, 2016.

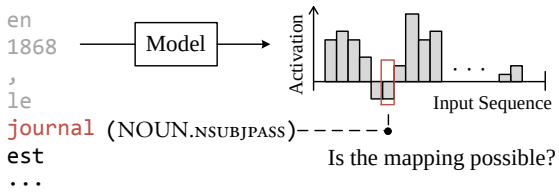


Figure 1: Experiment paradigm: correlating the dynamic activation pattern of the model neurons with linguistic features .

ID	(Non-)Linguistic Knowledge	Level
I	Sequence Length	Sequential
II	Gender / Definiteness Part-of-Speech	Lexical
III	Case / VerbForm / Mood Syntactic Role	Compositional

Table 1: List of the linguistic features to be correlated with model neuron behaviours.

so as to understand how a sequential neural model of sentence encodes linguistic properties of different level. By training vanilla LSTM language models with multilingual data and correlating the model neuron's activation with various linguistic features, we not only qualitatively show the activation pattern of a certain model neuron, but also quantify the selectivity of the neuron towards input language data or certain linguistic properties.

2 Methodology

2.1 A 'Brain' Metaphor of Artificial Model

Mitchell et al. (2008) correlates brain activities with linguistic stimuli under a popular brain-mapping paradigm. Since brain is a 'black box', researchers want to decode what is represented in a certain neuronal cluster of the brain at a certain time step. Here we propose that this paradigm can be applied to similar 'black-box' model, such as the neural network. This is what we call a 'brain' metaphor of the artificial model, as is visualized in Figure 1. We treat the neural network as a simplified 'brain'. We correlate the neuron behaviours with the input stimuli and design experiments to map the neuron activation to an explicit linguistic feature.

A sentence is, of course, a linear sequential arrangement of a cluster of words, but more than just a simple addition of words, as there exist complicated non-sequential syntactic relations. Thus, we consider three levels of features in the analysis of model behaviours, a) Sequential feature, a kind of superficial feature shared by any sequence data, b) Lexical feature, which is stable and almost independent of the sentence context, and c) Compositional feature, which is required for building the meaning of a sentence. Table 1 lists the details of the features involved in this paper.

2.2 Model Description

Since the goal is to understand the internal neurons' behaviour and how the behaviour patterns can be interpreted as a way to encode dynamic linguistic knowledge, we choose the most fundamental sequential sentence models as the research objects. We do not consider tree-structured model, as it explicitly involves linguistic structure in model architecture. We focus on word-based language model and compare it to two other counterparts in this paper.

Word-based Language Model Word-based language model (Mikolov et al., 2010) predicts the incoming word given the history context.

Character-based Language Model Instead of predicting the next word, character-based language model (Hermans and Schrauwen, 2013) predicts the incoming character given the history character sequence.

Task-specific Model A common task-specific model takes word sequence as input, but only predicts the category (e.g. sentiment) of the sentence after all the words are processed. In this paper, we consider a sequential model utilized for sentiment analysis task.

All the three sequential models are built on recurrent neural network with LSTM unit (Hochreiter and Schmidhuber, 1997). LSTM unit has a memory cell c and three gates: input gate i, output gate o and forget gate f, in addition to the hidden layer h of a vanilla RNN. The states of LSTM are updated as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o), \quad (4)$$

$$h_t = o_t \odot \tanh(c_t), \quad (5)$$

where x_t is the input vector at the current time step, σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

The dimension of the embeddings and the LSTM unit is 64. All three models use pretrained word embedding from Polyglot multilingual embeddings (Al-Rfou et al., 2013) trained with C&W (Collobert et al., 2011) model on Wikipedia. We train a lot of word-based and character-based LSTM language models with multilingual data from the Universal Treebank 1.2 (Joakim Nivre and Zhu, 2015), as well as a task-specific sentiment model on Stanford Sentiment Treebank (Socher et al., 2013b). We separate the training and testing data according to 90%/10% principle. We stop training when the loss of the test data does not decrease.

Regarding the analysis of the model behaviours, we collect the internal neuron activation of the hidden layer, three gates, and memory cell for all the data in the treebank/sentiment corpus[1]. For the sake of notation, we refer the hidden layer, input gate, output gate, forget gate and memory cell as h, i, f, o, c for three models, word-based language model (WL), character-based language model (CL) and task-specific model for sentiment analysis (SA). We mark the index of the neuron in the superscript and the meta information about the model in the subscript.

3 Qualitative Analysis

3.1 Sequential Feature

Karpathy et al. (2015) finds that some memory neurons of the character language model are selective towards the length of the input sequence. Similar patterns are also observed among the memory neuron activation pattern of the word-level language model as is shown in Figure 2, where deep purple color indicate strong negative activation and deep green color indicate strong positive activation. Moreover, we compute the correlation between the input sequence length and the activation pattern of every single neuron of h, i, f, o, c. Quantitative results in Figure 3 reveal that none of the hidden layer or gate neurons are strongly correlated with this sequential feature.

[1]The analyses cover languages such as English (en), German (de), Latin (la), Ancient Greek (grc), Bulgarian (bg), Spanish (es), Portuguese (pt), Italian (it), French (fr), Dutch (nl), Norwegian (no), Hindi (hi), Slovenian (sl), Hungarian (hu), Indonesian (id) and Chinese (zh).

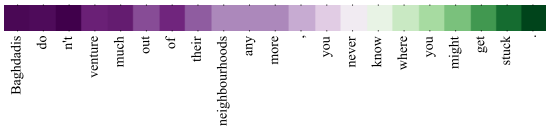


Figure 2: Memory neuron $c^{21}_{en,WL}$ that are sensitive to the length of the input word sequence.

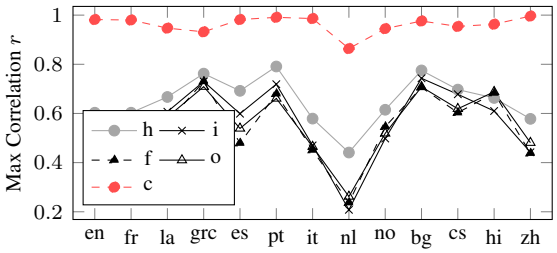
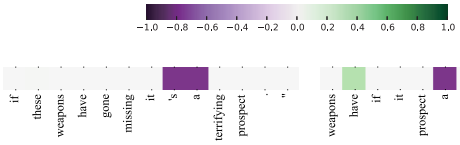


Figure 3: Comparison of neurons on correlating with the length of input sequence. Only the best correlation results are reported.



(a) $h^{30}_{en,WL}$ + raw sent (b) $h^{30}_{en,WL}$ + shuffled sent

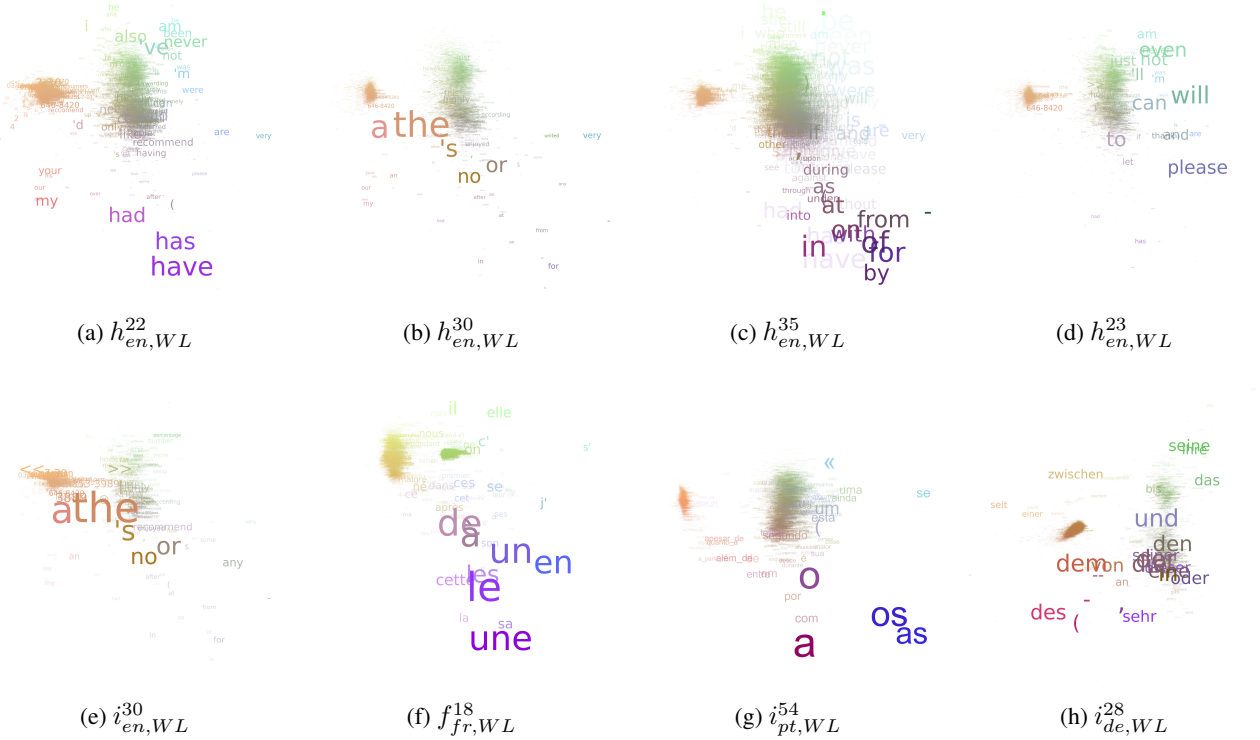
Figure 4: Visualising the activation of a neuron towards raw English sentences and sentences with shuffled word order.

3.2 Lexical Feature

For a inner neuron of the model, we can get the activation of a certain neuron in a certain model component towards a certain input word. A model neuron may be most active towards the words of some category instead of other words.

We notice that some neurons (e.g. Neuron $h^{30}_{en,WL}$) strongly activate towards functional words such as the determiners 'a', as is visualized in Figure 4. This activation can be observed even when we feed the model with a abnormal English sentence with shuffled word order.

Since it is not easy to go through all the neuron activation pattern, we design a visualization method to vividly show how a neuron selectively respond to



(a) $h^{22}_{en,WL}$ (b) $h^{30}_{en,WL}$ (c) $h^{35}_{en,WL}$ (d) $h^{23}_{en,WL}$

(e) $i^{30}_{en,WL}$ (f) $f^{18}_{fr,WL}$ (g) $i^{54}_{pt,WL}$ (h) $i^{28}_{de,WL}$

Figure 5: Visualising the lexical space responded by a certain neuron of the word-level language model trained on different languages.

certain words, inspired by the work in Cukur et al. (2013) and Huth et al. (2016).

Suppose the vocabulary of a language spans a default lexical space. An internal neuron of the artificial model modulates this linguistic space via showing selective activation pattern towards certain words. we carry out PCA on all the word embedding in the language model and extract the most prominent 3 principal components as the bases. Then we project the word vectors onto these three basis to get a new representation of the words in a low dimensional space. We draw all the words on a plane, where the location of each word is determined by the first two components and the text color is determined by three main components as RGB value. To visualize how a target neuron x respond to this lexical space, we modify the appearance of the word by scaling the size of the word text against the product of the log-transformed word frequency and the absolute value of the mean activation, and setting the degree of transparency of the text against the relative positive/negative activation strength among all the existed activation value of a target neuron x. In this way, large font size and low degree of transparency of a word w indicate that the target neuron x frequently activates towards the word w. This means that we can interpret a neuron's selectivity towards the lexical space just by looking for large and explicit words on the visualized two-dimensional space.

Figure 5 visualizes the lexical space responded by four hidden layer neuron of the English language model, as well as four gate neurons of different languages respectively. We can see that words with the similar grammatical functions are located near each other. Besides, it is interesting to see that some hidden layer neurons activate selectively towards determiner words, pronoun, preposition or auxiliary verbs. This phenomena have also been observed on gate neurons. For example, forget gate neuron $f^{18}_{fr,WL}$ activates towards the determiners in French. Input gate neuron $i^{54}_{pt,WL}$ activates towards the determiners in Portuguese. Notice that not all of the inner neurons show interpretable activation pattern towards the lexical space.

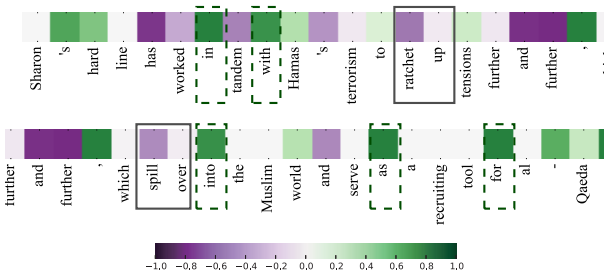


Figure 6: Visualising the Neuron $h^{35}_{en,WL}$ neuron activation towards verb-preposition composition.

3.3 Compositional Feature

To validate whether the internal neuron of the model can discriminate the local composition and long-distance composition, we choose the preposition as the object for observation.

In English, preposition can be combined with the previous verb to form a compound verbal phrase, such as 'check it *in*','give him *up*', 'find *out* what it will take'. This function of the preposition is annotated as the compound particle in the Universal Dependency Treebank. Another function of the preposition is to serve as the case marker, such as the preposition in the phrase 'lives *in* the central area', 'Performances will be performed *on* a project basis'. Given that these two functions of the preposition are not explicitly discriminated in the word form, the language model should tell the difference between the prepositions served as the compound particle and the prepositions served as the case marker if it indeed has the ability to handle word meaning composition.

For the hidden layer, we notice that hidden layer neuron $h^{35}_{en,WL}$ is sensitive to the function of the preposition. It only activates when the possible preposition does not form a composition with the former verb, as is vividly shown in Figure 6. The prepositions marked by dashed box serve as case marker while those in solid box form a phrase with previous verb. The activation pattern are obviously different. Similar pattern is also found in the gate neurons.

4 Quantitative Analysis

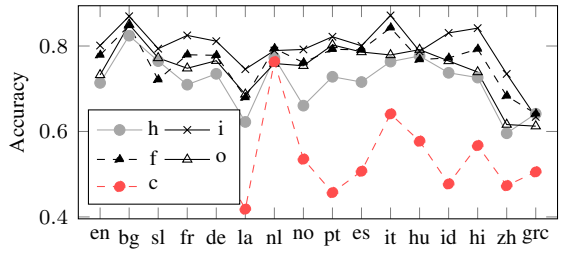
4.1 Decoding Lexical/Compositional Feature

Visualization only provides us with an intuitive idea of what a single neuron is encoding when processing language data. In this section, we employ a mapping paradigm to quantitatively reveal the linguistic knowledge distributed in the model components.

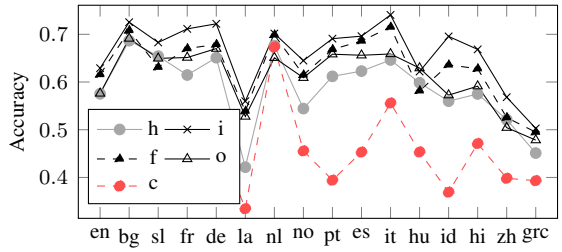
Instead of looking at one single neuron, here we use the whole 64 neurons of each model component as a 64-dimensional vector h, i, f, o, c respectively. The basic method is to decode interpretable linguistic features from target neuron clusters, which has been used in (Köhn, 2015; Qian et al., 2016). We hypothesize that there exists a map between a neuron cluster activation vector x and a high-level sparse linguistic feature vector y if the neuron cluster's activation pattern implicitly encode sufficient information about a certain lexical or compositional feature.

Hence we design a series of experiments to map the hidden layer, three gates, and memory cell vector activated by a target input word w in a sentence to the corresponding linguistic features of the word w, which are annotated in the Universal Dependency Treebank. Our experiments cover POS TAG, SYNTACTIC ROLE, GENDER, CASE, DEFINITENESS, VERB FORM and MOOD. These linguistic features are all represented as a one-hot vector. The mapping model is a simple softmax layer, with the activation vector as the input and the sparse vector as the output. For each linguistic feature of each language, a mapping model is trained on the randomly-selected 90% of all the word tokens and evaluated over the remaining 10%. Notice that GENDER, CASE, DEFINITENESS, VERB FORM, and MOOD only apply to certain word categories. We give a default 'N/A' tag to the words without these annotations so that all the word can be used for training. The evaluation result is only computed from the words with the features. This requires the mapping model to not only recognize the differences between the sub-categories of a linguistic feature (e.g. CASE), but also discriminate the words that we are interested in from other unrelated words (e.g. words without CASE annotations). Accuracies for each model component h, i, f, o, c are reported in Figure 7 and 8.

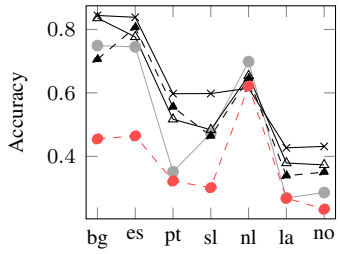
Comparing different model components, we notice that gate neurons except output gate are generally better than hidden layer and memory cell neurons on decoding linguistic knowledge. Input gate and



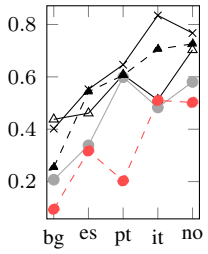
(a) POS TAG



(b) SYNTACTIC ROLE

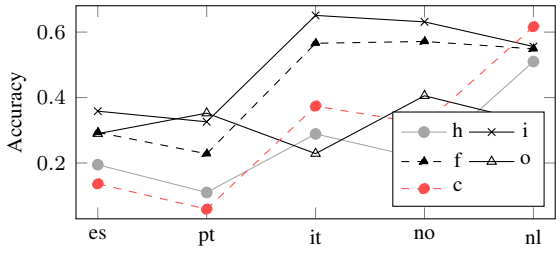


(c) CASE

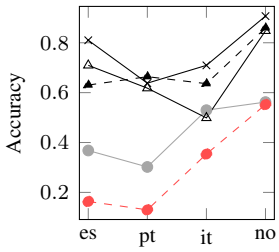


(d) GENDER

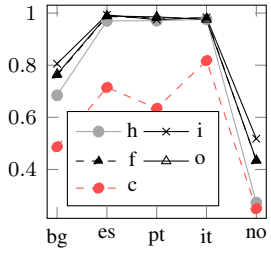
Figure 7: Comparison of neurons on decoding POS AG, SYNTACTIC ROLE, CASE, and GENDER.



(a) VERB FORM



(b) TENSE



(c) DEFINITENESS

Figure 8: Comparison of LSTM neurons on decoding VERB FORM, TENSE, and DEFINITENESS.

forget gate are the best, while memory cell is the worst. It shows that the gates of a recurrent language model are more sensitive to the grammar knowledge of the input words.

Comparing decoding results on different languages, we find that it is generally easier to decode POS TAG than SYNTACTIC ROLE for all the languages. One interesting thing is that the mapping model works better with Bulgarian, a slavic language, but worse on Norwegian on decoding CASE while the situation is opposite on decoding GENDER. It might be because that gender is a weakened grammatical feature in Bulgarian. Therefore, knowledge about GENDER may not be so important in building the grammatical structure of the Bulgarian language data.

4.2 The Dynamics of Neuron Behaviour

Since sentence meaning is dynamically constructed by processing the input sequence in a word-by-word way, it is reasonable to hypothesize that the linguistic feature of an input word w won't sharply decay in the process. Naturally, we would like to ask whether it is possible to decode, or at least partially infer, a word's property from the neuron behaviours of its context words. Specifically, if the model process a verbal phrase '*spill over*' or '*in the garden*', will the property of the word '*spill*', '*in*' be combined with the following word and decodable from the model neuron activation behaviours towards the following word, or will the property of the word '*over*', '*the garden*' be primed by the previous word and decodable from the model neuron behaviours towards the previous word?

To quantitatively explore this question, we carry out a mapping experiment similar to the previous one. The difference is that here we map the hidden layer, three gates, and memory cell vector activated by a target input word w in a sentence to the corresponding linguistic features of the previous/following word $w_{-2/-1}/w_{+1/+2}$ in a 5-word window context. Results in Figure 9 shows that the linguistic feature POS TAG is partially primed or

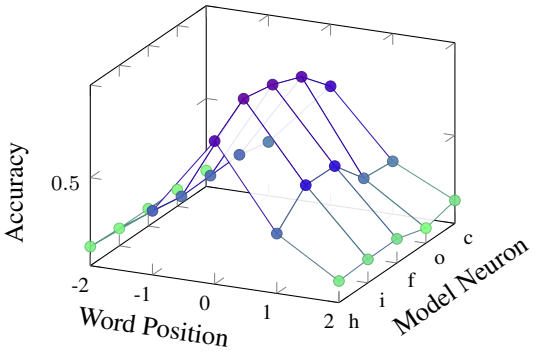


Figure 9: Neuron dynamics on decoding POS.

ISO	**Language**	f	i	o	h
en	English	0.316	-0.022	0.107	0.156
la	Latin	0.152	0.131	0.158	0.085
grc	Ancient Greek	0.293	0.248	0.166	0.274
pt	Portuguese	0.301	0.313	0.161	0.209
nl	Dutch	0.196	0.096	0.205	-0.134
no	Norwegian	0.335	0.057	0.269	0.033
bg	Bulgarian	0.324	0.280	0.071	-0.082

Table 2: Comparison of model components' correlation with tree structure stastistics.

kept in the context words in English. The longer distance, the less probability to decode it from the neuron activations. Still, the nearest context words w_{-1} and w_{+1} prime/keep the most relevant information of the target word w. Similar patterns are also found for other linguistic feature in other languages.

4.3 Correlation with Dependency Tree

Since the sequential model can modelling non-sequential input, we naturally want to know whether any component of the model is dynamically correlated with the statistics of tree structure. Inspired by the case study in Zhou and Xu (2015), we count the syntactic depth of each word in a sentence and compute the correlation between the depth sequence and the dynamics of the average activation of the model neurons in Table 2. We did not find strong correlation between the mean neuron activation dynamics with the syntactic tree depth. One possible explanation is that the language model only use the history information, while the depth of a word is computed in a relative global context.

5 Model Comparison

In this section, we would like to investigate whether different sentence modelling strategy and optimization objective affect the neuron's implicit encoding of linguistic knowledge, especially the grammatical properties.

5.1 Word vs. Character

It is obvious that word-based language model and character-based language model intend to model the language data at different granularity. Although both of them are effective, the latter is often criticized for an unreasonable modelling strategy.

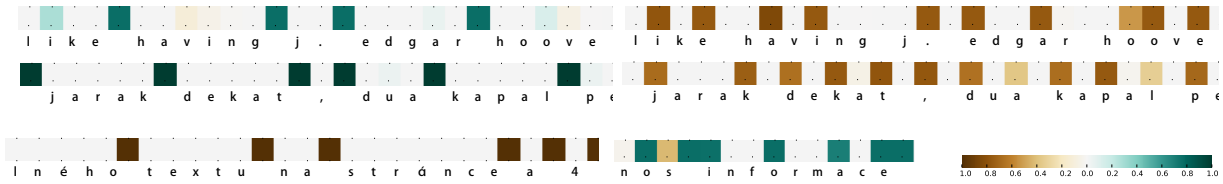
In addition to the findings in Karpathy et al. (2015), we see that some of the hidden layer neurons of the character-based language model seems to be sensitive to specific characters and character clusters, as is indicated from the visualization of the neuron activation pattern in Figure 10. We are surprised to find that some neuron of the hidden layer activates selectively towards white space character. This is interesting as it means that the model learns to detect word boundary, which is exactly an important linguistic feature.

Besides, some neuron activates selectively towards vowel/consonant characters in a phonographic language, such as English. This interesting phenomenon also indicates that the model implicitly captures the phonology system, since it can discriminate the vowel character clusters from the consonant character clusters. We also find these two detectors in other languages, such as Indonesian and Czech in Figure 10.

5.2 Word Prediction vs. Task-specific Model

We compare a word-based LSTM language model and a word-based LSTM sentiment model. Here, for a fair comparison, all the models are trained only on the Stanford Sentiment Treebank Dataset (Socher et al., 2013a). The results show that the neurons in these two models displays similar behaviours towards superficial sequential features, but totally different behaviours towards high-level linguistic features, such as semantic and syntactic knowledge.

Both some of the internal neurons of the memory cell in the language model and the sentiment model emerge to be sensitive to the length of the



(a) $h^{19}_{en,CL}$, $h^{33}_{id,CL}$, $h^{9}_{cs,CL}$ detect white space. (b) $h^{16}_{en,CL}$, $h^{7}_{id,CL}$, $h^{15}_{cs,CL}$ detect phonology.

Figure 10: Visualising the activation of hidden neurons of English, Indonesian and Czech language model.

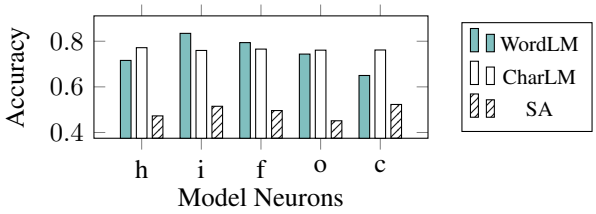


Figure 11: Comparison between the internal neurons of English word-based language model, character-based language model and sentiment model on decoding POS TAG.

input sequence, as we expected, since the sentence length is a non-linguistic features shared by all the sequential input. However, different optimization objectives force the models to represent and capture the linguistic properties of different aspects. The language model focus more on the syntactic aspect, as is visualized and quantified in previous sections. Neurons of the sentiment model tends to be sensitive only towards the sentiment aspect of the words, although the sentiment model use the similar LSTM unit, dimensionality and pretrained embedding. We apply the same visualization method in Section 3.2 to the 64 hidden layer neurons of the sentiment model and manually interpret the visualization results one by one. We did not see any strong activation pattern towards the functional words like those found in language model hidden layer neurons.

To quantify the differences of the linguistic knowledge encoded in different sentential model, we again use the previous feature-decoding experiment method. We compare the performance of the components in three models on decoding POS TAG from English data. Notice that we use Stanford POS Tagger (Kristina Toutanova and Singer, 2003) to automatically tag the sentences in the sentiment data. For the character-based language model, we use the neuron activation towards the end character of each words in the decoding experiment.

Results in Figure 11 shows that even a character-based language model can achieve pretty well on decoding the most important lexical features from the activation pattern of the internal neurons. This is a strong evidence that word-level feature detector can emerge from a pure character-based model. Sentiment model, on the contrary, fails to capture the grammatical knowledge, although we might think that a successful sentiment analysis model should be able to combines the grammar property of the words with the sentiment information. Current results indicate that for pure sequential model with vanilla LSTM units, the objective of the sentence modelling tasks will largely affect how the model acquires and encodes linguistic knowledge.

6 Related Works

Karpathy et al. (2015) explores the memory cell in character-based language model. Their visualization results show some interesting properties of the memory neurons in LSTM unit. However, their exploration on character-based model does not intend to correlate high-level linguistic knowledge, which are intuitively required for sequential modelling of a sentence.

Li et al. (2016) propose a method for visualizing RNN-based sentiment analysis models and word-based LSTM auto-encoder in NLP tasks. Li et al. (2015) investigates the necessity of tree structure for the modelling non-sequential properties of languages. Bowman et al. (2015) studies the LSTM's ability of capturing non-sequential tree structure. Despite the useful findings, these works make no attempts to investigate the internal states of the neurons for a better understanding of the model's power or weakness.

Our work not only provides qualitative visualization of model neurons' behaviours and detailed

quantitative investigation with multilingual evidence (16 for POS decoding experiment), but also reveal the influence of language syntactic complexity and modelling strategy on how well the internal neurons capture linguistic knowledge, which have been overlooked by previous work on interpreting neural network models.

7 Conclusion

In this work, we analyze the linguistic knowledge implicitly encoded in the sequential model of sentence. Through the visualization and quantification of the correlation between the neuron activation behaviour of different model components and linguistic features, we summarize that:

- Model neurons encode linguistic features at different level. Gate neurons encode more linguistic knowledge than memory cell neurons.
- Low-level sequential features are shared across models while high-level linguistic knowledge (lexical/compositional feature) are better captured by language model instead of task-specified model on sentiment analysis.
- Multilingual evidence indicates that the model are sensitive to the syntactic complexity of the language. It would also be a promising direction to incorporate the factor of language typological diversity when designing advanced general sequential model for languages other than English.
- Word-level feature detector can emerge from a pure character-based model, due to the utility of character composition.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches.*

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

Tolga Cukur, Shinji Nishimoto, Alexander G Huth, and Jack L Gallant. 2013. Attention during natural vision warps semantic representation across the human brain. *Nature Neuroscience*, 16(6):763–70.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. *arXiv preprint arXiv:1506.05230*.

M. Hermans and B. Schrauwen. 2013. Training and analysing deep recurrent neural networks. *Advances in Neural Information Processing Systems*, pages 190–198.

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

Alexander G. Huth, Wendy A. De Heer, Thomas L. Griffiths, Frdric E. Theunissen, and Jack L. Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*.

Maria Jesus Aranzabe Masayuki Asahara Aitziber Atutxa Miguel Ballesteros John Bauer Kepa Bengoetxea Riyaz Ahmad Bhat Cristina Bosco Sam Bowman Giuseppe G. A. Celano Miriam Connor Marie-Catherine de Marneffe Arantza Diaz de Ilarraza Kaja Dobrovoljc Timothy Dozat Tomaž Erjavec Richárd Farkas Jennifer Foster Daniel Galbraith Filip Ginter Iakes Goenaga Koldo Gojenola Yoav Goldberg Berta Gonzales Bruno Guillaume Jan Hajič Dag Haug Radu Ion Elena Irimia Anders Johannsen Hiroshi Kanayama Jenna Kanerva Simon Krek Veronika Laippala Alessandro Lenci Nikola Ljubešić Teresa Lynn Christopher Manning Ctlina Mrnduc David Mareček Héctor Martínez Alonso Jan Mašek Yuji Matsumoto Ryan McDonald Anna Missilä Verginica Mititelu Yusuke Miyao Simonetta Montemagni Shunsuke Mori Hanna Nurmi Petya Osenova Lilja Øvrelid Elena Pascual Marco Passarotti Cenel-Augusto Perez Slav Petrov Jussi Piitulainen Barbara Plank Martin Popel Prokopis Prokopidis Sampo Pyysalo Loganathan Ramasamy Rudolf Rosa Shadi Saleh Sebastian Schuster Wolfgang Seeker Mojgan Seraji Natalia Silveira Maria Simi Radu Simionescu Katalin Simkó Kiril Simov Aaron

Smith Jan Štěpánek Alane Suhr Zsolt Szántó Takaaki Tanaka Reut Tsarfaty Sumire Uematsu Larraitz Uria Viktor Varga Veronika Vincze Zdeněk Žabokrtský Daniel Zeman Joakim Nivre, Željko Agić and Hanzhi Zhu. 2015. Universal dependencies 1.2. In *LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague*.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Arne Köhn. 2015. Whats in an embedding? analyzing word embeddings through multilingual evaluation.

Christopher Manning Kristina Toutanova, Dan Klein and Yoram Singer. 2003. Part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

Jiwei Li, Minh Thang Luong, Jurafsky Dan, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *Proceedings of EMNLP*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating language universal and specific properties in word embeddings. In *Proceedings of ACL*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, and Andrew Ng andmChristopher Potts. 2013a. Parsing with compositional vector grammars. In *EMNLP*.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. 2014. Learning word representations with hierarchical sparse coding. *arXiv preprint arXiv:1406.2035*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter

Qi Zhang, Yang Wang, Yeyun Gong, Xuanjing Huang
Shanghai Key Laboratory of Data Science
School of Computer Science, Fudan University
Shanghai, P.R. China
{qz, ywang14, yygong12, xjhuang}@fudan.edu.cn

Abstract

Keyphrases can provide highly condensed and valuable information that allows users to quickly acquire the main ideas. The task of automatically extracting them have received considerable attention in recent decades. Different from previous studies, which are usually focused on automatically extracting keyphrases from documents or articles, in this study, we considered the problem of automatically extracting keyphrases from tweets. Because of the length limitations of Twitter-like sites, the performances of existing methods usually drop sharply. We proposed a novel deep recurrent neural network (RNN) model to combine keywords and context information to perform this problem. To evaluate the proposed method, we also constructed a large-scale dataset collected from Twitter. The experimental results showed that the proposed method performs significantly better than previous methods.

1 Introduction

Keyphrases are usually the selected phrases that can capture the main topics described in a given document (Turney, 2000). They can provide users with highly condensed and valuable information, and there are a wide variety of sources for keyphrases, including web pages, research articles, books, and even movies. In contrast to keywords, keyphrases usually contain two or more words. Normally, the meaning representations of these phrases are more precise than those of single words. Moreover, along with the increasing development of the internet, this kind of summarization has received continuous consideration in recent years from both the academic and entiprise communities (Witten et al., 1999; Wan and Xiao, 2008; Jiang et al., 2009; Zhao et al., 2011; Tuarob et al., 2015).

Because of the enormous usefulness of keyphrases, various studies have been conducted on the automatic extraction of keyphrases using different methods, including rich linguistic features (Barker and Cornacchia, 2000; Paukkeri et al., 2008), supervised classification-based methods (Witten et al., 1999; Wu et al., 2005; Wang et al., 2006), ranking-based methods (Jiang et al., 2009), and clustering-based methods (Mori et al., 2007; Danilevsky et al., 2014). These methods usually focus on extracting keyphrases from a single document or multiple documents. Typically, a large number of words exist in even a document of moderate length, where a few hundred words or more is common. Hence, statistical and linguistic features can be considered to determine the importance of phrases.

In addition to the previously mentioned methods, a few researchers have studied the problem of extracting keyphrases from collections of tweets (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012). In contrast to traditional web applications, Twitter-like services usually limit the content length to 140 characters. In (Zhao et al., 2011), the context-sensitive topical PageRank method was proposed to extract keyphrases by topic from a collection of tweets. NE-Rank was also proposed to rank keywords for the purpose of extracting topical

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 836–845,
Austin, Texas, November 1-5, 2016.

keyphrases (Bellaachia and Al-Dhelaan, 2012). Because multiple tweets are usually organized by topic, many document-level approaches can also be adopted to achieve the task. In contrast with the previous methods, Marujo et al. (2015) focused on the task of extracting keywords from single tweets. They used several unsupervised methods and word embeddings to construct features. However, the proposed method worked on the word level.

In this study, we investigated the problem of automatically extracting keyphrases from single tweets. Compared to the problem of identifying keyphrases from documents containing hundreds of words, the problem of extracting keyphrases from a single short text is generally more difficult. Many linguistic and statistical features (e.g., the number of word occurrences) cannot be determined and used. Moreover, the standard steps of keyphrase extraction usually include keyword ranking, candidate keyphrase generation, and keyphrase ranking. Previous works usually used separate methods to handle these steps. Hence, the error of each step is propagated, which may highly impact the final performance. Another challenge of keyphrase extraction on Twitter is the lack of training and evaluation data. Manual labelling is a time-consuming procedure. The labelling consistency of different labellers cannot be easily controlled.

To meet these challenges, in this paper, we propose a novel deep recurrent neural network (RNN) model for the joint processing of the keyword ranking, keyphrase generation, and keyphrase ranking steps. The proposed RNN model contains two hidden layers. In the first hidden layer, we capture the keyword information. Then, in the second hidden layer, we extract the keyphrases based on the keyword information using a sequence labelling method. In order to train and evaluate the proposed method, we also proposed a novel method to construct a dataset that contained a large number of tweets with golden standard keyphrases. The proposed dataset construction method was based on the hashtag definitions in Twitter and how these were used in specific tweets.

The main contributions of this work can be summarized as follows:

- We proposed a two-hidden-layer RNN-based method to jointly model the keyword ranking, keyphrase generation, and keyphrase ranking steps.
- To train and evaluate the proposed method, we proposed a novel method for constructing a large dataset, which consisted of more than one million words.
- Experimental results demonstrated that the proposed method could achieve better results than the current state-of-the-art methods for these tasks.

2 Proposed Methods

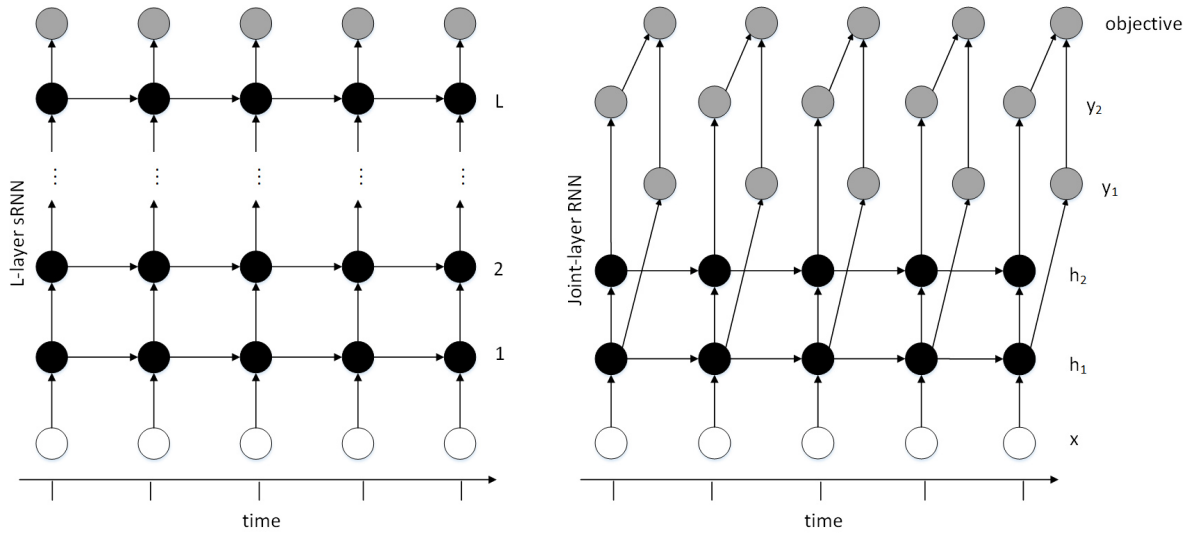
In this paper, we will first describe the deep recurrent neural network (RNN). Then, we will discuss the proposed joint-layer recurrent neural network model, which jointly processes the keyword ranking, keyphrase generation, and keyphrase ranking.

2.1 Deep Recurrent Neural Networks

One way to capture the contextual information of a word sequence is to concatenate neighboring features as input features for a deep neural network. However, the number of parameters rapidly increases according to the input dimension. Hence, the size of the concatenating window is limited. A recurrent neural network (RNN) can be considered to be a deep neural network (DNN) with an indefinite number of layers, which introduces the memory from previous time steps. A potential weakness of a RNN is its lack of hierarchical processing for the input at the current time step. To further provide hierarchical information through multiple time scales, deep recurrent neural networks (DRNNs) are explored (Hermans and Schrauwen, 2013). Fig. 1 (a) shows an L intermediate layer DRNN with full temporal connections (called a stacked RNN (sRNN) in (Pascanu et al., 2013)).

2.2 Joint-layer Recurrent Neural Networks

The proposed joint-layer recurrent neural network (joint-layer RNN) is a variant of an sRNN with two hidden layers. The joint-layer RNN has two output layers, which are combined into a objective layer. Suppose there is an L intermediate layer sRNN that has an output layer for each hidden layer. The l-th



(a) (b)

Figure 1: Deep recurrent neural network (DRNN) architectures: arrows represent connection matrices; white, black, and grey circles represent input frames, hidden states, and output frames, respectively; (a): L intermediate layer DRNN with recurrent connections at all levels (called stacked RNN); (b): joint-layer RNN folded out in time. Each hidden layer can be interpreted to be an RNN that receives the time series of the previous layer as input, where the hidden layer transforms into an output layer. Two output layers are combined via linear superposition into the objective function.

hidden activation is defined as:

$$\begin{aligned} \mathbf{h}_t^l &= f_h(\mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^l) \\ &= \phi_l(\mathbf{U}^l \mathbf{h}_{t-1}^l + \mathbf{W}^l \mathbf{h}_t^{l-1}), \end{aligned} \quad (1)$$

where $\mathbf{h}_t^l$ is the hidden state of the l-th layer at time t. $\mathbf{U}^l$ and $\mathbf{W}^l$ are the weight matrices for the hidden activation at time $t-1$ and the lower level activation $\mathbf{h}_t^{l-1}$, respectively. When $l = 1$, the hidden activation is computed using $\mathbf{h}_t^0 = \mathbf{x}_t$. ϕ_l is an element-wise non-linear function, such as the sigmoid function. The l-th output activation is defined as:

$$\begin{aligned} \hat{\mathbf{y}}_t^l &= f_o(\mathbf{h}_t^l) \\ &= \varphi_l(\mathbf{V}^l \mathbf{h}_t^l), \end{aligned} \quad (2)$$

where $\mathbf{V}^l$ is the weight matrix for the l-th hidden layer $\mathbf{h}_t^l$. φ_l is also an element-wise non-linear function, such as the softmax function.

A joint-layer recurrent neural network is an extension of a stacked RNN with two hidden layers. At time t, the training input, $\mathbf{x}_t$, of the network is the concatenation of features from a mixture within a window. We use word embedding as a feature in this paper. The output targets, $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$, and output predictions, $\hat{\mathbf{y}}_t^1$ and $\hat{\mathbf{y}}_t^2$, of the network indicate whether the current word is a keyword and part of a keyphrase, respectively. $\hat{\mathbf{y}}_t^1$ just has two values $True$ and $False$ indicating whether the current word is keyword. $\hat{\mathbf{y}}_t^2$ has 5 values $Single$, $Begin$, $Middle$, End and Not indicating the current word is a single keyword, the beginning of a keyphrase, the middle (neither beginning nor ending) of a keyphrase, the ending of a keyphrase or not a part of a keyphrase.

Since our goal is to extract a keyphrase from a word sequence, we adopt a framework to simultaneously model keyword finding and keyphrase extraction. Figure 1 (b) shows the architecture of our model. The hidden layer formulation is defined as:

$$\mathbf{h}_t^1 = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}^1) \quad (3)$$

$$\mathbf{h}_t^2 = f_h(\mathbf{h}_t^1, \mathbf{h}_{t-1}^2). \quad (4)$$

The output layer formulation is defined as:

$$\hat{\mathbf{y}}_t^1 = f_o(\mathbf{h}_t^1) \quad (5)$$

$$\hat{\mathbf{y}}_t^2 = f_o(\mathbf{h}_t^2). \quad (6)$$

2.3 Training

In this work, we joined learning the parameters θ in the deep neural network.

$$\boldsymbol{\theta} = \{\mathbf{X}, \mathbf{W}^1, \mathbf{W}^2, \mathbf{U}^1, \mathbf{U}^2, \mathbf{V}^1, \mathbf{V}^2\},$$

where $\mathbf{X}$ are the words embeddings, the other parameters are defined before. Once give a labeled sentence we can know both the keyword and keyphrase (keyphrase is made of keywords). At the first output layer we use our model to discriminate keyword and at the second output layer we use our model to discriminate keyphrase. Then we combine these two sub-objective which at different discrimination level into the final objective. The final objection is defined as:

$$J(\boldsymbol{\theta}) = \alpha J_1(\boldsymbol{\theta}) + (1-\alpha) J_2(\boldsymbol{\theta}), \tag{7}$$

where α is linear weighted factor. Given N training sequences $D = \left\{ \left(\mathbf{x}_t, \mathbf{y}_t^1, \mathbf{y}_t^2\right)_{t=1}^{T_n} \right\}_{n=1}^{N}$, the sub-objective formulation is defined as:

$$J_1(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} d(\hat{\mathbf{y}}_t^1, \mathbf{y}_t^1) \tag{8}$$

$$J_2(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} d(\hat{\mathbf{y}}_t^2, \mathbf{y}_t^2), \tag{9}$$

where $d(\mathbf{a}, \mathbf{b})$ is a predefined divergence measure between **a** and **b**, such as Euclidean distance or cross-entropy.

Eq. (8) and Eq. (9) show that we discover keyword and extract keyphrase at different level simultaneously. The experimental results will show that combination of different granularity discrimination can significantly improve the performance.

To minimize the objective function, we optimize our models by back-propagating the gradients with respect to the training objectives. The stochastic gradient descent (SGD) algorithm is used to train the models. The update rule for the i-th parameter θ_i at epoch e is as follows:

$$\theta_{e,i} = \theta_{e-1,i} - \lambda g_{e,i}, \tag{10}$$

where the λ is a global learning rate shared by all dimensions. g_e is the gradient of the parameters at the e-th iteration. We select the best model according to the validation set.

#tweets	W	T	$\bar{N}_w$	$\bar{N}_t$
41,644,403	147,377	112,515	13.22	1.0

Table 1: Statistical information of dataset. W, T, $\bar{N}_w$, and $\bar{N}_t$ are the vocabulary of words, number of tweets with hashtags, average number of words in each tweet, and average number of hashtags in each tweet, respectively.

3 Experiments

3.1 Data Construction

To analyze the effectiveness of our model for keyphrase extraction on Twitter, we constructed an evaluation dataset. We crawled a large number of tweets. Generally, for each user, we gathered about 3K tweets, with a final total of more than 41 million tweets.

From analyzing these tweets, we found that some of the hashtags can be considered as the keyphrases of the tweet. For example: "The Warriors take Game 1 of the #NBAFinals 104-89 behind a playoff career-high 20 from Shaun Livingston.". "NBA Finals" can be considered as the keyphrase of the twitter. Based on this intuition, to construct the dataset, we firstly filtered out all non-Latin tweets using regular expressions. Then, we removed any URL links from the tweets since we were focusing on the textual content. Tweets that start with the "@username" are generally considered replies and have a conversational nature more than topical nature. Therefore, we also removed any tweets that start with "@username" to focus on topical tweets only. Moreover, we designed some rules about the hashtags in tweets to filter the remaining tweets. First, one tweet could have only one hashtag. Second, the position of the hashtag had to be inside the tweet because we needed the hashtag and tweet to be semantically inseparable. When a hashtag appears inside a tweet, it is most likely to be an inseparable semantical part of the tweet and has important meaning. Therefore, we regarded this hashtag as a keyphrase of the tweet.

Each hashtag was split into keywords if it encompassed more than one word, for example "OldStockCanadians" for "Old Stock Canadians". After an effort to filter the tweets we finally had 110K tweets with the hashtags which could meet our

Algorithm 1 Twitter Dataset Construction

Require: Tweets list $tList$
Ensure: Filtered Tweets and hashtags

```
resultList ← ∅
while t in tList do
    if t not contains latin letters then
        continue
    end if
    if t starts with "@username" then
        continue
    end if
    removed any URL links from t
    if t not exactly contains one hashtag then
        continue
    end if
    get hashtag from t
    split hashtag into keywords
    resultList.append((t, hashtag))
end while
return resultList
```

needs. The pseudocode is defined in Alg. 1. The statistical information of the dataset can be seen in Table 1. To evaluate the quality of the tweets in our dataset, we randomly selected 1000 tweets from our dataset and chose three volunteers. Every tweet was assigned a score of 2 (perfectly suitable), 1 (suitable), or 0 (unsuitable) to indicate whether the hashtag of the tweet was a good keyphrase for it. The results showed that 90.2% were suitable and 66.1% were perfectly suitable. This demonstrated that our constructed dataset was good for keyphrase extraction on Twitter.

3.2 Experiment Configurations

To perform an experiment on extracting keyphrases, we used 70% as a training set, 10% as a development set, and 20% as a testing set. For evaluation metrics, we used the precision (P), recall (R), and F1-score (F1) to evaluate the performance. The precision was calculated based on the percentage of keyphrases truly identified among the keyphrases labeled by the system. Recall was calculated based on the keyphrases truly identified among the golden standard keyphrases.

In the experiments, we use word embeddings as input to the neural network. The word embeddings we used in this work were pre-trained vectors trained on part of a Google News dataset (about 100 billion words). A skip-gram model (Mikolov et al., 2013) was used to generate these 300-dimensional vectors for 3 million words and phrases. We used the word embeddings to initialize our word weight matrix. The matrix was updated in the training process.

The default parameters of our model are as follows: The window size is 3, number of neurons in the hidden layer is 300, and α is 0.5, which were chosen based on the performance using the valid set.

3.3 Methods for Comparison

Several algorithms were implemented and used to evaluate the validity of the proposed approach. Among these algorithms, CRF, RNN, LSTM, and R-CRF treat the keyphrase extraction task as a sequence labelling task. Automatic keyword extraction on Twitter (AKET) uses an unsupervised method to extract keywords on Twitter.

- **CRF**: The keyphrase extraction task can be formalized as a sequence labeling task that involves the algorithmic assignment of a categorical label to each word of a tweet. CRF is a type of discriminative undirected probabilistic graphical model and can process a sequence labeling task. Hence, we applied CRF to extract keyphrases on Twitter.

- **RNN**: A recurrent neural network (RNN) is a type of artificial neural network where the connections between units form a directed cycle. This creates an internal state of the network that allows it to exhibit dynamic temporal behavior. In an RNN model, word embedding is introduced to represent the semantics of words.

- **LSTM**: Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture. Unlike traditional RNNs, an LSTM network is well-suited to learn from experience to classify, process, and predict time series when there are very long time lags of unknown size between important events.

- **R-CRF**: A recurrent conditional random field (R-CRF)(Yao et al., 2014) is a mixture model

	P	R	F1
CRF	72.37%	71.82%	72.09%
RNN	78.65%	70.08%	74.14%
LSTM	77.52%	71.19%	74.22%
R-CRF	79.29%	73.15%	76.10%
AKET	11.00%	46.10%	17.80%
Joint-layer RNN	**80.74**%	**81.19**%	**80.97**%

Table 2: Keyphrase Extraction on Twitter

combining an RNN and a CRF. This model has the advantages of both the CRF and RNN. The previous work showed that the performance of R-CRF can be significantly improved.

- **AKET** (Automatic Keyword Extraction on Twitter) (Marujo et al., 2015): Several unsupervised methods and word embeddings were used to construct features to obtain keyword.

3.4 Experiment Results

Table 2 shows the performances of different methods on the dataset for keyphrase extraction. From the results, we observe that the joint-layer RNN achieved a better performance than the state-of-the-art methods. The relative improvement in the F-score of the joint-layer RNN over the second best result was 6.1%. AKET performed the worst. This was because AKET worked on the word level. Of the other methods, CRF performed the worst, RNN and LSTM were almost the same but better than CRF, and R-CRF was the best of these methods, with the exception of our joint-layer RNN. The results can be explained by the word embedding and long short-term memory cell providing some benefits. The best result was found with our joint-layer RNN. This indicated that the joint processing of the keyword finding and keyphrase extraction worked well and could to some degree demonstrate the effectiveness of our model in keyphrase extraction on Twitter.

To further analyze the keyword extraction results on Twitter, we compared AKET and our method. In Table 3, we can see that except for the recall, AKET is a little better than our method, but our method performed significantly better than AKET in the precision and F-score. This indicates that our model indeed has better performance in keyword finding.

	P	R	F1
AKET	20.68%	**87.56**%	33.46%
Joint-layer RNN	**87.45**%	85.38%	**86.40**%

Table 3: Keyword Extraction on Twitter

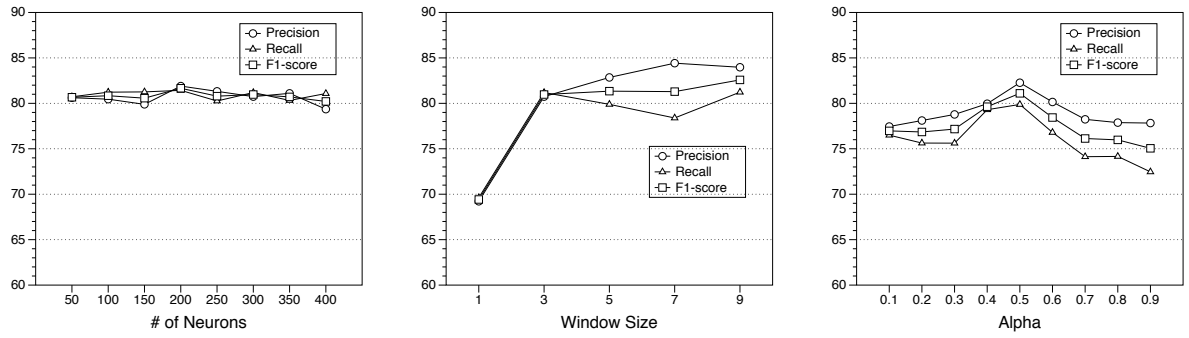
In summary, the experimental results conclusively demonstrated that the proposed joint-layer RNN method is superior to the state-of-the-art methods when measured using commonly accepted performance metrics on Twitter.

To analysis the sensitivity of the hyper-parameters of the joint-layer RNN, we conducted several empirical experiments on the dataset.

Fig.2(a) shows the performances of the joint-layer RNN with different numbers of neurons in the hidden layers. To simplify, we made hidden layer 1 and hidden layer 2 have the same number of neurons. In the figure, the x-axis denotes the number of neurons, and the y-axis denotes the precision, recall, and F-score. The data used for constructing the test set were the same as we used in the previous section. From the figure, we can observe that the number of neurons in the hidden layers do not highly affect the final performance. Three performance indicators of the joint-layer RNN change stably with different numbers of neurons.

Fig.2(b) shows the performances of the joint-layer RNN with different window sizes. In the figure, the x-axis denotes the different window size, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that when the window size is one, the three performance indicators of joint-layer RNN perform badly. Then, as the window size increases, the three performance indicators change stably. The main reason may possibly be that when the window size is one, the model just uses the current word information. When the window size increases, the model uses the context information of the current word but the most important context information is nearby the current word.

Fig.2(c) shows the performances of the joint-layer RNN with different α values. In the figure, the x-axis denotes the value of α used for training, and the y-axis denotes the precision, recall, and F-score.



(a) (b) (c)

Figure 2: (a): Performance with varying number of neurons in the hidden layer; (b): Performance with varying window size; (c): Performance with varying α.

	P	R	F1
WEU	80.74%	81.19%	80.97%
WENU	74.10%	69.30%	71.62%
REU	79.01%	79.75%	79.38%
RENU	78.16%	64.55%	70.70%

Table 4: Effects of embedding on performance. WEU, WENU, REU and RENU represent word embedding update, word embedding without update, random embedding update and random embedding without update respectively.

We can see that the best performance is obtained when α is around 0.5. This indicates that our model emphasizes the combination of keyword finding and keyphrase extraction.

Table 4 lists the effects of word embedding. We can see that the performance when updating the word embedding is better than when not updating, and the performance of word embedding is a little better than random word embedding. The main reason is that the vocabulary size is 147,377, but the number of words from tweets that exist in the word embedding trained on the Google News dataset is just 35,133. This means that 76.2% of the words are missing. This also confirms that the proposed joint-layer RNN is more suitable for keyphrase extraction on Twitter.

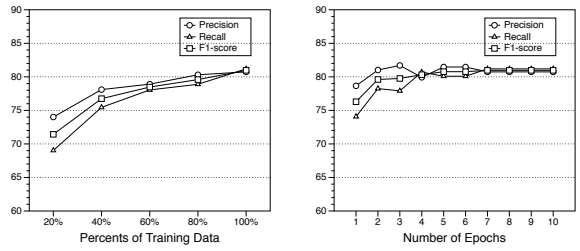
Fig.3(a) shows the performances of the joint-layer RNN with different percentages of training data. In the figure, the x-axis denotes the percentages of data used for training, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that as the amount of training data increases, the three performance indicators of the joint-layer RNN consequently improve. When the percentage of training data is greater than 60% of the whole dataset, the performance indicators slowly increase. The main reason may possibly be that the number concepts included in these data sets are small. However, on the other hand, we can say that the proposed joint-layer RNN method can achieve acceptable results with a few ground truths. Hence, it can be easily adopted for other data sets.

Since the keyphrase extraction training process is solved using an iterative procedure, we also evaluated its convergence property. Fig.3 (b) shows the precision, recall, and F-score performances of the joint-layer RNN. In the figure, the x-axis denotes the number of epochs for optimizing the model, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that the joint-layer RNN can coverage with less than six iterations. This means that the joint-layer RNN can achieve a stable and superior performance under a wide range of parameter values.

4 Related Work

In general, keyphrase extraction methods can be roughly divided into two groups: supervised machine learning approaches and unsupervised ranking approaches.

In the supervised line of research, keyphrase extraction is treated as a classification problem, in which a candidate must be classified as either a keyphrase (i.e., keyphrases) or not (i.e., non-



(a) (b)

Figure 3: (a): Effects of train size on performance; (b): Effects of the number of epochs on performance.

keyphrases). A classifier needs to be trained using annotated training data. The trained model is then applied to documents for which keyphrases are to be identified. For example (Frank et al., 1999) developed a system called KEA that used two features: tf-idf and first occurrence of the term and used them as input to Naive Bayes (Hulth, 2003) used linguistic knowledge (i.e., part-of-speech tags) to determine candidate sets: potential pos-patterns were used to identify candidate phrases from the text. Tang et al. (2004) applied Bayesian decision theory for keyword extraction. Medelyan and Witten extended the KEA to KEA++, which uses semantic information on terms and phrases extracted from a domain specific thesaurus, thus enhances automatic keyphrase extraction (Medelyan and Witten, 2006).

In the unsupervised line of research, keyphrase extraction is formulated as a ranking problem. A well-known approach is the Term Frequency Inverse Document Frequency (TF-IDF) (Sparck Jones, 1972; Zhang et al., 2007; Lee and Kim, 2008). Measures like term frequencies (Wu and Giles, 2013; Rennie and Jaakkola, 2005; Kireyev, 2009), inverse document frequencies, topic proportions, etc. and knowledge of specific domain are applied to rank terms in documents which are aggregated to score the phrases. The ranking based on tf-idf has been shown to work well in practice (Hasan and Ng, 2010). Mihalcea et al. proposed the TextRank, which constructs keyphrases using the PageRank values obtained on a graph based ranking model for graphs extracted from texts (Mihalcea and Tarau, 2004). Liu et al. proposed to extract keyphrases by adopting a clustering-based approach, which ensures that the document is semantically covered by these keyphrases (Liu et al., 2009). Ali Mehri et al. put forward a method for ranking the words in texts, which can also be used to classify the correlation range between word-type occurrences in a text, by using non-extensive statistical mechanics (Mehri and Darooneh, 2011).

Recurrent neural networks(RNNs) (Elman, 1990) has been applied to many sequential prediction tasks, which is an important class of naturally deep architecture. In NLP, RNNs deal with a sentence as a sequence of tokens and have been successfully applied to various tasks like spoken language understanding (Mesnil et al., 2013) and language modeling (Mikolov et al., 2011). Classical recurrent neural networks incorporate information from preceding, there are kinds of variants, bidirectional RNNs are also useful for NLP tasks, especially when making a decision on the current token, information provided by the following tokens is generally useful.

5 Conclusion

In this work, we proposed a novel deep recurrent neural network (RNN) model to combine keywords and context information to perform the keyphrase extraction task. The proposed model can jointly process the keyword ranking and keyphrase generation task. It has two hidden layers to discriminate keywords and classify keyphrases, and these two sub-objectives are combined into a final objective function. We evaluated the proposed method on a dataset filtered from ten million crawled tweets. The proposed method can achieve better results than the state-of-the-art methods. The experimental results demonstrated the effectiveness of the proposed method for keyphrase extraction on single tweets.

6 Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Advances in Artificial Intelligence*.

Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. Ne-rank: A novel graph-based keyphrase extraction in twitter. In *Proceedings of IEEE CS*.

Marina Danilevsky, Chi Wang, Nihit Desai, Xiang Ren, Jingyi Guo, and Jiawei Han. 2014. Automatic construction and ranking of topical keyphrases on collections of short documents. In *Proceedings of SDM*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*.

Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of COLING*.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Proceedings of NIPS*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*.

Xin Jiang, Yunhua Hu, and Hang Li. 2009. A ranking approach to keyphrase extraction. In *Proceedings of SIGIR*.

Kirill Kireyev. 2009. Semantic-based estimation of term informativeness. In *Proceedings of NAACL*.

Sungjick Lee and Han-joon Kim. 2008. News keyword extraction for topic tracking. In *Proceedings of NCM*.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*.

Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *Proceedings of ACL*.

Olena Medelyan and Ian H Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of JCDL*.

Ali Mehri and Amir H Darooneh. 2011. Keyword extraction by nonextensivity measure. *Physical Review E*.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of INTERSPEECH*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of ACL*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of ICASSP*.

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

Junichiro Mori, Mitsuru Ishizuka, and Yutaka Matsuo. 2007. Extracting keyphrases to represent relations in social networks from web. In *Proceedings of IJCAI*.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv*.

Mari-Sanna Paukkeri, Ilari T Nieminen, Matti Pöllä, and Timo Honkela. 2008. A language-independent approach to keyphrase extraction and evaluation. In *Proceedings of COLING*.

Jason DM Rennie and Tommi Jaakkola. 2005. Using term informativeness for named entity detection. In *Proceedings of SIGIR*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *JDoc*.

Jie Tang, Juan-Zi Li, Ke-Hong Wang, and Yue-Ru Cai. 2004. Loss minimization based keyword distillation. In *Advanced Web Technologies and Applications*.

Suppawong Tuarob, Wanghuan Chu, Dong Chen, and Conrad S Tucker. 2015. Twittdict: Extracting social oriented keyphrase semantics from twitter. *IJCNLP*.

Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*.

Jiabing Wang, Hong Peng, and Jing-song Hu. 2006. Automatic keyphrases extraction from document using neural network. In *Advances in Machine Learning and Cybernetics*.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of DL*.

Zhaohui Wu and C Lee Giles. 2013. Measuring term informativeness in context. In *Proceedings of NAACL*.

Yi-fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. 2005. Domain-specific keyphrase extraction. In *Proceedings of CIKM*.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *Proceedings of ICASSP*.

Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *JDIM*.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming

Li. 2011. Topical keyphrase extraction from twitter.
In *Proceedings of ACL*.

Solving and Generating Chinese Character Riddles

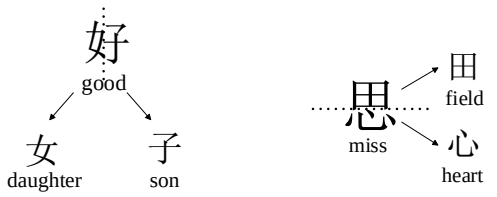
Chuanqi Tan[†] * **Furu Wei**[‡] **Li Dong**[+] **Weifeng Lv**[†] **Ming Zhou**[‡]
[†]State Key Laboratory of Software Development Environment, Beihang University, China
[‡]Microsoft Research Asia [+]University of Edinburgh
[†]tanchuanqi@nlsde.buaa.edu.cn [+]li.dong@ed.ac.uk
[‡]{fuwei, mingzhou}@microsoft.com [†]lwf@buaa.edu.cn

Abstract

Chinese character riddle is a riddle game in which the riddle solution is a single Chinese character. It is closely connected with the shape, pronunciation or meaning of Chinese characters. The riddle description (sentence) is usually composed of phrases with rich linguistic phenomena (such as pun, simile, and metaphor), which are associated to different parts (namely radicals) of the solution character. In this paper, we propose a statistical framework to solve and generate Chinese character riddles. Specifically, we learn the alignments and rules to identify the metaphors between phrases in riddles and radicals in characters. Then, in the solving phase, we utilize a dynamic programming method to combine the identified metaphors to obtain candidate solutions. In the riddle generation phase, we use a template-based method and a replacement-based method to obtain candidate riddle descriptions. We then use Ranking SVM to rerank the candidates both in the solving and generation process. Experimental results in the solving task show that the proposed method outperforms baseline methods. We also get very promising results in the generation task according to human judges.

1 Introduction

The riddle is regarded as one of the most unique and vital elements in traditional Chinese culture, which is usually composed of a riddle description and a corresponding solution. The character riddle is one of the most popular forms of various riddles in which the riddle solution is a single Chinese character. While English words are strings of letters together, Chinese characters are composed of radicals that associate with meaning or metaphor. In other words, Chinese characters are usually positioned into some common structures, such as upper-lower structure, left-right structure, inside-outside structure, which means they can be decomposed into other characters or radicals. For example, "好" (good), a character with left-right structure, can be decomposed into "女" (daughter) and "子" (son). As illustrated in Figure 1(a), the left part of "好" is "女" and the right part is "子". "女" and "子" are called the "radical" of "好". Figure 1(b) is another example of the character "思" (miss) with an upper-lower structure.



(a) Left-Right Structure (b) Upper-Lower Structure

Figure 1: Examples of the structure of Chinese characters

One of the most important characteristics of character riddle lies in the structure of Chinese characters. Unlike the common riddles which imply the object in the riddle descriptions, character riddles pay more attention to structures such as combination of radicals and decomposition of characters. According to these characteristics, metaphors in the

*The work was done when the first author and the third author were interns at Microsoft Research Asia.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 846–855,
Austin, Texas, November 1-5, 2016.

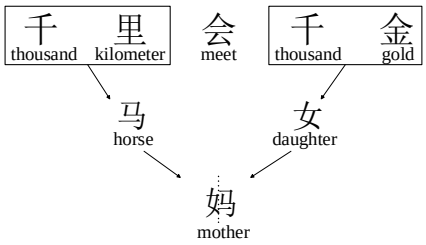


Figure 2: An example of Chinese character riddle: The solution "妈" is composed of the radical "女" derived from "千 金" and "马" derived from "千 里".

riddles always imply the radicals of characters.

We show an example of a Chinese character riddle in Figure 2. The riddle description is "千 里 会 千 金" and the riddle solution is "妈". In this example, "千 里" (thousand kilometer) aligns with "马" (horse) because in Chinese culture it is said that a good horse can run thousands of kilometers per day. Furthermore, "千 金" (thousand gold) aligns with "女" (daughter) because of the analogy that a daughter is very important in the family. The final solution "妈" is composed of these two metaphors because the radical "女" meets the radical "马". Radicals can be derived not only from the meaning of metaphors, but also from the structure of characters. We will describe the alignments and rules in detail in Section 3.

In this paper, we propose a statistical framework to solve and generate Chinese character riddles. We show our pipeline in Figure 3. First, we learn the common alignments and the combination rules from large riddle-solution pairs which are mined from the Web. The alignments and rules are used to identify the metaphors in the riddles. Second, in the solving phase, we utilize a dynamic programming algorithm on the basis of the alignments and rules to figure out the candidate solutions. For the generating phase, we use a template-based method and a replacement-based method based on the decomposition of the character to generate the candidate riddles. Finally, we employ Ranking SVM to rank the candidates in both the solving and generation task. We conduct the evaluation on 2,000 riddles in the riddle solving task and 100 Chinese characters in the riddle generation task. Experimental results show that the proposed method outperforms baseline methods in the solving task. We also get very promising results in the generation task according to human judges.

2 Related Work

To the best of our knowledge, no previous work has studied on Chinese riddles. For other languages, there are a few approaches concentrated on solving English riddles. Pepicello and Green (1984) describe the various strategies incorporated in riddles. (De Palma and Weiner, 1992; Weiner and De Palma, 1993) use the knowledge representation system to solve English riddles that consist of a single sentence question followed by a single sentence answer. They propose to build the relation between the phonemic representation and their associated lexical concepts. Binsted and Ritchie (1994) implement a program JAPE which generates riddles from humour-independent lexical entries and evaluate the behaviour of the program by 120 children (Binsted et al., 1997). Olaosun and Faleye (2015) identify meaning construction strategies in selected English riddles in the web and account for the mental processes involved in their production, which shows that the meaning of a riddle is an imposed meaning that relates to the logical, experiential, linguistic, literary and intuitive judgments of the riddles. Besides, there are some studies in Yoruba(Akínyemí, 2015b; Akínyemí, 2015a; Magaji, 2014). All of these works focus on the semantic meaning, which is different from Chinese character riddles that focus on the structure of characters.

Another popular word game is Crossword Puzzles (CPs) that normally has the form of a square or rectangular grid of white and black shaded squares. The white squares on the border of the grid or adjacent to the black ones are associated with clues. Compared with our riddle task, the clues in the CPs are derived from each question where the radicals in solution are derived from the metaphors in the riddles. Proverb (Littman et al., 2002) is the first system for the automatic resolution of CPs. Ernandes et al. (2005) utilize a web-search module to find sensible candidates to questions expressed in natural language and get the final answer by ranking the candidates. And the rule-based module and the dictionary module are mentioned in his work. The tree kernel is used to rerank the candidates proposed by Barlacchi et al. (2014) for automatic resolution of crossword puzzles.

From another perspective, there are a few projects

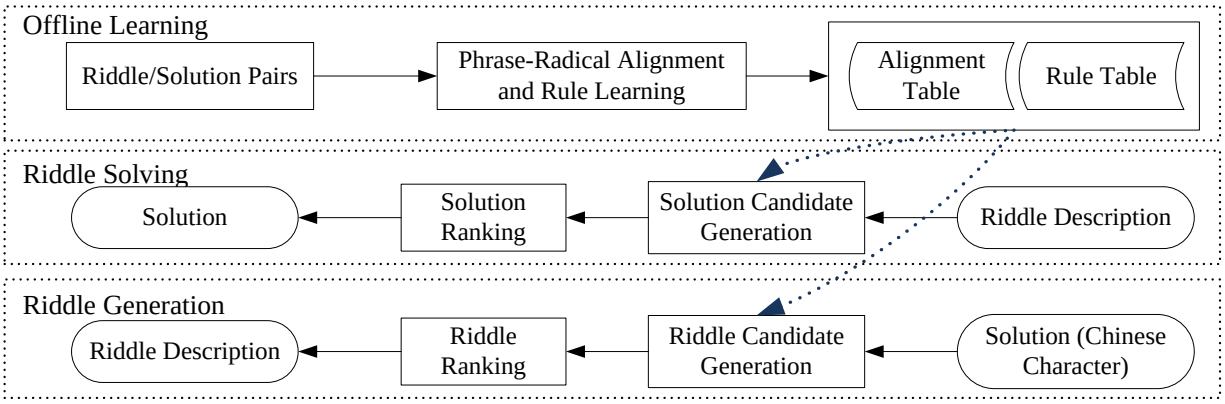


Figure 3: The pipeline of offline learning, riddle solving and riddle generation

on Chinese language cultures, such as the couplet generation and the poem generation. A statistical machine translation (SMT) framework is proposed to generate Chinese couplets and classic Chinese poetry (He et al., 2012; Zhou et al., 2009; Jiang and Zhou, 2008). Jiang and Zhou (2008) use a phrase-based SMT model with linguistic filters to generate Chinese couplets satisfied couplet constraints, using both human judgments and BLEU scores as the evaluation. Zhou et al. (2009) use the SMT model to generate quatrain with a human evaluation. He et al. (2012) generate Chinese poems with the given topic words by combining a statistical machine translation model with an ancient poetic phrase taxonomy. Following the approaches in SMT framework, it is valid to regard the metaphors with its radicals as the alignments. There are several works using neural network to generate Chinese poems(Zhang and Lapata, 2014; Yi et al., 2016). Due to the limited data and strict rules, it is hard to transfer to the riddle generation.

3 Phrase-Radical Alignments and Rules

The metaphor is one of the key components in both solving and generation. On the one hand we need to identify these metaphors since each of them aligns a radical in the final solution. On the other hand, we need to integrate these metaphors into the riddle descriptions to generate riddles. Thus, how to extract the metaphors of riddles becomes a big challenge in our task. Below we introduce our method to extract the metaphors based on the phrase-radical alignments and rules.

We exploit the phrase-radical alignments as to describe the simple metaphors, e.g. "千 里" aligns "马", which aligns the phrase and the radical by the meaning. We employ a statistical framework with a word alignment algorithm to automatically mine phrase-radical metaphors from riddle dataset. Considering the alignment is often represented as the matching between successive words in the riddle and a radical in the solution, we propose two methods specifically to extract alignments. The first method in according with (Och and Ney, 2003) is described as follows. With a riddle description q and corresponding solution s, we tokenize the input riddle q to character as $(w_1, w_2, \ldots, w_n)$ and decompose the solution s into radicals as $(r_1, r_2, \ldots, r_m)$. We count all $([w_i, w_j], r_k)(i, j \in [1, n], k \in [1, m])$ as alignments. The second method takes into account more structural information of characters. Let (w_1, w_2) denote two successive characters in the riddle q. If w_1 is a radical of w_2 and the rest parts of w_2 as r appear in the solution q, we strongly support that $((w_1, w_2), r)$ is a alignment. It is identical if w_2 is a radical of w_1. We count all alignments and filter out the alignments whose occurrence number is lower than 3. Some high-frequency alignments are shown in Table 1. For example, "四方"(square) aligns "口"(mouth) because of the similar shape and "二十载"(two decades) aligns "艹"(grass) because "艹" looks like two small "十"s.

Besides alignments are represented as common collocations, there is another kind of common metaphors concentrating on the structure of characters. We define 6 categories of rules shown in Table 2 to identify this kind of metaphors. A

Bigram Alignments	Radical	Frequency	Trigram Alignments	Radical	Frequency
西 湖 (west lake)	氵 (water)	77	二 十 载 (two decades)	艹 (grass)	21
四 方 (square)	口 (mouth)	40	党 中 央 (center of party)	口 (mouth)	19
千 里 (thousand kilometer)	马 (horse)	36	意 中 人 (sweetheart)	日 (sun)	16

Table 1: The high-frequency alignments

Category	Description	Examples
Half	take half of the matched placeholder as radicals	[半折断边](.) [half,snap,break,side](.)
A-B	remove the B as radical in A to compose a new Chinese character	[减走无缺](.)(.) [subtract,leave,not,lack](.)(.)
UpperRemove	remove the upper-side radical of the matched placeholder	(.)[字]0,1[下南] (.)[character](0,1)[lower,south]
LowerRemove	remove the lower-side radical of the matched placeholder	[首前上北](.) [top,front,up,north](.)
LeftRemove	remove the left-side radical of the matched placeholder	(.)[字]0,1[右东] (.)[character](0,1)[right,east]
RightRemove	remove the right-side radical of the matched placeholder	(.)[字]0,1[左西] (.)[character](0,1)[left,west]

Table 2: The descriptions and examples of rules

rule is often represented as an operation that applies to a character for obtaining parts of it as radicals. For example, the character "上" (up) is usually represented as an operation to get the upper radical of the corresponding character. We extract the rules from the phrase-radical alignments we just obtain. In a phrase-radical alignment, if a radical appears in the one part of a character, we support that this radical is derived from this character, which means the other words in the phrase may describe an operation to this character. We replace this radical to a placeholder and generate a candidate rule with the corresponding direction by the radical position in this character. Thus, for each phrase-radical alignment $([w_1, w_n], r)$, we count $(w_1, \ldots, w_{i-1}, (.), w_{i+1}, \ldots, w_n)$ as a potential rule only if r is a radical of w_i. We count all rules learned from data, and filter out the rules whose occurrence number is lower than 5. Some rules are shown in Table 2. The word or phrase in the rule "A-B" mostly has the analogous meaning of "removing". The word or phrase in the rule "Half" mostly has the analogous meaning of "half". As for the rules "LeftRemove", "RightRemove", "UpperRemove" and "LowerRemove", there are usually a word or phrase that means "removing" as well as the others mean the "position" and "direction".

We mine 14,090 phrase-radical alignments in total. More than 1,000 Chinese characters have at least one alignment, and there are 27 characters with more than 100 alignments. Common radicals are almost all contained in our alignments set. Chinese character is mostly composed of these common radical, so these alignments are enough for our task. We extract 193 rules in total for all categories of rules, all of them are applied to the riddle solving and the riddle generation.

4 Riddle Solving and Generation

4.1 Solving Chinese Character Riddles

The process of solving riddles has two components. First, we identify the metaphors in the riddle as much as possible by matching the phrase-radical alignments and rules, and integrate these metaphors to obtain a candidate set of solutions. Each candidate contains the corresponding parsing clues that imply how and why it is generated as its features. Second, we employ a ranking model to determine the best solution as output. Below we introduce our method to generate solution candidates, and we will

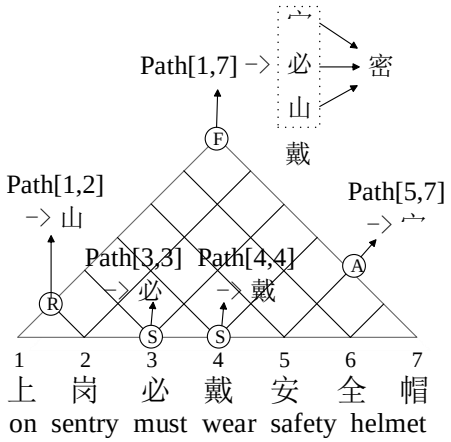


Figure 4: The decoding process of "上 岗 必 戴 安 全 帽". -R: Path[1,2] records the clue that "上 岗" matches "山" by the rule. -S: Path[3,3] records the clue that "必" matches itself and Path[4,4] records "戴". -A: Path[5,7] records the clue that "安 全 帽" matches "宀" by the alignment. -F: We get a final solution candidate in Path[1,7] by above clues. In this example, the character "戴" from Path[4,4] is irrelevant to the solution.

introduce the ranking model in Section 4.3.

It is common that two metaphors do not share a character and the metaphor is composed of successive characters. Therefore, we utilize a dynamic programming algorithm based on the CYK algorithm (Kasami, 1965) to identify the metaphors with the help of the learned alignments and the predefined rules. We describe the algorithm in Algorithm 1.

An example to illustrate our algorithm is "上 岗 必 戴 安 全 帽", where the corresponding solution is "密". As shown in Figure 4, "上 岗"(on sentry) aligns "山" by matching the rule "上(up) (.)" which means to take the upper part of the character "岗". "必" and "戴" aligns itself. And the phrase "安 全 帽"(safety helmet) aligns to the radical "宀" by the alignments because of the analogical shape. Our ranking model will get the final solution "密" by these clues.

4.2 Generating Chinese Character Riddles

Two major components are required in the process of riddle generation. The first step is to generate a list of candidates of riddle descriptions for a Chinese character as the solution. The second step is to rank the candidate riddle descriptions and select the top-N (e.g. 10) candidates as the output. Below we introduce our method to generate candidates of riddle descriptions, and we will introduce the ranking model in Section 4.3.

Algorithm 1: Candidate generation for riddle solving

Input : Riddle q, Alignment, Rule
Output: Path[1,n]

Tokenize the input riddle q to $w_1, w_2, \ldots, w_n$;
for $len \leftarrow 0$ **to** $n-1$ **do**
  for $j - i = len$ **do**
    if *len = 0* **then**
      Character can align itself ;
      $Path[i,j].Add([w_i, w_i] \rightarrow w_i)$;
    end
    else if $[w_i, w_j]$ *in Alignment* **then**
      Obtain the corresponding radical r in Alignment ;
      $Path[i,j].Add([w_i, w_j] \rightarrow r)$;
    end
    else if $[w_i, w_j]$ *matchs Rule* **then**
      Run the predefined operation of the Rule, obtain radical r ;
      $Path[i,j].Add([w_i, w_j] \rightarrow r)$;
    end
    foreach *k in [i,j-1]* **do**
      $Path[i,j].Add(Path[i,k] \oplus Path[k+1,j])$;
    end
  end
end

We propose two strategies to generate the candidate riddle descriptions for a given Chinese character, called the template-based method and the replacement based-method, respectively. First we show our template-based method to generate riddles. The most natural method is to connect the metaphor of each radical. For a character and its possible splitting RD = rd_i, we select a corresponding metaphor by the alignment or rule, and then we connect all metaphor without any other conjunction words to form a riddle. The further method is to add a few conjunction words between each metaphor, which can make the riddle more coherent. We remove the recognized metaphors in riddle sentences,

Feature	Description
Correct_Radical	number of radicals matched
Missing_Radical	number of radicals not matched
Disappearing_Radical	number of radicals that disappear in all characters of riddle descriptions
Single_Matching	number of clues derived from character itself
Alignment_Matching	number of clues derived from alignments
Rule_Matching	number of clues derived from rules
Length_Rate	ratio of the length of clues
Frequency	prior probability of this character as a solution

Table 3: Features for riddle solving

Feature	Description
Riddle_Length	length in characters of the candidate riddle
Riddle_Relative_Length	abs(Riddle_Length-5) because the length of common riddles is between 3 and 7
Number_Radical	number of radicals that the character decompose
Avg_Freq_Character	average number of frequencies of characters in riddle
Max_Freq_Radical	maximized number of frequencies of characters in riddle
Number_Alignment	number of alignments used for generating the candidate
Length_Alignment	length of words from alignments
Number_Rule	number of rules used for generating the candidate
Length_Rule	length of words from rules
LM_Score_R	score of language model trained by Chinese riddles, poems and couplets
LM_Score_G	score of language model trained by web documents

Table 4: Features for riddle generation

and count the unigram and bigram word frequency of the rest words. These words are usually common conjunctions. We sample these words based on the frequency distribution and add them into the riddles to connect the metaphor of each radical.

Second, we use an alternative replacement-based method to generate the candidate riddle descriptions. Instead of generating the riddle descriptions totally from scratch, we try to replacement part of an existing riddle to generate a new riddle description. Let $w = (w_1, w_2, \ldots, w_n)$ denote the word sequence of a riddle description on our dataset, where n denotes the length of the riddle in character. Let $[w_i, w_j]$ (i,j $\in$ [1,n]) denote the word span that can be aligned to a radical rd, and let X=($x_1, \ldots, x_m$) denotes the corresponding phrase descriptions of rd. We then replace $[w_i, w_j] \in X$ with the other alternative phrases descriptions of rd in X. We try all the possible replacements to generate riddle candidates. This method can generate candidate riddles that are more natural and fluent.

4.3 Ranking Model

Above we introduce the algorithm to solve and generate candidates, respectively. Then, we develop a ranking model to determine the final output. Below we show the ranking model.

The ranking score is calculated as

$$Score(c) = \sum_{i=1}^{m} \lambda_i * g_i(c) \quad (1)$$

where c represents a candidate, $g_i(c)$ represents the i-th feature in the ranking model, m represents the number of features in total, and λ_i represents the weight of the feature. The features of riddle solving and riddle generation are in Table 3 and Table 4, respectively. We use Ranking SVM (Joachims, 2006)[1] to do the model training to get the feature weights. The weights of the features are trained with riddle-solution pairs. Specifically, in the riddle solving task, for the set of solution candidates, we hold that the original solution as the positive sample and others are the negative samples. Using the dynamic programming algorithm to obtain a list of solution candidates, the training process try to optimize the feature weights so that the ranking score of the original solution is greater than any of the ones from the

[1] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

candidate list. In the riddle generation task, we select 100 characters on the basis of the frequency distribution of characters as a solution. For each character we use the riddle generation module to generate a list of riddle candidates. And we label these candidates manually where the better riddle descriptions get the higher score. Then the training process optimizes the feature weights.

5 Experimental Study

5.1 Dataset

We crawl 77,308 character riddles including riddle descriptions with its solution from the Web. All of these riddle-solution pairs concentrate on the structure of characters.

A stroke table, that contains 3,755 characters encoded in the first level of GB2312-80, is provided to describe how a Chinese character is decomposed into its corresponding radicals. Characters may have more than one splitting forms and a character is typically composed of no more than 3 radicals.

The data for training language model in riddle style include two parts: One is the corpus of riddles mentioned above, and the other is a corpus of Chinese poem and Chinese couplets because of the similar language style. We follow the method that proposed by (He et al., 2012; Zhou et al., 2009), to download the <Tang Poems>,<Song Poems>, <Ming Poems>, <Qing Poems>, <Tai Poems> from the Internet, and use the method proposed by Fan et al. (2007) to recursively mine those data with the help of some seed poems and couplets. It amounts to more than 3,500,000 sentences and 670,000 couplets. Besides the language model trained in riddle style, we also train a general language model with the web documents.

5.2 Evaluation on Riddle Solving

We randomly select 2,000 riddles from the riddle dataset as the test data, and 500 riddles as the development data, while the rest as training data.

Our system always returns a ranking list of candidate solutions, so we use the Acc@k (k = 1, 5, 10) as the evaluation metric. The Acc@k is the fraction of questions which obtain correct answers in their top-k results.

Giza++ (Och, 2001) is a common tool to extract the alignment between bilingual corpuses. We use it as our baseline system that extracts the alignments automatically. And we use the Jaccard similarity coefficient as the baseline ranking metric. The Jaccard similarity coefficient is defined as:

$$J(A,B) = \frac{A \bigcap B}{A \bigcup B} \quad (2)$$

where A means the radicals set of the solution and B means the radicals set of the candidate.

Feature Set	Acc@1	Acc@5	Acc@10
G	10.3	12.0	13.6
G+A	17.0	19.2	19.9
A	18.7	22.7	24.2
G+A+R	28.4	31.0	31.4
A+R	**28.8**	**31.8**	**32.1**

Table 5: Results of evaluation on test dataset with 2,000 riddles. -G: The alignments from GIZA++. -A: The alignments extracted following our method in Section 3. -R: Using the rules to identify the metaphors between the phrase and the radical following our method in Section 3. Our method (A+R) achieves better performances than the baseline methods from GIZA++.

Ranking Method	Acc@1	Acc@5	Acc@10
Jaccard Similarity	26.2	30.2	31.2
Ranking SVM	**28.8**	**31.8**	**32.1**

Table 6: Results of evaluation between ranking methods using the feature set (A+R). The Ranking SVM achieves better performances than the baseline metric from Jaccard similarity coefficient.

The results are reported in the Table 5 and Table 6. The baseline method can only give about one-tenth correct solution at the Acc@1. Compared with the baseline model, by using the alignments extracted by our method, the system can improve 6.7% at the Acc@1 and 6.3% at Acc@10. A phenomenon is that only using the alignments we extract has the better results than combining it with the alignments from Giza++ because metaphors matching between phrases and characters are particular in our riddle task. Small changes in the phrase can affect the character that it implies and it may be not a metaphor even if a character in phrase is changed. Furthermore, by using rules to identify the metaphors in riddles, we get an improvement of 10.1% at Acc@1, which proves the validity of the

Score	Criterion
5	Elegant metaphors, totally coherent
4	Correct metaphors, mostly coherent
3	Acceptable metaphors, more of less coherent
2	Tolerable metaphors, little coherent
1	Wrong metaphors, incoherent

Table 7: The criterion of riddle evaluation

rule we define. The results prove that it is valid to use the alignments and rules that we extract to identify the metaphors in our character riddle task. The comparison between Jaccard similarity coefficient and our Ranking SVM method shows that the Ranking SVM is better with an improvement of 2.6% at Acc@1, which prove that compared to the Jaccard similarity coefficient, the Ranking SVM determine the solution more correct if we successfully identify all metaphors in riddle descriptions. Moreover, there is less improvement beyond Acc@5, which means the ranking model gets better results even if the system cannot identify all metaphors in riddle descriptions. We think that unlike the Jaccard similarity coefficient which only uses the features between the candidate character and the correct solution, the ranking model uses extra features in the riddles descriptions, e.g. the number of disappearing radicals, which helps to exclude obvious wrong candidates.

5.3 Evaluation on Riddle Generation

Because there is no previous work about Chinese riddle generation, in order to prove its soundness, we conduct human evaluations on this task in accordance with the following two reasons. Firstly, the generated riddles, which is different from the certain and unique solution in the riddle solving task, are varied. So it is hard to measure the quality of generated riddles with a well defined answer set. Secondly, small differences in riddles have a great effect on the corresponding solution. It may imply distinct radicals even if only a character in the metaphors is changed. The existing metrics such as BLEU, are not suitable for our task. Based on above analysis, each riddle that the system generates is evaluated by human annotators according to a 5 division criterion described in Table 7.

We randomly sample 100 characters following the distribution of the character as a solution. The system generates riddle descriptions following the methods in Section 4.2 for each character. Sometimes the riddles we generate exist in our training data. We remove these riddles for the reason that we want to evaluate the ability of generating new riddles. In order to avoid the influence of annotators and compare the riddles generated by the system with the riddles written by human beings, the riddles are randomly disordered so that the annotators do not know the generating method of each riddle. For each character, we select 5 riddles generated by the template-base method, 5 riddles generated by the replacement-based method, and 2 riddles from the riddles dataset written by human beings, which form a set of 12 riddles in total. The annotators score each riddle according to the above criterion.

Method	Avg(Score)
Template-based Method	3.49
Replacement-based Method	4.14
Riddle from dataset	4.38

Table 8: Human evaluation of different methods

The result is shown in Table 8. The riddles written by human beings from the riddle dataset get the highest score than the riddles generated by the system. The riddles generated by the replacement-based method have a greater improvement than the basic template-based method. We consider that the replacement-based method retains some human information, which makes the generated riddles more coherent.

Another result is that the riddle whose solution is a common character or is composed of common radicals gets the higher score, which is explicit that we can get the better results if we have the more alternative metaphors of a radical.

Below we show two examples of the riddle descriptions generated with the solution “思”(miss) which often decompose into “田”(field) and “心”(heart) shown in Figure 1(b).

- 三 星 伴 月 似 画 里 (Three stars with the moon, like in the picture): The radical “田” is the inside part of “画”. The shape of “心” is three points and a curved line, which looks like three stars around a crescent.
- 日 日 相 系 在 心 头 (Every day in my heart):

The radical "田" is composed of two "日"s, and "心" occurs in the riddle description. The character "头"(top) means the radical "田" is on the top position.

6 Conclusion

We introduce a novel approach to solving and generating Chinese character riddles. We extract alignments and rules to capture the metaphors of phrases in riddle descriptions and radicals in the solution characters. In total, we obtain 14,090 alignments that imply the metaphors between phrases and radicals as well as 193 rules in 6 categories formed as regular expressions. To solve riddles, we utilize a dynamic programming algorithm to combine the identified metaphors based on the alignments and rules to obtain the candidate solutions. To generate riddles, we propose a template-based method and a replacement-based method to generate candidate riddle descriptions. We employ the Ranking SVM to rank the candidates on both the riddle solving and generation. Our method outperforms baseline methods in the solving task. We also get promising results in the generation task by human evaluation.

Acknowledgments

The first author and the fourth author are supported by the National Natural Science Foundation of China (Grant No. 61421003).

References

Akíntúndé Akínyemí. 2015a. Riddles and metaphors: The creation of meaning. pages 37–87. Springer.

Akíntúndé Akínyemí. 2015b. Yorùbá riddles in performance: Content and context. In *Orature and Yoruba Riddles*, pages 11–35. Springer.

Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. 2014. Learning to rank answer candidates for automatic resolution of crossword puzzles. In *CoNLL*, pages 39–48.

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. Technical report, University of Edinburgh, Department of Artificial Intelligence.

Kim Binsted, Helen Pain, and Graeme Ritchie. 1997. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition*, 5(2):305–354.

Paul De Palma and E Judith Weiner. 1992. Riddles: accessibility and knowledge representation. In *Proceedings of the 14th conference on Computational linguistics-Volume 4*, pages 1121–1125. Association for Computational Linguistics.

Marco Ernandes, Giovanni Angelini, and Marco Gori. 2005. Webcrow: A web-based system for crossword solving. In *AAAI*, pages 1412–1417.

Cong Fan, Long Jiang, Ming Zhou, and Shi-Long Wang. 2007. Mining collective pair data from the web. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 7, pages 3997–4002. IEEE.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Long Jiang and Ming Zhou. 2008. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 377–384. Association for Computational Linguistics.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Tadao Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report, DTIC Document.

Michael L Littman, Greg A Keim, and Noam Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1):23–55.

Maryam Yusuf Magaji. 2014. Morphology, syntax and functions of the kilba folk riddles. *International Journal on Studies in English Language and LiteratureIJSELL.*

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Franz Josef Och. 2001. Training of statistical translation models.

Ibrahim Esan Olaosun and James Oladunjoye Faleye. 2015. A cognitive semantic study of some english riddles and their answers in amidst a tangled web. *Asian Journal of Social Sciences & Humanities Vol*, 4:2.

William J Pepicello and Thomas A Green. 1984. *Language of riddles: new perspectives*. The Ohio State University Press.

E Judith Weiner and Paul De Palma. 1993. Some pragmatic features of lexical ambiguity and simple riddles. *Language & communication*, 13(3):183–193.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating chinese classical poems with rnn encoder-decoder. *arXiv preprint arXiv:1604.01537*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680.

Ming Zhou, Long Jiang, and Jing He. 2009. Generating chinese couplets and quatrain using a statistical approach. In *PACLIC*, pages 43–52.

Structured prediction models for RNN based sequence labeling in clinical text

Abhyuday N Jagannatha[1], **Hong Yu**[1,2]
[1] University of Massachusetts, MA, USA
[2] Bedford VAMC and CHOIR, MA, USA
abhyuday@cs.umass.edu , hong.yu@umassmed.edu

Abstract

Sequence labeling is a widely used method for named entity recognition and information extraction from unstructured natural language data. In the clinical domain one major application of sequence labeling involves extraction of relevant entities such as medication, indication, and side-effects from Electronic Health Record Narratives. Sequence labeling in this domain presents its own set of challenges and objectives. In this work we experiment with Conditional Random Field based structured learning models with Recurrent Neural Networks. We extend the previously studied CRF-LSTM model with explicit modeling of pairwise potentials. We also propose an approximate version of skip-chain CRF inference with RNN potentials. We use these methods[1] for structured prediction in order to improve the exact phrase detection of clinical entities.

1 Introduction

Patient data collected by hospitals falls into two categories, structured data and unstructured natural language texts. It has been shown that natural text clinical documents such as discharge summaries, progress notes, etc are rich sources of medically relevant information like adverse drug events, medication prescriptions, diagnosis information etc. Information extracted from these natural text documents can be useful for a multitude of purposes ranging from drug efficacy analysis to adverse effect surveillance.

A widely used method for Information Extraction from natural text documents involves treating the text as a sequence of tokens. This format allows sequence labeling algorithms to label the relevant information that should be extracted. Several sequence labeling algorithms such as Conditional Random Fields (CRFs), Hidden Markov Models (HMMs), Neural Networks have been used for information extraction from unstructured text. CRFs and HMMs are probabilistic graphical models that have a rich history of Natural Language Processing (NLP) related applications. These methods try to jointly infer the most likely label sequence for a given sentence.

Recently, Recurrent (RNN) or Convolutional Neural Network (CNN) models have increasingly been used for various NLP related tasks. These Neural Networks by themselves however, do not treat sequence labeling as a structured prediction problem. Different Neural Network models use different methods to synthesize a context vector for each word. This context vector contains information about the current word and its neighboring content. In the case of CNN, the neighbors comprise of words in the same filter size window, while in Bidirectional-RNNs (Bi-RNN) they contain the entire sentence.

Graphical models and Neural Networks have their own strengths and weaknesses. While graphical models predict the entire label sequence jointly, they usually rely on special hand crafted features to provide good results. Neural Networks (especially Re-

[1] Code is available at https://github.com/abhyudaynj/LSTM-CRF-models

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 856–865,
Austin, Texas, November 1-5, 2016.

current Neural Networks), on the other hand, have been shown to be extremely good at identifying patterns from noisy text data, but they still predict each word label in isolation and not as a part of a sequence. In simpler terms, RNNs benefit from recognizing patterns in the surrounding input features, while structured learning models like CRF benefit from the knowledge about neighboring label predictions. Recent work on Named Entity Recognition by (Huang et al., 2015) and others have combined the benefits of Neural Networks(NN) with CRF by modeling the unary potential functions of a CRF as NN models. They model the pairwise potentials as a paramater matrix $[A]$ where the entry $A_{i,j}$ corresponds to the transition probability from the label i to label j. Incorporating CRF inference in Neural Network models helps in labeling exact boundaries of various named entities by enforcing pairwise constraints.

This work focuses on labeling clinical events (medication, indication, and adverse drug events) and event related attributes (medication dosage, route, etc) in unstructured clinical notes from Electronic Health Records. Later on in the Section 4, we explicitly define the clinical events and attributes that we evaluate on. In the interest of brevity, for the rest of the paper, we use the broad term "Clinical Entities" to refer to all medically relevant information that we are interested in labeling.

Detecting entities in clinical documents such as Electronic Health Record notes composed by hospital staff presents a somewhat different set of challenges than similar sequence labeling applications in the open domain. This difference is partly due to the critical nature of medical domain, and partly due to the nature of clinical texts and entities therein. Firstly, in the medical domain, extraction of exact clinical phrase is extremely important. The names of clinical entities often follow polynomial nomenclature. Disease names such as *Uveal melanoma* or *hairy cell leukemia* need to be identified exactly, since partial names (*hairy cell* or *melanoma*) might have significantly different meanings. Additionally, important clinical entities can be relatively rare events in Electronic Health Records. For example, mentions of Adverse Drug Events occur once every six hundred words in our corpus. CRF inference with NN models cited previously do improve exact phrase labeling. However, better ways of modeling the pairwise potential functions of CRFs might lead to improvements in labeling rare entities and detecting exact phrase boundaries.

Another important challenge in this domain is a need to model long term label dependencies. For example, in the sentence *"the patient exhibited **A** secondary to **B**"*, the label for ***A*** is strongly related to the label prediction of ***B***. ***A*** can either be labeled as an adverse drug reaction or a symptom if ***B*** is a Medication or Diagnosis respectively. Traditional linear chain CRF approaches that only enforce local pairwise constraints might not be able to model these dependencies. It can be argued that RNNs may implicitly model label dependencies through patterns in input features of neighboring words. While this is true, explicitly modeling the long term label dependencies can be expected to perform better.

In this work, we explore various methods of structured learning using RNN based feature extractors. We use LSTM as our RNN model. Specifically, we model the CRF pairwise potentials using Neural Networks. We also model an approximate version of skip chain CRF to capture the aforementioned long term label dependencies. We compare the proposed models with two baselines. The first baseline is a standard Bi-LSTM model with softmax output. The second baseline is a CRF model using handcrafted feature vectors. We show that our frameworks improve the performance when compared to the baselines or previously used CRF-LSTM models. To the best of our knowledge, this is the only work focused on usage and analysis of RNN based structured learning techniques on extraction of clinical entities from EHR notes.

2 Related Work

As mentioned in the previous sections, both Neural Networks and Conditional Random Fields have been widely used for sequence labeling tasks in NLP. Specially, CRFs (Lafferty et al., 2001) have a long history of being used for various sequence labeling tasks in general and named entity recognition in particular. Some early notable works include McCallum et. al. (2003), Sarawagi et al. (2004) and Sha et. al. (2003). Hammerton et. al. (2003) and Chiu et. al. (2015) used Long Short Term Memory (LSTM)

(Hochreiter and Schmidhuber, 1997) for named entity recognition.

Several recent works on both image and text based domains have used structured inference to improve the performance of Neural Network based models. In NLP, Collobert et al (2011) used Convolutional Neural Networks to model the unary potentials. Specifically for Recurrent Neural Networks, Lample et al. (2016) and Huang et. al. (2015) used LSTMs to model the unary potentials of a CRF.

In biomedial named entity recognition, several approaches use a biological corpus annotated with entities such as protein or gene name. Settles (2004) used Conditional Random Fields to extract occurrences of protein, DNA and similar biological entity classes. Li et. al. (2015) recently used LSTM for named entity recognition of protein/gene names from BioCreative corpus. Gurulingappa et. al. (2010) evaluated various existing biomedical dictionaries on extraction of adverse effects and diseases from a corpus of Medline abstracts.

This work uses a real world clinical corpus of Electronic Health Records annotated with various clinical entities. Jagannatha et. al. (2016) recently showed that RNN based models outperform CRF models on the task of Medical event detection on clinical documents. Other works using a real world clinical corpus include Rochefort et al. (2015), who worked on narrative radiology reports. They used a SVM-based classifier with bag of words feature vector to predict deep vein thrombosis and pulmonary embolism. Miotto et. al. (2016) used a denoising autoencoder to build an unsupervised representation of Electronic Health Records which could be used for predictive modeling of patient's health.

3 Methods

We evaluate the performance of three different Bi-LSTM based structured prediction models described in section 3.2, 3.3 and 3.4. We compare this performance with two baseline methods of Bi-LSTM(3.1) and CRF(3.5) model.

3.1 Bi-LSTM (baseline)

This model is a standard bidirectional LSTM neural network with word embedding input and a Softmax Output layer. The raw natural language input sentence is processed with a regular expression tokenizer into sequence of tokens $\mathbf{x} = [x_t]_1^T$. The token sequence is fed into the embedding layer, which produces dense vector representation of words. The word vectors are then fed into a bidirectional RNN layer. This bidirectional RNN along with the embedding layer is the main machinery responsible for learning a good feature representation of the data. The output of the bidirectional RNN produces a feature vector sequence $\boldsymbol{\omega}(\mathbf{x}) = [\omega(\mathbf{x})]_1^T$ with the same length as the input sequence $\mathbf{x}$. In this baseline model, we do not use any structured inference. Therefore this model alone can be used to predict the label sequence, by scaling and normalizing $[\omega(\mathbf{x})]_1^T$. This is done by using a softmax output layer, which scales the output for a label l where $l \in \{1, 2, ..., L\}$ as follows:

$$P(\tilde{y}_t = j|\mathbf{x}) = \frac{\exp(\omega(\mathbf{x})_t W_j)}{\sum_{l=1}^{L} \exp(\omega(\mathbf{x})_t W_l)} \quad (1)$$

The entire model is trained end-to-end using categorical cross-entropy loss.

3.2 Bi-LSTM CRF

This model is adapted from the Bi-LSTM CRF model described in Huang et. al. (2015). It combines the framework of bidirectional RNN layer$[\omega(\mathbf{x})]_1^T$ described above, with linear chain CRF inference. For a general linear chain CRF the probability of a label sequence $\tilde{y}$ for a given sentence $\mathbf{x}$ can be written as :

$$P(\tilde{y}|\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{N} \exp\{\phi(\tilde{y}_t) + \psi(\tilde{y}_t, \tilde{y}_{t+1})\} \quad (2)$$

Where $\phi(y_t)$ is the unary potential for the label position t and $\psi(y_t, y_{t+1})$ is the pairwise potential between the positions *t,t+1*. Similar to Huang et. al. (2015), the outputs of the bidirectional RNN layer $\boldsymbol{\omega}(\mathbf{x})$ are used to model the unary potentials of a linear chain CRF. In particular, the NN based unary potential $\phi_{nn}(y_t)$ is obtained by passing $\boldsymbol{\omega}(\mathbf{x})_t$ through a standard feed-forward $tanh$ layer. The binary potentials or transition scores are modeled as a matrix $[A]_{L \times L}$. Here L equals the number of possible labels including the *Outside* label. Each element $A_{i,j}$ represents the transition score from label i to j. The

probability for a given sequence $\tilde{y}$ can then be calculated as :

$$P(\tilde{y}|\mathbf{x};\theta) = \frac{1}{Z}\prod_{t=1}^{T}\exp\{\phi_{nn}(\tilde{y}_t) + A_{\tilde{y}_t,\tilde{y}_{t+1}}\} \quad (3)$$

The network is trained end-to-end by minimizing the negative log-likelihood of the ground truth label sequence $\hat{\mathbf{y}}$ for a sentence $\mathbf{x}$ as follows:

$$\mathcal{L}(\mathbf{x},\hat{\mathbf{y}};\theta) = -\sum_{t}\sum_{y_t}\delta(y_t = \hat{y}_t)\log P(y_t|\mathbf{x};\theta)\} \quad (4)$$

The negative log likelihood of given label sequence for an input sequence is calculated by sum-product message passing. Sum-product message passing is an efficient method for exact inference in Markov chains or trees.

3.3 Bi-LSTM CRF with pairwise modeling

In the previous section, the pairwise potential is calculated through a transition probability matrix $[A]$ irrespective of the current context or word. For reasons mentioned in section 1, this might not be an effective strategy. Some clinical entities are relatively rare. Therefore transition from an *Outside* label to a clinical label might not be effectively modeled by a fixed parameter matrix. In this method, the pairwise potentials are modeled through a non-linear Neural Network which is dependent on the current word and context. Specifically, the pairwise potential $\psi(y_t, y_{t+1})$ in equation 2 is computed by using a one dimensional CNN with 1-D filter size 2 and $tanh$ non-linearity. At every label position t, it takes $[\omega(\mathbf{x})_t; \omega(\mathbf{x})_{t+1}]$ as input and produces a $L \times L$ pairwise potential output $\psi_{nn}(y_t, y_{t+1})$. This CNN layer effectively acts as a non-linear feed-forward neuron layer, which is repeatedly applied on consecutive pairs of label positions. It uses the output of the bidirectional LSTM layer at positions t and $t+1$ to prepare the pairwise potential scores.

The unary potential calculation is kept the same as in Bi-LSTM-CRF. Substituting the neural network based pairwise potential $\psi_{nn}(y_t, y_{t+1})$ into equation 2 we can reformulate the probability of the label sequence $\tilde{y}$ given the word sequence $\mathbf{x}$ as :

$$P(\tilde{y}|\mathbf{x};\theta) = \frac{1}{Z}\prod_{t=1}^{N}\exp\{\phi_{nn}(\tilde{y}_t) + \psi_{nn}(\tilde{y}_t, \tilde{y}_{t+1})\} \quad (5)$$

Labels	Num. of Instances	Avg word length $\pm$ std
ADE	1807	1.68 $\pm$ 1.22
Indication	3724	2.20 $\pm$ 1.79
Other SSD	40984	2.12 $\pm$ 1.88
Severity	3628	1.27 $\pm$ 0.62
Drugname	17008	1.21 $\pm$ 0.60
Duration	926	2.01 $\pm$ 0.74
Dosage	5978	2.09 $\pm$ 0.82
Route	2862	1.20$\pm$ 0.47
Frequency	5050	2.44$\pm$ 1.70

Table 1: Annotation statistics for the corpus.

The neural network is trained end-to-end with the objective of minimizing the negative log likelihood in equation 4. The negative log-likelihood scores are obtained by sum-product message passing.

3.4 Approximate Skip-chain CRF

Skip chain models are modifications to linear chain CRFs that allow long term label dependencies through the use of *skip edges*. These are basically edges between label positions that are not adjacent to each other. Due to these skip edges, the skip chain CRF model (Sutton and McCallum, 2006) explicitly models dependencies between labels which might be more than one position apart. The joint inference over these dependencies are taken into account while decoding the best label sequence. However, unlike the two models explained in the preceding section, the skip-chain CRF contains loops between label variables. As a result we cannot use the sum-product message passing method to calculate the negative log-likelihood.The loopy structure of the graph in skip chain CRF renders exact message passing inference intractable. Approximate solutions for these models include loopy belief propagation(BP) which requires multiple iterations of message passing.

However, an approach like loopy BP is prohibitively expensive in our model with large Neural Net based potential functions. The reason for this is that each gradient descent iteration for a combined RNN-CRF model requires a fresh calculation of the marginals. In one approach to mitigate this, Lin et. al. (2015) directly model the messages in the message passing inference of a 2-D grid CRF for image segmentation. This bypasses the need for modeling the potential function, as well as calculating the ap-

proximate messages on the graph using loopy BP.

Approximate CRF message passing inference: Lin et. al. (2015) directly estimate the factor to variable message using a Neural Network that uses input image features. Their underlying reasoning is that the factor-to-variable message from factor F to label variable y_t for any iteration of loopy BP can be approximated as a function of all the input variables and previous messages that are a part of that factor. They only model one iteration of loopy BP, and empirically show that it leads to an appreciable increase in performance. This allows them to model the messages as a function of only the input variables, since the messages for the first iteration of message passing are computed using the potential functions alone.

We follow a similar approach for calculation of variable marginals in our skip chain model. However, instead of estimating individual factor-to-variable messages, we exploit the sequence structure in our problem and estimate groups of factor-to-variable messages. For any label node y_t, the first group contains factors that involve nodes which occur before y_t in the sentence (from left). The second group of factor-to-variable messages corresponds to factors involving nodes occurring later in the sentence. We use recurrent computational units like LSTM to estimate the sum of log factor-to-variable messages within a group. Essentially, we use bidirectional recurrent computation to estimate all the incoming factors from left and right separately.

To formulate this, let us assume for now that we are using skip edges to connect the current node t to m preceding and m following nodes. Each edge, skip or otherwise, is denoted by a factor which contains the binary potential of the edge and the unary potential of the connected node. As mentioned earlier, we will divide the factors associated with node t into two sets, $F_L(t)$ and $F_R(t)$. Here $F_L(t)$, contains all factors formed between the variables from the group $\{y_{t-m}, ..., y_{t-1}\}$ and y_t. So we can formulate the combined message from factors in $F_L(t)$ as

$$\beta_L(y_t) = [\sum_{F \in F_L(t)} \beta_{F \to t}(y_t)] \tag{6}$$

The combined messages from factors in $F_R(t)$ which contains variables from y_{t+1} to y_{t+m} can be formulated as :

$$\beta_R(y_t) = [\sum_{F \in F_R(t)} \beta_{F \to t}(y_t)] \tag{7}$$

We also need the unary potential of the label variable t to compose its marginal. The unary potentials of each variable from $\{y_{t-m}, ..., y_{t-1}\}$ and $\{y_{t+1}, ..., y_{t+m}\}$ should already be included in their respective factors. The log of the unnormalized marginal $\bar{P}(y_t|\mathbf{x})$ for the variable y_t, can therefore be calculated by

$$\log \bar{P}(y_t|\mathbf{x}) = \beta_R(y_t) + \beta_L(y_t) + \phi(y_t) \tag{8}$$

Similar to Lin et. al. (2015), in the interest of limited network complexity, we use only one message passing iteration. In our setup, this means that a variable-to-factor message from a neighboring variable y_i to the current variable y_t contains only the unary potentials of y_i and binary potential between y_i , y_t. As a consequence of this, we can see that $\beta_L(y_t)$ can be written as :

$$\beta_L(y_t) = \sum_{i=t-1}^{t-m} \log \sum_{y_i} [\exp \psi(y_t, y_i) + \phi(y_i)] \tag{9}$$

Similarly, we can formulate a function for $\beta_R(y_t)$ in a similar way :

$$\beta_R(y_t) = \sum_{i=t+1}^{t+m} \log \sum_{y_i} [\exp \psi(y_t, y_i) + \phi(y_i)] \tag{10}$$

Modeling the messages using RNN: As mentioned previously in equation 8, we only need to estimate $\beta_L(y_t)$, $\beta_R(y_t)$ and $\phi(\mathbf{y}_t)$ to calculate the marginal of variable y_t. We can use $\phi_{nn}(\mathbf{y}_t)$ framework introduced in section 3.2 to estimate the unary potential for y_t. We use different directions of a bidirectional LSTM to estimate $\beta_R(y_t)$ and $\beta_L(y_t)$. This eliminates the need to explicitly model and learn pairwise potentials for variables that are not immediate neighbors.

The input to this layer at position t is $[\phi_{nn}(y_t); \psi_{nn}(y_t, y_{t+1})]$ (composed of potential functions described in section 3.3). This can be viewed as an LSTM layer aggregating beliefs about y_t from the unary and binary potentials of $[y]_1^{t-1}$

	Strict Evaluation (Exact Match)			Relaxed Evaluation (Word based)		
Models	**Recall**	**Precision**	**F-score**	**Recall**	**Precision**	**F-score**
CRF	0.7385	0.8060	0.7708	0.7889	0.8040	0.7964
Bi-LSTM	0.8101	0.7845	0.7971	0.8402	0.8720	0.8558
Bi-LSTM CRF	0.7890	0.8066	0.7977	0.8068	**0.8839**	0.8436
Bi-LSTM CRF-pair	0.8073	**0.8266**	0.8169	0.8245	0.8527	0.8384
Approximate Skip-Chain CRF	**0.8364**	0.8062	**0.8210**	**0.8614**	0.8651	**0.8632**

Table 2: Cross validated micro-average of Precision, Recall and F-score for all clinical tags

to approximate the sum of messages from left side $\beta_L(y_t)$. Similarly, $\beta_R(y_t)$ can be approximated from the LSTM aggregating information from the opposite direction. Formally, $\beta_L(y_t)$ is approximated as a function of neural network based unary and binary potentials as follows:

$$\beta_L(y_t) \approx f([\phi_{nn}(y_i); \psi_{nn}(y_i, y_{i+1})]_1^{t-1}) \quad (11)$$

Using LSTM as a choice for recurrent computation here is advantageous, because LSTMs are able to learn long term dependencies. In our framework, this allows them to learn to prioritize more relevant potential functions from the sequence $[[\phi_{nn}(y_i); \psi_{nn}(y_i, y_{i+1})]_1^{t-1}$. Another advantage of this method is that we can approximate skip edges between all preceding and following nodes, instead of modeling just *m* surrounding ones. This is because LSTM states are maintained throughout the sentence.

The partition function for y_t can be easily obtained by using logsumexp over all label entries of the unnormalized log marginal shown in equation 8 as follows:

$$Z_t = \sum_{y_t} \exp[\beta_R(y_t) + \beta_L(y_t) + \phi(y_t)] \quad (12)$$

Here the partition function Z is a different for different positions of t. Due to our approximations, it is not guaranteed that the partition function calculated from different marginals of the same sentence are equal. The normalized marginal can be now calculated by normalizing $\log \bar{P}(y_t|\mathbf{x})$ in equation 8 using Z_t.

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{y}}; \theta) = -\sum_t \sum_{y_t} \delta(y_t = \hat{y}_t)(\beta_R(y_t; \theta) + \beta_L(y_t; \theta) + \phi(y_t; \theta) - \log Z_t(\theta)) \quad (13)$$

The model is optimized using cross entropy loss between the true marginal and the predicted marginal. The loss for a sentence $\mathbf{x}$ with a ground truth label sequence $\hat{\mathbf{y}}$ is provided in equation 13.

3.5 CRF (baseline)

We use the linear chain CRF, which is a widely used model in extraction of clinical named entities. As mentioned previously, Conditional Random Fields explicitly model dependencies between output variables conditioned on a given input sequence.

The main inputs to CRF in this model are not RNN outputs, but word inputs and their corresponding word vector representations. We add additional sentence features consisting of four vectors. Two of them are bag of words representation of the sentence sections before and after the word respectively. The remaining two vectors are dense vector representations of the same sentence sections. The dense vectors are calculated by taking the mean of all individual word vectors in the sentence section. We add these features to explicitly mimic information provided by the bidirectional chains of the LSTM models.

4 Dataset

We use an annotated corpus of 1154 English Electronic Health Records from cancer patients. Each note was annotated[2] by two annotators who label clinical entities into several categories. These categories can be broadly divided into two groups, Clinical Events and Attributes. Clinical events include any specific event that causes or might contribute to a change in a patient's medical status. Attributes are phrases that describe certain important properties about the events.

[2]The annotation guidelines can be found at https://github.com/abhyudaynj/LSTM-CRF-models/blob/master/annotation.md

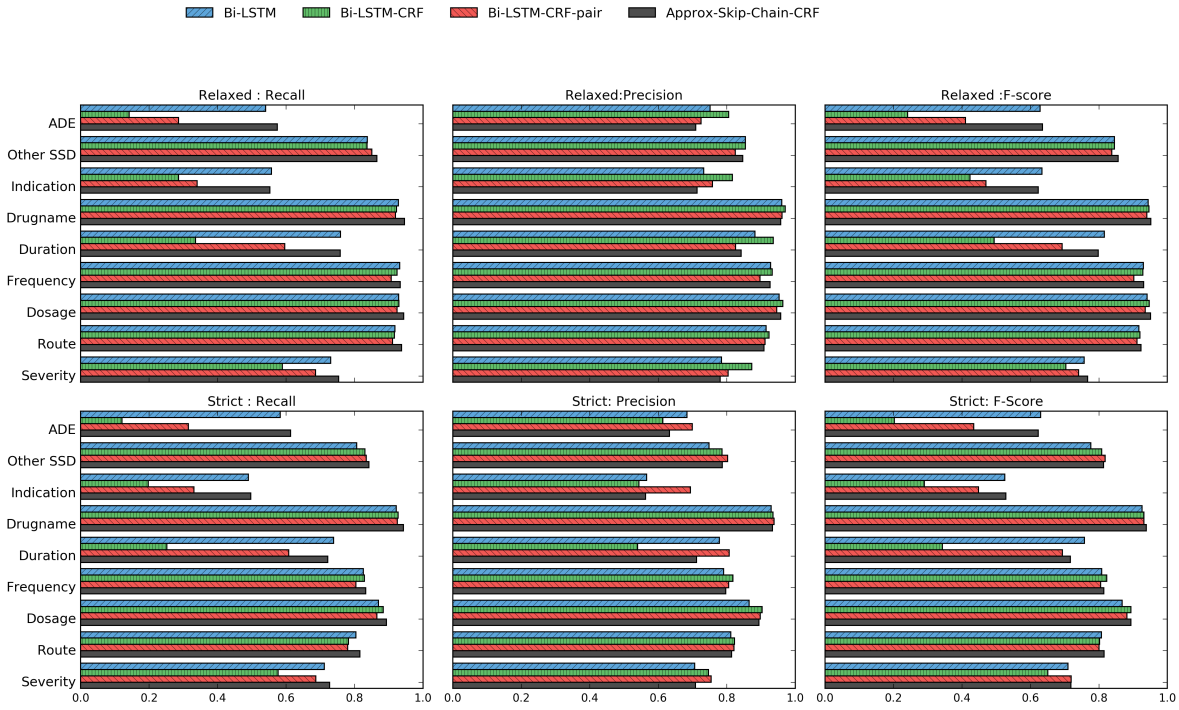


Figure 1: Plots of Recall, Precision and F-score for RNN based methods. The metrics with prefix *Strict* are using phrase based evaluation. *Relaxed* metrics use word based evaluation.Bar-plots are in order with Bi-LSTM on top and Approx-skip-chain-CRF at the bottom.

Clinical Event categories in this corpus are *Adverse Drug Event (ADE)*, *Drugname* , *Indication* and *Other Sign Symptom and Diseases (Other SSD)*. *ADE*, *Indication* and *Other SSD* are events having a common vocabulary of Sign, Symptoms and Diseases (SSD). They can be differentiated based on the context that they are used in. A certain SSD should be labeled as *ADE* if it can be manually identified as a side effect of a drug based on the evidence in the clinical note. It is an *Indication* if it is an affliction that a doctor is actively treating with a medication. Any other SSD that does not fall into the above two categories (for e.g. an SSD in patients history) is labeled as *Other SSD*. *Drugname* event labels any medication or procedure that a physician prescribes.

The attribute categories contain the following properties, *Severity* , *Route*, *Frequency*, *Duration* and *Dosage*. *Severity* is an attribute of the SSD event types , used to label the severity a disease or symptom. *Route*, *Frequency*, *Duration* and *Dosage* are attributes of *Drugname*. They are used to label the medication method, frequency of dosage, duration of dosage, and the dosage quantity respectively. The annotation statistics of the corpus are provided in the Table 1.

5 Experiments

Each document is split into separate sentences and the sentences are tokenized into individual word and special character tokens. The models operate on the tokenized sentences. In order to accelerate the training procedure, all LSTM models use batch-wise training using a batch of 64 sentences. In order to do this, we restricted the sentence length to 50 tokens. All sentences longer than 50 tokens were split into shorter size samples, and shorter sentences were pre-padded with masks. The CRF baseline model(3.5) does not use batch training and so the sentences were used unaltered.

The first layer for all LSTM models was a 200 dimensional word embedding layer. In order to improve performance, we initialized embedding layer values in these models with a skip-gram word embedding (Mikolov et al., 2013). The skip-gram embedding was calculated using a combined corpus

of PubMed open access articles, English Wikipedia and an unlabeled corpus of around hundred thousand Electronic Health Records. The EHRs used in the annotated corpus are not in this unlabeled EHR corpus. This embedding is also used to provide word vector representation to the CRF baseline model.

The bidirectional LSTM layer which outputs $\boldsymbol{\omega}(\mathbf{x})$ contains LSTM neurons with a hidden size ranging from 200 to 250. This hidden size is kept variable in order to control for the number of trainable parameters between different LSTM based models. This helps ensure that the improved performance in these models is only because of the modified model structure, and not an increase in trainable parameters. The hidden size is varied in such a way that the number of trainable parameters are close to 3.55 million parameters. Therefore, the Approx skip chain CRF has 200 hidden layer size, while standard Bi-LSTM model has 250 hidden layer. Since the $\boldsymbol{\omega}(\mathbf{x})$ layer is bidirectional, this effectively means that the Bi-LSTM model has 500 hidden layer size, while Approx skip chain CRF model has 400 dimensional hidden layer.

We use dropout (Srivastava et al., 2014) with a probability of 0.50 in all LSTM models in order to improve regularization performance. We also use batch norm (Ioffe and Szegedy, 2015) between layers wherever possible in order to accelerate training. All RNN models are trained in an end-to-end fashion using Adagrad (Duchi et al., 2011) with momentum. The CRF model was trained using L-BFGS with L2 regularization. We use Begin Inside Outside (BIO) label modifiers for models that use CRF objective.

We use ten-fold cross validation for our results. The documents are divided into training and test documents. From each training set fold, 20% of the sentences form the validation set which is used for model evaluation during training and for early stopping.

We report the word based and exact phrase match based micro-averaged recall, precision and F-score. Exact phrase match based evaluation is calculated on a per phrase basis, and considers a phrase as positively labeled only if the phrase exactly matches the true boundary and label of the reference phrase. Word based evaluation metric is calculated on labels of individual words. A word's predicted label is considered as correct if it matches the reference label, irrespective of whether the remaining words in its phrase are labeled correctly. Word based evaluation is a more relaxed metric than phrase based evaluation.

6 Results

The micro-averaged Precision, Recall and F-score for all five models are shown in Table 2. We report both strict (exact match) and relaxed (word based) evaluation results. As shown in Table 2, the best performance is obtained by Skip-Chain CRF (0.8210 for strict and 0.8632 for relaxed evaluation). All LSTM based models outperform the CRF baseline. Bi-LSTM-CRF and Bi-LSTM-CRF-pair models using exact CRF inference improve the precision of strict evaluation by 2 to 5 percentage points. Bi-LSTM CRF-pair achieved the highest precision for exact-match. However, the recall (both strict and relaxed) for exact CRF-LSTM models is less than Bi-LSTM. This reduction in recall is much less in the Bi-LSTM-pair model. In relaxed evaluation, only the Skip Chain model has a better F-score than the baseline LSTM. Overall, Bi-LSTM-CRF-pair and Approx-Skip-Chain models lead to performance improvements. However, the standard Bi-LSTM-CRF model does not provide an appreciable increase over the baseline.

Figure 1 shows the breakdown of performance for each RNN model with respect to individual clinical entity labels. CRF baseline model performance is not shown in Figure 1, because its performance is consistently lower than Bi-LSTM-CRF model across all label categories. We use pairwise t-test on strict evaluation F-score for each fold in cross validation, to calculate the statistical significance of our scores. The improvement in F-score for Bi-LSTM-CRF-pair and Approx-Skip Chain as compared to Bi-LSTM baseline is statistically significant ($p < 0.01$). The difference in Bi-LSTM-CRF and Bi-LSTM baseline, does not appear to be statistically significant ($p > 0.05$). However, the improvements over CRF baseline for all LSTM models are statistically significant.

7 Discussion

Overall, Approx-Skip-Chain CRF model achieved better F-scores than CRF,Bi-LSTM and Bi-LSTM-

CRF in both strict and relaxed evaluations. The results of strict evaluation, as shown in Figure 1, are our main focus of discussion due to their importance in the clinical domain. As expected, two exact inference-based CRF-LSTM models (Bi-LSTM-CRF and Bi-LSTM-CRF-pair) show the highest precision for all labels. Approx-Skip-Chain CRF's precision is lower(due to approximate inference) but it still mostly outperforms Bi-LSTM. The recall for Skip Chain CRF is almost equal or better than all other models due to its robustness in modeling dependencies between distant labels. The variations in recall contribute to the major differences in F-scores. These variations can be due to several factors including the rarity of that label in the dataset, the complexity of phrases of a particular label, etc.

We believe, exact CRF-LSTM models described here require more training samples than the baseline Bi-LSTM to achieve a comparable recall for labels that are complex or "difficult to detect". For example, as shown in table 1, we can divide the labels into frequent (*Other SSD*, *Indication*, *Severity*, *Drugname*, *Dosage*, and *Frequency*) and rare or sparse (*Duration*, *ADE*, *Route*). We can make a broad generalization, that exact CRF models (especially Bi-LSTM-CRF) have somewhat lower recall for rare labels. This is true for most labels except for *Route*, *Indication*, and *Severity*. The CRF models have very close recall (0.780,0.782) to the baseline Bi-LSTM (0.803) for *Route* even though its number of incidences are lower (2,862 incidences) than *Indication* (3,724 incidences) and *Severity* (3,628 incidences), both of which have lower recall even though their incidences are much higher.

Complexity of each label can explain the aforementioned phenomenon. *Route* for instance, frequently contains unique phrases such as "by mouth" or "p.o.," and is therefore easier to detect. In contrast, *Indication* is ambiguous. Its vocabulary is close to two other labels: *ADE* (1,807 incidences) and the most populous *Other SSD* (40,984 incidences). As a consequence, it is harder to separate the three labels. Models need to learn cues from surrounding context, which is more difficult and requires more samples. This is why the recall for *Indication* is lower for CRF-LSTM models, even though its number of incidences is higher than *Route*. To further support our explanation, our results show that the exact CRF-LSTM models mislabeled around 40% of *Indication* words as *Other SSD*, as opposed to just 20 % in case of the Bi-LSTM baseline. The label *Severity* is a similar case. It contains non-label-specific phrases such as "not terribly", "very rare" and "small area," which may explain why almost 35% of *Severity* words are mislabeled as *Outside* by the bi-LSTM-CRF as opposed to around 20% by the baseline.

It is worthwhile to note that among exact CRF-LSTM models, the recall for Bi-LSTM-CRF-pair is much better than Bi-LSTM-CRF even for sparse labels. This validates our initial hypothesis that Neural Net based pairwise modeling may lead to better detection of rare labels.

8 Conclusion

We have shown that modeling pairwise potentials and using an approximate version of Skip-chain inference increase the performance of the LSTM-CRF models. We also show that these models perform much better than baseline LSTM and CRF models. These results suggest that the structured prediction models are good directions for improving the exact phrase extraction for clinical entities.

Acknowledgments

We thank the UMassMed annotation team: Elaine Freund, Wiesong Liu, Steve Belknap, Nadya Frid, Alex Granillo, Heather Keating, and Victoria Wang for creating the gold standard evaluation set used in this work. We also thank the anonymous reviewers for their comments and suggestions.

This work was supported in part by the grant HL125089 from the National Institutes of Health (NIH). We also acknowledge the support from the United States Department of Veterans Affairs (VA) through Award 1I01HX001457. This work was also supported in part by the Center for Intelligent Information Retrieval. The contents of this paper do not represent the views of CIIR, NIH, VA or the United States Government.

References

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Harsha Gurulingappa, Roman Klinger, Martin Hofmann-Apitius, and Juliane Fluck. 2010. An empirical evaluation of resources for the identification of diseases and adverse effects in biomedical literature. In *2nd Workshop on Building and evaluating resources for biomedical text mining (7th edition of the Language Resources and Evaluation Conference)*.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

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

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Abhyuday Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of NAACL-HLT*, pages 473–482.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. 2015. Biomedical named entity recognition based on extended recurrent neural networks. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 649–652. IEEE.

Guosheng Lin, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2015. Deeply learning the messages in message passing inference. In *Advances in Neural Information Processing Systems*, pages 361–369.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

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

Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. 2016. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094.

Christian M Rochefort, Aman D Verma, Tewodros Eguale, Todd C Lee, and David L Buckeridge. 2015. A novel method of adverse event detection can accurately identify venous thromboembolisms (vtes) from narrative electronic health record data. *Journal of the American Medical Informatics Association*, 22(1):155–165.

Sunita Sarawagi and William W Cohen. 2004. Semimarkov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.

Learning to Represent Review with Tensor Decomposition for Spam Detection

Xuepeng Wang[1,2], **Kang Liu**[1], **Shizhu He**[1] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{xpwang, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn

Abstract

Review spam detection is a key task in opinion mining. To accomplish this type of detection, previous work has focused mainly on effectively representing fake and non-fake reviews with discriminative features, which are discovered or elaborately designed by experts or developers. This paper proposes a novel review spam detection method that learns the representation of reviews automatically instead of heavily relying on experts' knowledge in a data-driven manner. More specifically, according to 11 relations (generated automatically from two basic patterns) between reviewers and products, we employ tensor decomposition to learn the embeddings of the reviewers and products in a vector space. We collect relations between any two entities (reviewers and products), which results in much useful and global information. We concatenate the review text, the embeddings of the reviewer and the reviewed product as the representation of a review. Based on such representations, the classifier could identify the opinion spam more precisely. Experimental results on an open Yelp dataset show that our method could effectively enhance the spam detection accuracy compared with the state-of-the-art methods.

1 Introduction

With the development of E-commerce, more and more customers share their experiences about products and services by posting reviews on the web. These reviews could heavily guide the purchasing behaviors of customers. The products which receive more positive reviews tend to attract more consumers and result in more profits. Studies on Yelp.com have shown that an extra half-star rating could cause a restaurant to sell out 19% more products (Anderson and Magruder, 2012), and a one-star increase leads to a 5-9% profit increase (Luca, 2011). Therefore, more and more sellers and manufacturers have begun to place emphasis on analyzing reviews. However, the question remains: is every online review trustful? It has been reported that up to 25% of the reviews on Yelp.com could be fraudulent[1]. Due to the great profit or reputation, impostors or spammers energetically post fake reviews on the web to promote or defame targeted products (Jindal and Liu, 2008). Such fake reviews could mislead consumers and damage the online review websites' reputations. Therefore, it is necessary and urgent to detect fake reviews (review spam).

To accomplish this goal, much work has been conducted. They commonly regard this task as a classification task and most efforts are devoted to exploring useful features for representing target reviews. Li et al. (2013) and Kim et al. (2015) represent reviews with linguistic features; Lim et al. (2010) and Mukherjee et al. (2013c) represent reviews with reviewers' behavioral features[2]; Wang et al. (2011) and Akoglu et al. (2013) explore graph structure features[3]; Mukherjee et al. (2013b),

[1] http://www.bbc.com/news/technology-24299742

[2] Reviewers' spammer-like behaviors, e.g., if a reviewer continuously posts reviews within a short period of time, (s)he might be a spammer, and her (his) posted reviews could be spam.

[3] A kind of behavioral features which contain much interactions between reviewers and products

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 866–875,
Austin, Texas, November 1-5, 2016.

Rayana and Akoglu (2015) use the combination of aforementioned features. According to the existing studies, reviewers' behavioral features have been proven to be more effective than reviews' linguistic features for detecting review spam (Mukherjee et al., 2013c). It is because that foxy spammers could easily disguise their writing styles and forge reviews, discovering discriminative linguistic features is very difficult. Recently, most of the researchers (Rayana and Akoglu, 2015) have focused on the reviewers' behavioral features, the intuition behind which is to capture the reviewers' actions and supposes that those reviews written with spammer-like behaviors would be spam.

Although, the existing work has made significant progress in combating review spamming, they also have several limitations as follows. (1) The representations of reviews rely heavily on experts' prior knowledge or developers' ingenuity. To discover more discriminative features for representing reviews, previous work (Mukherjee et al., 2013b; Rayana and Akoglu, 2015) have spent lots of manpower and time on the statistics of the review datasets. Besides, experts' prior knowledge or developers' ingenuity is not always reliable with the variations of domains and languages. For example, based on the datasets from Dianping site[4], Li et al. (2015) find that the real users tend to review the restaurants nearby, but the spammers are not restricted to the geographical location, they may come from anywhere. However, it is not true in the Yelp datasets (Mukherjee et al., 2013b). We found that 72% of the Yelp's review spam is posted from the areas near the restaurants, but only 64% of the authentic reviews are near the restaurants. Therefore, how to learn the representations of reviews directly from data instead of heavily relying on the experts' prior knowledge or developers' ingenuity becomes crucial and urgent. (2) Furthermore, limited by the experts' knowledge, previous work only uses partial information of the review system. For example, traditional behavioral features (Lim et al., 2010; Mukherjee et al., 2013c) only utilize the information of individual reviewer. Although the work (Wang et al., 2011; Rayana and Akoglu, 2015) have tried to employ graph structure to consider the interactions among the reviewers and products, it is a kind of local interaction defined within the same product review page. However, the interaction among the reviewers and products from different review pages also provides much useful and global information, which is ignored by the previous work.

To tackle the problems described above, we propose a novel review spam detection method which can learn the representations of reviews instead of heavily relying on the experts' knowledge, developers' ingenuity, or spammer-like assumption, and can reserve the original information with a global manner. Inspired by the work about distributional representation or embedding for text and knowledge base, we propose a tensor factorization-based model to learn the representation of each review automatically. The finally learnt representation of each review is determined by the original data, rather than the features or clues found by experts. More specifically, we defined two basic patterns without any experts' knowledge, developers' ingenuity, or spammer-like assumptions. Based on the two basic patterns, we extended 11 interactive relations between entities (reviewers and products) in terms of time, locations, social contact, etc. Then, we build a 3-mode tensor on these 11 interactive relations between reviewers and products. In order to reserve the original information with a global manner, we collect the relations of any two entities regardless of whether they are from the same review page. In this way, we could reserve the original information of the data as much as possible, which dispenses with human selection. Next, we utilize tensor factorization to perform tensor decomposition, and the representations of reviewers and products are embedded in a latent vector space by collective learning. Afterward, we could obtain vector representations (embeddings) for both the reviewers and products. Then, we concatenate the review text (e.g., bigram), the embedding of a reviewer and the reviewed product as the representation of a review. In this way, the representations of reviews driven by data could be learnt in the entire review system in a global manner. Finally, such representations are fed into a classifier to detect the review spam.

In summary, this paper makes the following contributions:

- It addresses the spam detection issue with a

[4]http://www.dianping.com

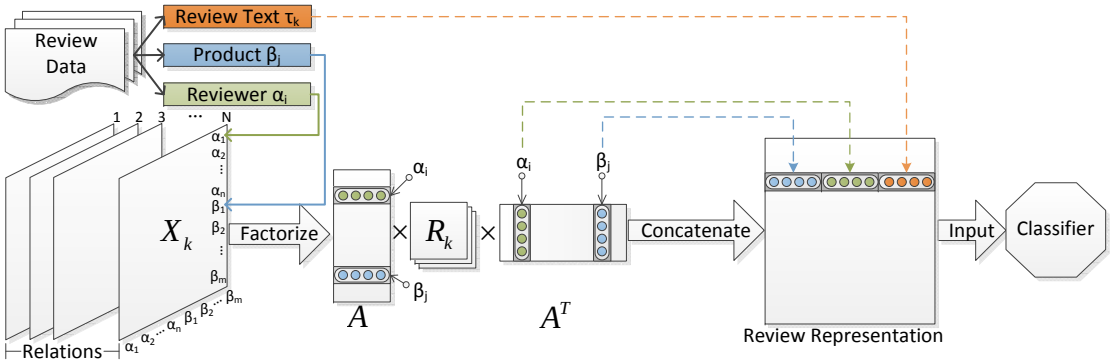


Figure 1: Illustrated of our method. The α_i denotes the i-th reviewer, and the β_j denotes the j-th product.

new perspective. Specifically, it learns the representation of reviews directly from the data. The key advantage is that it can represent the reviews instead of heavily relying on human ingenuity cost, experts' knowledge or any spammer-like assumption.

- It collects the relations between any two entities regardless of whether they are from the same review page, which results in much global information. With the help of tensor factorization, it could collectively embed the information of different relations into the final representations of reviews, and further optimize the representations. Therefore it could faithfully reflect the original characteristics of the entire review system with a global manner.
- An extra advantage is that the learnt representations of reviews are embeddings in a latent space. They are hardly comprehended by human beings included spammers. It's a robust detection method in contrast to the previous methods in which the reviews are represented by the explicit detecting clues and features. Once have realized the explicit features that were captured, experienced spammers could change their spamming strategies.
- The method of this paper renders 89.2% F1-score in detecting restaurant review spam which is higher than the F1-score of 86.1% rendered by the method in (Mukherjee et al., 2013b) (in hotel domain, it's 87.0% vs 84.8%). These experimental results give good confidence to the proposed approach, and the learnt representations of reviews are more robust and effective than in previous methods.

2 The Proposed Method

In this section, we propose our method (shown in Figure 1) in detail. Compared with the previous work, we address the review spam detection issue by learning the representation of the reviews automatically in a latent space without experts' knowledge. First, we extend 11 interactive relations between entities (reviewers and products) from the two basic patterns in terms of time, locations, social contact, etc. Then, our method generates 11 relation matrices of the reviewers (α_i) and products (β_j). After that, we construct a 3-mode tensor $\mathbf{X}$, where each slice X_k in $\mathbf{X}$ denotes the link relationship between the reviewers and products in the relation k. Second, we factorize the tensor $\mathbf{X}$ by employing the algorithm RESCAL (Nickel et al., 2011). In the factorization results, A represents the embeddings of the reviewers (α_i) and products (β_j) in the latent space with the collective learning. Third, we concatenate the review text (bigram), the embedding of its reviewer and the reviewed product together, as the representation of the review. Last, the concatenated embedding of the review is fed into a classifier (e.g., SVM) to detect whether it is a fake or non-fake review.

2.1 Relation Matrices Generation

In the review system, there are two kinds of entities: reviewers and products[5]. Each entity has several attributes, e.g., the attribute 'location' of a restaurant is Chicago (the restaurant is regarded as a product). More details are shown in Table 1.

To learn the representations of reviews directly from the data instead of experts' knowledge, we defined two basic patterns:

[5]The product refers to a hotel/restaurant in our experiments.

Reviewer Attribute	Product Attribute
set of reviewed products	set of reviewers
set of reviews (rating score, time)	set of reviews (rating score, time)
website joining date	average rating
friend count	review count
location	location

Table 1: Entities and Attributes

Pattern 1:Record the relationships between two entities.

Pattern 2:Record the relationships between attributes of two entities.

These patterns do not contain any spammer-like prior assumption, just record the natural relation in the original review system. Based on the two basic patterns, we extended 11 interactive relations between entities and their attributes (showed in Table 1). They will be described in detail as follows. Meanwhile, we define that $avg(a_{k,i}) = \frac{1}{n}\sum_{k=1}^{n} a_{k,i}$.

1. **Have reviewed:** This relation records whether a reviewer has reviewed a product. If reviewer α_i reviewed product β_j, the value $X[i,j,1]$ in this relation matrix $X[:,:,1]$ is 1, otherwise it's 0.

2. **Rating score:** What score (1 to 5 star) a reviewer-rated product receives. The value $X[i,j,2] \in \{1,2,...,5\}$.

3. **Commonly reviewed products:** The number of products that a reviewer commonly reviewed with other reviewers. The value $X[i,j,3] = |P_{ij}|, P_{ij} = P_i \cap P_j$; P_i is the product set reviewed by reviewer α_i.

4. **Commonly reviewed time difference:** The time differences that a reviewer who commonly reviews with other reviewers on the same products. The value $X[i,j,4] = \overline{t_i} - \overline{t_j}$, where $\overline{t_i} = avg(t_{k,i})$; $t_{k,i}$ is the time that the reviewer α_i reviewed the product β_k in the P_{ij} set.

5. **Commonly reviewed rating difference:** The rating differences that a reviewer who commonly reviews with other reviewers on the same products. The value $X[i,j,5] = \overline{r_i} - \overline{r_j}$, where $\overline{r_i} = avg(r_{k,i})$; $r_{k,i}$ is the score of the reviewer α_i rated the product β_k in P_{ij} set.

6. **Date difference of websites joined:** The date differences of joining review websites between a reviewer and others. The value $X[i,j,6] = d_i - d_j$, where d_i is the date on which reviewer α_i joining websites.

7. **Average rating difference:** The differences in the average rating of a reviewer over all his reviews compared with other reviewers. The value $X[i,j,7] = \overline{\gamma_i^r} - \overline{\gamma_j^r}$, where $\overline{\gamma_i^r} = avg(\gamma_{k,i}^r)$; $\gamma_{k,i}^r$ is the score with which the reviewer α_i rated the product β_k in P_i.

   The differences in the average rating of a product over all its reviews compared with other products. $X[i,j,7] = \overline{\gamma_i^p} - \overline{\gamma_j^p}$, where $\overline{\gamma_i^p} = avg(\gamma_{k,i}^p)$; $\gamma_{k,i}^p$ is the score of review k in R_i^β, which is the review set for product β_i.

8. **Friend count difference:** The differences in the friend count of a reviewer compared to others. At the review website, a reviewer can make friends with others. The value $X[i,j,8] = f_i - f_j$; where f_i is the friend count of reviewer α_i.

9. **Have the same location or not:** Whether two reviewers/products are from the same city or whether a reviewer has the same location with a product. If two entities have the same location, the value $X[i,j,9] = 1$, otherwise $X[i,j,9] = 0$.

10. **Common reviewers:** The number of the same reviewers that a product has with other products. The value $X[i,j,10] = |\Theta_{ij}|$, where $\Theta_{ij} = \Theta_i \cap \Theta_j$; Θ_i is the set of reviewers who reviewed product β_i.

11. **Review count difference:** The differences in the reviews count of any two reviewers. The value $X[i,j,11] = |R_i^\alpha| - \left|R_j^\alpha\right|$, where R_i^α is the reviews set of reviewer α_i. Or the differences in the reviews count of any two products, where $X[i,j,11] = \left|R_i^\beta\right| - \left|R_j^\beta\right|$, where R_i^β is the reviews set of product β_i.

According to the relations that we present above, we build 11 relation matrices among the reviewers and products. To unify the values of different matrices to a reference system, we normalize with the

sigmoid function. Thus, the value '0' will be normalized to '0.5'. Moreover, we set the values that make no sense to '0', such as the value between two products in Relation 1: Have reviewed. Then, we unite the 11 matrices to form the adjacent tensor. Each of the matrices is a slice of the tensor. The reviewers and products are regarded as the same entities in the tensor. We build two separate tensors for the hotel domain and restaurant domain respectively. Next, we perform tensor factorization to learn the representations (embeddings) of reviewers and products. Note that the word "relation" is normally used for binary (0/1) relations, but some values of aforementioned relations could be between 0 and 1. However, our experiments show that this type of relation is actually practicable. Besides, there is not any spammer-like assumption in the relations. Namely, the values of relations don't indicate how suspicious the reviewers are. The values faithfully reflect the original characteristics of the entire review system. This can help to reduce the need of carefully designing expert features and the understanding of domains as much as possible.

2.2 Learning to Represent Reviews

In general case, a review contains the text, the reviewer and the reviewed product. We firstly learn to represent reviewers and products. As mentioned above, based on the relations, we could construct an adjacency tensor $\mathbf{X}$. Then, we convert the global relation information related reviewers and products into embeddings through tensor factorization, where an efficient factorization algorithm called RESCAL (Nickel et al., 2011) is employed. First, we introduce it briefly.

To identify latent components in a tensor for collective learning, Nickel et al. (2011) proposed RESCAL, which is a tensor factorization algorithm. Given a tensor $X_{n' \times n' \times m'}$, RESCAL aims to have a rank-r approximation, where each slice X_k is factorized as

$$X_k \approx \boldsymbol{A} R_k \boldsymbol{A}^T, \text{ for all } k = 1...m', \qquad (1)$$

$\mathbf{A}$ is an $n' \times r$ matrix, where the i-th row denotes the i-th entity. R_k is an asymmetric $r \times r$ matrix that describes the interactions of the latent components according to the k-th relation. Note that while R_k differs in each slice, $\mathbf{A}$ remains the same.

$\mathbf{A}$ and R_k are derived by minimizing the loss function below.

$$\min_{\boldsymbol{A}, R_k} f(\boldsymbol{A}, R_k) + \lambda \cdot g(\boldsymbol{A}, R_k), \qquad (2)$$

where $f(\boldsymbol{A}, R_k) = \frac{1}{2}(\sum_k \parallel X_k - \boldsymbol{A} R_k \boldsymbol{A}^T \parallel_F^2)$ is the mean-squared reconstruction error, and $g(\boldsymbol{A}, R_k) = \frac{1}{2}(\parallel \boldsymbol{A} \parallel_F^2 + \sum_k \parallel R_k \parallel_F^2)$ is the regularization term.

In our method, slice X_k is the k-th relation above. The i-th entity is the i-th reviewer or product.

As mentioned in Section 2.1, in order to obtain more useful and global information automatically, we collect the relations of any two entities no matter whether they are from the same review page. Then we could embed the informations over multi-relations into the finally learnt representation by the tensor factorization. As Nickel et al. (2011) proved, all the relations have a determining influence on the learnt latent-component representation of the i-th entity. It removes the noise of the original data by learning through the global loss function. Consequently, we get the representation of reviewers and products with a further optimization by the collective learning.

2.3 Detecting Review Spam in Latent Space

After learning the representations of reviewers and products, we begin to represent the reviews that were written by reviewers for the products. Our final purpose is to detect the review spam. We concatenate the review text (bigram), the embedding of a reviewer and the reviewed product as the representation of a review. The representations of the review text by bigram have been proved to be effective in several previous work (Mukherjee et al., 2013b; Rayana and Akoglu, 2015; Kim et al., 2015). It's also a kind of data-driven representation. Then, we take the embeddings of the reviews as the input to the classifiers. Here, we use the linear kernel SVM model to compare with the experimental results in (Mukherjee et al., 2013b) and (Rayana and Akoglu, 2015).

3 Experiments

3.1 Datasets and Evaluation Metrics

Datasets: To evaluate the proposed method, we conducted experiments on Yelp dataset that was used in

previous studies (Mukherjee et al., 2013b; Mukherjee et al., 2013c; Rayana and Akoglu, 2015). Although there are other datasets for evaluation, such as (Jindal and Liu, 2008), (Lim et al., 2010; Xie et al., 2012) and (Ott et al., 2011), they are generated by human labeling or crowd sourcing and have been proved not to be reliable since human labeling fake reviews is quite poor (Ott et al., 2011). There was lack of real-life and nearly ground truth data, until Mukherjee et al. (2013c) proposed the Yelp review dataset. The statistics of the Yelp dataset are listed in Table 2. The reviewed product here refers to a hotel or restaurant.

Evaluation Metrics: We select precision (P), recall (R), F1-Score (F1) and accuracy (A) as metrics.

Domain	Hotel	Restaurant
fake	802	8368
non-fake	4876	50149
%fake	14.1%	14.3%
#reviews	5678	58517
#reviewers	5124	35593

Table 2: Yelp Labeled Dataset Statistics.

3.2 Our Method vs. The State-of-the-art Methods

To illustrate the effectiveness of the proposed approach, we select several state-of-the-arts for comparison. The first one is SPEAGLE$^+$ (Rayana and Akoglu, 2015), which is a kind of graph-based method. The representations of reviews in (Rayana and Akoglu, 2015) are combined with linguistic features, behavioral features and review graph structure features. It's a semi-supervised method. For a fair comparison with our 5-fold CV classification, we set the ratio of labeled data in SPEAGLE$^+$ to 80%. The second one is Mukherjee et al. (2013b). KC and Mukherjee (2016) also conduct experiments on the restaurant subset in Table 2. But they mainly focus on analyzing the effects of temporal dynamics. It's not our focus. So we didn't take it into comparison. In our experiments, we employ behavioral features (Mukherjee_BF) and both of behavioral and linguistic features (Mukherjee_BF+Bigram) proposed in Mukherjee et al. (2013b), respectively. The parameters used in these compared methods are same as the original papers. For our approach, we set the parameter r to 150, λ to 10, and the iteration number to 100.

The compared results are shown in Table 3. We utilize our learnt embeddings of reviewers (Ours_RE), both of reviewers' embeddings and products' embeddings (Ours_RE+PE), respectively. Moreover, to perform fair comparison, like Mukherjee et al. (2013b), we add representations of the review text in classifier (Ours_RE+PE+Bigram). From the results, we can observe that our method could outperform all state-of-the-arts in both the hotel and restaurant domains. It proves that our method is effective. Furthermore, the improvements in both the hotel and restaurant domains prove that our model possesses preferable domain-adaptability. It could represent the reviews more accurately and globally by learning from the original data, rather than the experts' knowledge or assumption.

3.3 The Effectiveness of Learning to Represent Review

To further prove the representations learnt by our method are effective for detecting review spam, we compare the learnt representation (embeddings) of reviewers (Ours_RE) (Table 3 (a,b) rows 7, 8) with existing behavioral features of reviewers (Mukherjee_BF) (Mukherjee et al., 2013b) (Table 3 (a,b) rows 3, 4). In results, using the learnt reviewers' representations in our method, results in around 2.0% (in 50:50) and 4.0% (in N.D.) improvement in F1 and A in the hotel domain, and results in around 2.1% (in 50:50) and 7.0%(in N.D.) improvement in F1 and A in the restaurant domain. These results show that our data-driven representations of reviewers are more helpful for review spam detection than existing reviewers' behavioral features, and that new method embeds more useful and accurate information from the original data. It isn't limited to experts' knowledge. Moreover, the latent representations are more robust because they are hardly perceived by spammers. Having realized the explicit existing behavioral features, crafty spammers tend to change their spamming strategies. Consider the feature "Review Length", which is used in (Mukherjee et al., 2013b), as an example. They find that the average review length of the spammers is quite short compared with non-spammers. However, once a crafty spammer realizes that he left this type of footprint, he could produce a review that is as long as the non-

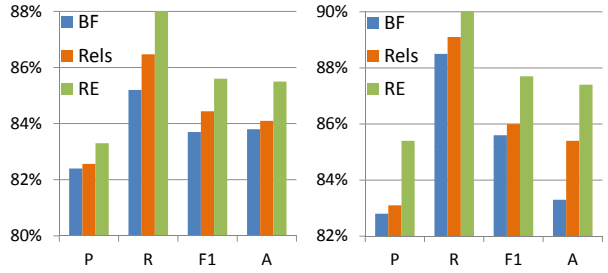
Method	C.D.	P	R	F1	A
SPEAGLE$^+$(80%)	50:50	75.7	83.0	79.1	81.0
	N.D.	26.5	56.0	36.0	80.4
Mukherjee_BF	50:50	82.4	85.2	83.7	83.8
	N.D.	41.4	84.6	55.6	82.4
Mukherjee_BF+Bigram	50:50	82.8	86.9	84.8	85.1
	N.D.	46.5	82.5	59.4	84.9
Ours_RE	50:50	83.3	88.1	**85.6**	**85.5**
	N.D.	47.1	83.5	**60.2**	**85.0**
Ours_RE+PE	50:50	83.6	89.0	**86.2**	**85.7**
	N.D.	47.5	84.1	**60.7**	**85.3**
Ours_RE+PE+Bigram	50:50	84.2	89.9	**87.0**	**86.5**
	N.D.	48.2	85.0	**61.5**	**85.9**

(a) Hotel

P	R	F1	A	
80.5	83.2	81.8	82.5	1
50.1	70.5	58.6	82.0	2
82.8	88.5	85.6	83.3	3
48.2	87.9	62.3	78.6	4
84.5	87.8	86.1	86.5	5
48.9	87.3	62.7	82.3	6
85.4	90.2	**87.7**	**87.4**	7
56.9	90.1	**69.8**	**85.8**	8
86.0	90.7	**88.3**	**88.0**	9
57.4	89.9	**70.1**	**86.1**	10
86.8	91.8	**89.2**	**89.9**	11
58.2	90.3	**70.8**	**87.8**	12

(b) Restaurant

Table 3: Classification results across the behavioral features (BF), the reviewer embeddings (RE) , product embeddings (PE) and bigram of the review texts. Training uses balanced data (50:50). Testing uses two class distributions (C.D.): 50:50 (balanced) and Natural Distribution (N.D.). Improvements of our method are statistically significant with p<0.005 based on paired *t*-test.



(a) Hotel **(b)** Restaurant

Figure 2: SVM 5-fold CV classification results across behavioral features (BF), 11 relations (Rels) and reviewer embeddings (RE) in our method. Both training and testing use balanced data (50:50). Improvements are statistically significant with p<0.005 based on paired *t*-test.

Dropped Relation	Hotel		Restaurant	
	F1	A	F1	A
1	-2.1	-2.0	-2.0	-3.1
2	-2.3	-2.1	-1.9	-2.9
3	-3.9	-4.0	-4.0	-6.3
4	-3.7	-3.5	-3.6	-5.5
5	-3.5	-3.6	-2.8	-4.5
6	-2.5	-2.5	-3.4	-5.2
7	-3.2	-3.2	-3.3	-5.0
8	-2.8	-2.6	-3.0	-4.6
9	-4.0	-3.7	-3.7	-5.4
10	-2.2	-2.4	-1.8	-2.8
11	-2.6	-2.4	-2.7	-4.4

Table 4: SVM 5-fold CV classification results by dropping relations from our method utilizing RE+PE+Bigram. Both training and testing use balanced data (50:50). Differences in classification metrics for each dropped relation are statistically significant with p<0.01 based on paired *t*-test.

spammers to pretend to be a normal reviewer. Besides, as there isn't any spammer-like assumption in our extended relations (Section 2.1), crafty spammers have little influence on them.

We also compared existing behavioral features (BF) (Mukherjee et al., 2013b) with detecting review spam by only employing the 11 generated relations (Rels). We take the relation matrix row of each reviewer as the representations of the reviews. According to the results shown in Figure 2, the 11 generated relations (Rels) results in an obvious improvement than the existing behavioral features (BF) (Mukherjee et al., 2013b) (Table 3 (a,b) row 3) in both the hotel and restaurant domains. It proves that the generated relations could obtain more useful and global informations, as they collect the relations of any entities (reviewers and products) regardless of whether they are from the same review page. Furthermore, Figure 2 also showed that the embeddings of reviewers (RE) learnt by the tensor decomposition perform better than the Rels. As we mentioned in Section 2.2, the tensor decomposition embeds the informations over all the relations collectively, and removes the noise of the original data by learning through the global loss function. Consequently, we get the representations with a further optimization.

3.4 The Effectiveness of Product Embeddings

In general case, a review contains the review text, the reviewer and the reviewed product. But most of the previous work represent the reviews with the reviewers' behavioral features and the reviews' linguistic features. The products are seldom represented. As shown in Table 3 (a,b) rows 9,10 , the representations which added the products embeddings

perform better than just using the reviewer embeddings. Statistics of the datasets suggest that there are about 1% of spammers who not only write fake reviews, but also write non-fake reviews. Liu (2015) also proved that some reviewers have contributed many genuine reviews and have built up their reputation; then they started to spam for some businesses, or even sell their accounts to spammers. Compared with previous work, our method by adding product embeddings could distinguish the reviews of the same reviewer for different products.

3.5 The Effects of Different Relations

We also drop relations of our method with a graceful degradation. Table 4 shows the performances of our method utilizing BF+PE+Bigram for hotel and restaurant domains. We found that dropping Relations 1, 2 and 10 results in a relatively gentle reduction (about 2.2%) in F1-score. According to our survey, the sparseness of the slices generated by Relation 1, 2 and 10 is about 99.9%. For this reason, the result is a relatively gentle reduction. Dropping other relations also result in a 2.5-4.0% performance reduction. It proves that each relation has an influence on the learning to represent reviews.

4 Related Work

Jindal and Liu (2008) first propose the problem of review spam detection. They identify three categories of spam: fake reviews (also called untruthful opinions), reviews on the brand only, and non-reviews.Stepping studies focus on studying fake reviews because of its difficulty to be detected. Most efforts are devoted to represent fake and non-fake reviews with effective features.

Linguistic Features Ott et al. (2011) apply psychological and linguistic clues to identify review spam. They produce the first dataset of gold-standard deceptive review spam, employing crowdsourcing through the Amazon Mechanical Turk. Harris (2012) explores several human- and machine-based assessment methods with writing style features. Feng et al. (2012a) investigate syntactic stylometry for review spam detection. Li et al. (2013) propose a generative LDA-based topic modeling approach for fake review detection. They (Li et al., 2014b) further investigate the general difference of language usage between deceptive and truthful reviews. Li et al. (2014a) propose a positive-unlabeled learning method base on unigrams and bigrams. Kim et al. (2015) carry out a frame-based deep semantic analysis on deceptive opinions.

Behavioral Features Lim et al. (2010) investigate reviewers' rating behavioral features. Jindal et al. (2010) identify unusual review patterns which can represent suspicious behaviors of reviews. Li et al. (2011) provide a two-view semi-supervised method, co-training method base on behavioral features. Feng et al. (2012b) study the distributions of behavioral features. Xie et al. (2012) explore the singleton reviews with abnormal temporal patterns. Mukherjee et al. (2012) study the group spammers' behavioral features. Mukherjee et al. (2013a) propose a principal method which models the spamicity of reviewers. Fei et al. (2013) model the reviewers' co-occurrence in review bursts. Mukherjee et al. (2013c) prove that reviewers' behavioral features are more effective than reviews' linguistic features for detecting review spam. Li et al. (2015) explore the temporal and spatial patterns at Dianping.com. KC and Mukherjee (2016) analyze the temporal dynamics of opinion spamming.

Graph Structure Wang et al. (2011) investigate the review graph features of online store review. Akoglu et al. (2013) exploit the network effect among reviewers and products.

Combined Features There are also some work which explores methods via the combined features referred above. Mukherjee et al. (2013b) prose a method base on the linguistic features and behavioral features. Rayana and Akoglu (2015) propose a model that utilizes clues from review text, reviewers' behaviors and the review graph structure.

5 Conclusion and Future Work

This paper proposes a new review spam detection method that learns the representations of reviews instead of heavily relying on experts' knowledge in a data-driven manner. A 3-mode tensor is built on the relations which are generated from two patterns, and a tensor factorization algorithm is used to automatically learn the vector representations of reviewers and products. Afterwards, we concatenate the review text, the embedding of a reviewer and the

reviewed product as the representation of a review. Then, a classifier is applied to detect the review spam. Experimental results prove the effectiveness of the proposed method, which learns more robust review representations. In future work, we plan to explore a more effective way to learn the embeddings of review text.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018), the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61502493). We would like to thank Prof. Bing Liu for sharing the Yelp review dataset with us, and the anonymous reviewers for their detailed comments and suggestions.

References

Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11.

Michael Anderson and Jeremy Magruder. 2012. Learning from the crowd: Regression discontinuity estimates of the effects of an online review database*. *The Economic Journal*, 122(563):957–989.

Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting burstiness in reviews for review spammer detection. In *ICWSM*. Citeseer.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012a. Syntactic stylometry for deception detection. In *Proceedings of the 50th ACL: Short Papers-Volume 2*, pages 171–175. ACL.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012b. Distributional footprints of deceptive product reviews. In *ICWSM*.

C Harris. 2012. Detecting deceptive opinion spam using human computation. In *Workshops at AAAI on Artificial Intelligence*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the First WSDM*, pages 219–230. ACM.

Nitin Jindal, Bing Liu, and Ee-Peng Lim. 2010. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th CIKM*, pages 1549–1552. ACM.

Santosh KC and Arjun Mukherjee. 2016. On the temporal dynamics of opinion spamming: Case studies on yelp. In *Proceedings of the 25th International Conference on World Wide Web*, pages 369–379. International World Wide Web Conferences Steering Committee.

Seongsoon Kim, Hyeokyoon Chang, Seongwoon Lee, Minhwan Yu, and Jaewoo Kang. 2015. Deep semantic frame-based deceptive opinion spam analysis. In *Proceedings of the 24th CIKM*, pages 1131–1140. ACM.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI Proceedings*, volume 22, page 2488.

Jiwei Li, Claire Cardie, and Sujian Li. 2013. Topicspam: a topic-model based approach for spam detection. In *ACL (2)*, pages 217–221.

Huayi Li, Bing Liu, Arjun Mukherjee, and Jidong Shao. 2014a. Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas*, 18(3):467–475.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014b. Towards a general rule for identifying deceptive opinion spam. ACL.

Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. 2015. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Ninth International AAAI Conference on Web and Social Media*.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th CIKM*, pages 939–948. ACM.

Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Michael Luca. 2011. Reviews, reputation, and revenue: The case of yelp. com. *Com (September 16, 2011). Harvard Business School NOM Unit Working Paper*, (12-016).

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st WWW*, pages 191–200. ACM.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD*, pages 632–640. ACM.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013b. Fake review detection: Classification and analysis of real and pseudo reviews. Technical report, Technical Report UIC-CS-2013-03, University of Illinois at Chicago.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S Glance. 2013c. What yelp fake review filter might be doing? In *ICWSM*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective

learning on multi-relational data. In *Proceedings of the 28th ICML*, pages 809–816.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th ACL: Human Language Technologies-Volume 1*, pages 309–319. ACL.

Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 985–994. ACM.

Guan Wang, Sihong Xie, Bing Liu, and Philip S Yu. 2011. Review graph based online store review spammer detection. In *Proceedings of the 11th ICDM*, pages 1242–1247. IEEE.

Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. 2012. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th KDD*, pages 823–831. ACM.

Stance Detection with Bidirectional Conditional Encoding

Isabelle Augenstein and **Tim Rocktäschel**
Department of Computer Science
University College London
i.augenstein@ucl.ac.uk, t.rocktaschel@cs.ucl.ac.uk

Andreas Vlachos and **Kalina Bontcheva**
Department of Computer Science
University of Sheffield
{a.vlachos, k.bontcheva}@sheffield.ac.uk

Abstract

Stance detection is the task of classifying the attitude Previous work has assumed that either the target is mentioned in the text or that training data for every target is given. This paper considers the more challenging version of this task, where targets are not always mentioned and no training data is available for the test targets. We experiment with conditional LSTM encoding, which builds a representation of the tweet that is dependent on the target, and demonstrate that it outperforms encoding the tweet and the target independently. Performance is improved further when the conditional model is augmented with bidirectional encoding. We evaluate our approach on the SemEval 2016 Task 6 Twitter Stance Detection corpus achieving performance second best only to a system trained on semi-automatically labelled tweets for the test target. When such weak supervision is added, our approach achieves state–of-the-art results.

1 Introduction

The goal of stance detection is to classify the attitude expressed in a text towards a given target, as "positive", "negative", or "neutral". Such information can be useful for a variety of tasks, e.g. Mendoza et al. (2010) showed that tweets stating actual facts were affirmed by 90% of the tweets related to them, while tweets conveying false information were predominantly questioned or denied. In this paper we focus on a novel stance detection task, namely tweet stance detection towards previously unseen targets (mostly entities such as politicians or issues of public interest), as defined in the SemEval Stance Detection for Twitter task (Mohammad et al., 2016). This task is rather difficult, firstly due to not having training data for the targets in the test set, and secondly, due to the targets not always being mentioned in the tweet. For example, the tweet "@realDonaldTrump is the only honest voice of the @GOP" expresses a positive stance towards the target *Donald Trump*. However, when stance is annotated with respect to *Hillary Clinton* as the implicit target, this tweet expresses a negative stance, since supporting candidates from one party implies negative stance towards candidates from other parties.

Thus the challenge is twofold. First, we need to learn a model that interprets the tweet stance towards a target that might not be mentioned in the tweet itself. Second, we need to learn such a model without labelled training data for the target with respect to which we are predicting the stance. In the example above, we need to learn a model for *Hillary Clinton* by only using training data for other targets. While this renders the task more challenging, it is a more realistic scenario, as it is unlikely that labelled training data for each target of interest will be available.

To address these challenges we develop a neural network architecture based on conditional encoding (Rocktäschel et al., 2016). A long-short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is used to encode the target, followed by a second LSTM that encodes the tweet using the encoding of the target as its initial state. We show that this approach achieves better F1 than an SVM baseline, or an independent LSTM encoding of the tweet and the target. Results improve fur-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 876–885,
Austin, Texas, November 1-5, 2016.

ther (0.4901 F1) with a bidirectional version of our model, which takes into account the context on either side of the word being encoded. In the context of the shared task, this would have been the second best result, except for an approach which uses automatically labelled tweets for the test targets (F1 of 0.5628). Lastly, when our bidirectional conditional encoding model is trained on such data, it achieves state-of-the-art performance (0.5803 F1).

2 Task Setup

The SemEval 2016 Stance Detection for Twitter shared task (Mohammad et al., 2016) consists of two subtasks, Task A and Task B. In Task A the goal is to detect the stance of tweets towards targets given labelled training data for all test targets (*Climate Change is a Real Concern*, *Feminist Movement*, *Atheism*, *Legalization of Abortion* and *Hillary Clinton*). In Task B, which is the focus of this paper, the goal is to detect stance with respect to an unseen target, *Donald Trump*, for which labeled training/development data is not provided.

Systems need to classify the stance of each tweet as "positive" (FAVOR), "negative" (AGAINST) or "neutral" (NONE) towards the target. The official metric reported for the shared task is F1 macro-averaged over the classes FAVOR and AGAINST. Although the F1 of NONE is not considered, systems still need to predict it to avoid precision errors for the other two classes.

Even though participants were not allowed to manually label data for the test target *Donald Trump*, they were allowed to label data automatically. The two best-performing systems submitted to Task B, pkudblab (Wei et al., 2016) and LitisMind (Zarrella and Marsh, 2016) made use of this, thus changing the task to weakly supervised seen target stance detection, instead of an unseen target task. Although the goal of this paper is to present stance detection methods for targets for which no training data is available, we show that they can also be used successfully in a weakly supervised framework and outperform the state-of-the-art on the SemEval 2016 Stance Detection for Twitter dataset.

3 Methods

A common stance detection approach is to treat it as a sentence-level classification task similar to sentiment analysis (Pang and Lee, 2008; Socher et al., 2013). However, such an approach cannot capture the stance of a tweet with respect to a particular target, unless training data is available for each of the test targets. In such cases, we could learn that a tweet mentioning *Donald Trump* in a positive manner expresses a negative stance towards *Hillary Clinton*. Despite this limitation, we use two such baselines, one implemented with a Support Vector Machine (SVM) classifier and one with an LSTM network, in order to assess whether we are successful in incorporating the target in stance prediction.

A naive approach to incorporate the target in stance prediction would be to generate features concatenating the target with words from the tweet. Ignoring the issue that such features would be rather sparse, a classifier could learn that some words have target-dependent stance weights, but it still assumes that training data is available for each target.

In order to learn how to combine the stance target with the tweet in a way that generalises to unseen targets, we focus on learning distributed representations and ways to combine them. The following sections develop progressively the proposed bidirectional conditional LSTM encoding model, starting from independently encoding the tweet and the target using LSTMs.

3.1 Independent Encoding

Our initial attempt to learn distributed representations for the tweets and the targets is to encode the target and tweet independently as k-dimensional dense vectors using two LSTMs (Hochreiter and Schmidhuber, 1997).

$$
\begin{aligned}
\mathbf{H} &= \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} \\
\mathbf{i}_t &= \sigma(\mathbf{W}^i \mathbf{H} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f \mathbf{H} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o \mathbf{H} + \mathbf{b}^o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{H} + \mathbf{b}^c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

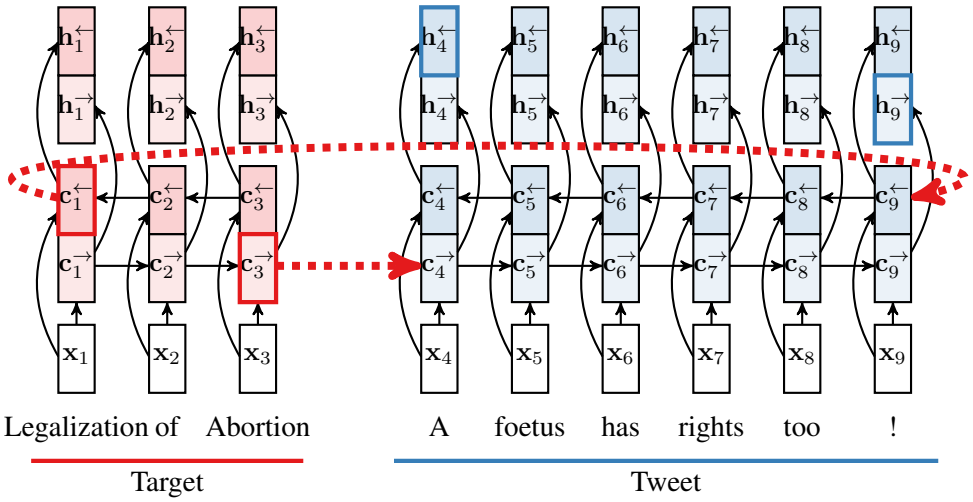


Figure 1: Bidirectional encoding of tweet conditioned on bidirectional encoding of target ($[\mathbf{c}_3^{\rightarrow}\ \mathbf{c}_1^{\leftarrow}]$). The stance is predicted using the last forward and reversed output representations ($[\mathbf{h}_9^{\rightarrow}\ \mathbf{h}_4^{\leftarrow}]$).

Here, $\mathbf{x}_t$ is an input vector at time step t, $\mathbf{c}_t$ denotes the LSTM memory, $\mathbf{h}_t \in \mathbb{R}^k$ is an output vector and the remaining weight matrices and biases are trainable parameters. We concatenate the two output vector representations and classify the stance using the softmax over a non-linear projection

$$\text{softmax}(\tanh(\mathbf{W}^{\text{ta}}\mathbf{h}_{\text{target}} + \mathbf{W}^{\text{tw}}\mathbf{h}_{\text{tweet}} + \mathbf{b}))$$

into the space of the three classes for stance detection where $\mathbf{W}^{\text{ta}}, \mathbf{W}^{\text{tw}} \in \mathbb{R}^{3\times k}$ are trainable weight matrices and $\mathbf{b} \in \mathbb{R}^3$ is a trainable class bias. This model learns target-independent distributed representations for the tweets and relies on the non-linear projection layer to incorporate the target in the stance prediction.

3.2 Conditional Encoding

In order to learn target-dependent tweet representations, we use conditional encoding as previously applied to the task of recognising textual entailment (Rocktäschel et al., 2016). We use one LSTM to encode the target as a fixed-length vector. Then, we encode the tweet with another LSTM, whose state is initialised with the representation of the target. Finally, we use the last output vector of the tweet LSTM to predict the stance of the target-tweet pair.

Formally, let $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ be a sequence of target word vectors, $(\mathbf{x}_{T+1}, \ldots, \mathbf{x}_N)$ be a sequence of tweet word vectors and $[\mathbf{h}_0\ \mathbf{c}_0]$ be a start state of zeros. The two LSTMs map input vectors and a previous state to a next state as follows:

$$\begin{aligned}
[\mathbf{h}_1\ \mathbf{c}_1] &= \text{LSTM}^{\text{target}}(\mathbf{x}_1, \mathbf{h}_0, \mathbf{c}_0) \\
&\ldots \\
[\mathbf{h}_T\ \mathbf{c}_T] &= \text{LSTM}^{\text{target}}(\mathbf{x}_T, \mathbf{h}_{T-1}, \mathbf{c}_{T-1}) \\
[\mathbf{h}_{T+1}\ \mathbf{c}_{T+1}] &= \text{LSTM}^{\text{tweet}}(\mathbf{x}_{T+1}, \mathbf{h}_0, \mathbf{c}_T) \\
&\ldots \\
[\mathbf{h}_N\ \mathbf{c}_N] &= \text{LSTM}^{\text{tweet}}(\mathbf{x}_N, \mathbf{h}_{N-1}, \mathbf{c}_{N-1})
\end{aligned}$$

Finally, the stance of the tweet w.r.t. the target is classified using a non-linear projection

$$\mathbf{c} = \tanh(\mathbf{W}\mathbf{h}_N)$$

where $\mathbf{W} \in \mathbb{R}^{3\times k}$ is a trainable weight matrix. This effectively allows the second LSTM to read the tweet in a target-specific manner, which is crucial since the stance of the tweet depends on the target (recall the Donald Trump example above).

3.3 Bidirectional Conditional Encoding

Bidirectional LSTMs (Graves and Schmidhuber, 2005) have been shown to learn improved representations of sequences by encoding a sequence from left to right and from right to left. Therefore, we adapt the conditional encoding model from Section 3.2 to use bidirectional LSTMs, which represent the target and the tweet using two vectors for each of them, one obtained by reading the target

and then the tweet left-to-right (as in the conditional LSTM encoding) and one obtained by reading them right-to-left. To achieve this, we initialise the state of the bidirectional LSTM that reads the tweet by the last state of the forward and reversed encoding of the target (see Figure 1). The bidirectional encoding allows the model to construct target-dependent representations of the tweet such that when a word is considered, both its left- and the right-hand side context are taken into account.

3.4 Unsupervised Pretraining

In order to counter-balance the relatively small amount of training data available (5,628 instances in total), we employ unsupervised pre-training by initialising the word embeddings used in the LSTMs with an appropriately trained word2vec model (Mikolov et al., 2013). Note that these embeddings are used only for initialisation, as we allow them to be optimised further during training.

In more detail, we train a word2vec model on a corpus of 395,212 unlabelled tweets, collected with the Twitter Keyword Search API[1] between November 2015 and January 2016, plus all the tweets contained in the official SemEval 2016 Stance Detection datasets (Mohammad et al., 2016). The unlabelled tweets are collected so that they contain the targets considered in the shared task, using up to two keywords per target, namely "hillary", "clinton", "trump", "climate", "femini", "aborti". Note that Twitter does not allow for regular expression search, so this is a free text search disregarding possible word boundaries. We combine this large unlabelled corpus with the official training data and train a skip-gram word2vec model (dimensionality 100, 5 min words, context window of 5).

Tweets and targets are tokenised with the Twitter-adapted tokeniser twokenize[2]. Subsequently, all tokens are lowercased, URLs are removed, and stopword tokens are filtered (i.e. punctuation characters, Twitter-specific stopwords ("rt", "#semst", "via").

As it will be shown in our experiments, unsupervised pre-training is quite helpful, since it is difficult to learn representations for all the words using only the relatively small training datasets available.

[1] https://dev.twitter.com/rest/public/search

[2] https://github.com/leondz/twokenize

Corpus	Favor	Against	None	All
TaskA_Tr+Dv	1462	2684	1482	5628
TaskA_Tr+Dv_HC	224	722	332	1278
TaskB_Unlab	-	-	-	278,013
TaskB_Auto-lab*	4681	5095	4026	13,802
TaskB_Test	148	299	260	707
Crawled_Unlab*	-	-	-	395,212

Table 1: Data sizes of available corpora. TaskA_Tr+Dv_HC is the part of TaskA_Tr+Dv with tweets for the target Hillary Clinton only, which we use for development. TaskB_Auto-lab is an automatically labelled version of TaskB_Unlab. Crawled_Unlab is an unlabelled tweet corpus collected by us.

Finally, to ensure that the proposed neural network architectures contribute to the performance, we also use the word vectors from word2vec to develop a Bag-of-Word-Vectors baseline (BOWV), in which the tweet and target representations are fed into a logistic regression classifier with L2 regularization (Pedregosa et al., 2011).

4 Experiments

Experiments are performed on the SemEval 2016 Task 6 corpus for Stance Detection on Twitter (Mohammad et al., 2016). We report experiments for two different experimental setups: one is the *unseen target* setup (Section 5), which is the main focus of this paper, i.e. detecting the stance of tweets towards previously unseen targets. We show that conditional encoding, by reading the tweets in a target-specific way, generalises to unseen targets better than baselines which ignore the target. Next, we compare our approach to previous work in a *weakly supervised framework* (Section 6) and show that our approach outperforms the state-of-the-art on the SemEval 2016 Stance Detection Subtask B corpus.

Table 1 lists the various corpora used in the experiments and their sizes. TaskA_Tr+Dv is the official SemEval 2016 Twitter Stance Detection TaskA training and development corpus, which contain instances for the targets *Legalization of Abortion*, *Atheism*, *Feminist Movement*, *Climate Change is a Real Concern* and *Hillary Clinton*. TaskA_Tr+Dv_HC is the part of the corpus which contains only the *Hillary Clinton* tweets, which we use for development purposes. TaskB_Test is the TaskB test corpus on which we report results containing *Donald Trump* testing instances.

TaskB_Unlab is an unlabelled corpus containing *Donald Trump* tweets supplied by the task organisers, and TaskB_Auto-lab* is an automatically labelled version of a small portion of the corpus for the weakly supervised stance detection experiments reported in Section 6. Finally, Crawled_Unlab* is a corpus we collected for unsupervised pre-training (see Section 3.4).

For all experiments, the official task evaluation script is used. Predictions are post processed so that if the target is contained in a tweet, the highest-scoring non-neutral stance is chosen. This was motivated by the observation that in the training data most target-containing tweets express a stance, with only 16% of them being neutral. The code used in our experiments is available from `https://github.com/sheffieldnlp/stance-conditional`.

4.1 Methods

We compare the following baseline methods:

- SVM trained with word and character tweet n-grams features (SVM-ngrams-comb) Mohammad et al. (2016)
- a majority class baseline (Majority baseline), reported in (Mohammad et al., 2016)
- bag of word vectors (BoWV) (see Section 3.4)
- independent encoding of tweet and the target with two LSTMs (Concat) (see Section 3.1)
- encoding of the tweet only with an LSTM (TweetOnly) (see Section 3.1)

to three versions of conditional encoding:

- target conditioned on tweet (TarCondTweet)
- tweet conditioned on target (TweetCondTar)
- a bidirectional encoding model (BiCond)

5 Unseen Target Stance Detection

As explained earlier, the challenge is to learn a model without any manually labelled training data for the test target, but only using the data from the Task A targets. In order to avoid using any labelled data for *Donald Trump*, while still having a (labelled) development set to tune and evaluate our models, we used the tweets labelled for *Hillary Clinton* as a development set and the tweets for the remaining four targets as training. We refer to this as the *development setup*, and all models are tuned using this setup. The labelled *Donald Trump* tweets were only used in reporting our final results.

Method	Stance	P	R	F1
BoWV	FAVOR	0.2444	0.0940	0.1358
	AGAINST	0.5916	0.8626	0.7019
	Macro			0.4188
TweetOnly	FAVOR	0.2127	0.5726	0.3102
	AGAINST	0.6529	0.4020	0.4976
	Macro			0.4039
Concat	FAVOR	0.1811	0.6239	0.2808
	AGAINST	0.6299	0.4504	0.5252
	Macro			0.4030
TarCondTweet	FAVOR	0.3293	0.3649	0.3462
	AGAINST	0.4304	0.5686	0.4899
	Macro			0.4180
TweetCondTar	FAVOR	0.1985	0.2308	0.2134
	AGAINST	0.6332	0.7379	0.6816
	Macro			0.4475
BiCond	FAVOR	0.2588	0.3761	0.3066
	AGAINST	0.7081	0.5802	0.6378
	Macro			**0.4722**

Table 2: Results for the *unseen target* stance detection development setup.

For the final results we train on all the data from the development setup and evaluate on the official Task B test set, i.e. the *Donald Trump* tweets. We refer to this as our *test setup*.

Based on a small grid search using the development setup, the following settings for LSTM-based models were chosen: input layer size 100 (equal to the word embedding dimension), hidden layer size of 60, training for max 50 epochs with initial learning rate 1e-3 using ADAM (Kingma and Ba, 2014) for optimisation, dropout 0.1. Models were trained using cross-entropy loss. The use of one, relatively small hidden layer and dropout help to avoid overfitting.

5.1 Results and Discussion

Results for the unseen target setting show how well conditional encoding is suited for learning target-dependent representations of tweets, and crucially, how well such representations generalise to unseen targets. The best performing method on both development (Table 2) and test setups (Table 3) is BiCond, which achieves an F1 of 0.4722 and 0.4901 respectively. Notably, Concat, which learns an in-

Method	Stance	P	R	F1
BoWV	FAVOR	0.3158	0.0405	0.0719
	AGAINST	0.4316	0.8963	0.5826
	Macro			0.3272
TweetOnly	FAVOR	0.2767	0.3851	0.3220
	AGAINST	0.4225	0.5284	0.4695
	Macro			0.3958
Concat	FAVOR	0.3145	0.5270	0.3939
	AGAINST	0.4452	0.4348	0.4399
	Macro			0.4169
TarCondTweet	FAVOR	0.2322	0.4188	0.2988
	AGAINST	0.6712	0.6234	0.6464
	Macro			0.4726
TweetCondTar	FAVOR	0.3710	0.5541	0.4444
	AGAINST	0.4633	0.5485	0.5023
	Macro			0.4734
BiCond	FAVOR	0.3033	0.5470	0.3902
	AGAINST	0.6788	0.5216	0.5899
	Macro			**0.4901**

Table 3: Results for the *unseen target* stance detection test setup.

EmbIni	NumMatr	Stance	P	R	F1
Random	Sing	FAVOR	0.1982	0.3846	0.2616
		AGAINST	0.6263	0.5929	0.6092
		Macro			0.4354
	Sep	FAVOR	0.2278	0.5043	0.3138
		AGAINST	0.6706	0.4300	0.5240
		Macro			0.4189
PreFixed	Sing	FAVOR	0.6000	0.0513	0.0945
		AGAINST	0.5761	0.9440	0.7155
		Macro			0.4050
	Sep	FAVOR	0.1429	0.0342	0.0552
		AGAINST	0.5707	0.9033	0.6995
		Macro			0.3773
PreCont	Sing	FAVOR	0.2588	0.3761	0.3066
		AGAINST	0.7081	0.5802	0.6378
		Macro			**0.4722**
	Sep	FAVOR	0.2243	0.4103	0.2900
		AGAINST	0.6185	0.5445	0.5792
		Macro			0.4346

Table 4: Results for the *unseen target* stance detection development setup using BiCond, with single vs separate embeddings matrices for tweet and target and different initialisations

dependent encoding of the target and the tweets, does not achieve big F1 improvements over TweetOnly, which learns a representation of the tweets only. This shows that it is not sufficient to just take the target into account, but is is important to learn target-dependent encodings for the tweets. Models

Method	inTwe	Stance	P	R	F1
Concat	Yes	FAVOR	0.3153	0.6214	0.4183
		AGAINST	0.7438	0.4630	0.5707
		Macro			0.4945
	No	FAVOR	0.0450	0.6429	0.0841
		AGAINST	0.4793	0.4265	0.4514
		Macro			0.2677
TweetCondTar	Yes	FAVOR	0.3529	0.2330	0.2807
		AGAINST	0.7254	0.8327	0.7754
		Macro			0.5280
	No	FAVOR	0.0441	0.2143	0.0732
		AGAINST	0.4663	0.5588	0.5084
		Macro			0.2908
BiCond	Yes	FAVOR	0.3585	0.3689	0.3636
		AGAINST	0.7393	0.7393	0.7393
		Macro			**0.5515**
	No	FAVOR	0.0938	0.4286	0.1538
		AGAINST	0.5846	0.2794	0.3781
		Macro			0.2660

Table 5: Results for the *unseen target* stance detection development setup for tweets containing the target vs tweets not containing the target.

that learn to condition the encoding of tweets on targets outperform all baselines on the test set.

It is further worth noting that the Bag-of-Word-Vectors baseline achieves results comparable with TweetOnly, Concat and one of the conditional encoding models, TarCondTweet, on the dev set, even though it achieves significantly lower performance on the test set. This indicates that the pre-trained word embeddings on their own are already very useful for stance detection. This is consistent with findings of other works showing the usefulness of such a Bag-of-Word-Vectors baseline for the related tasks of recognising textual entailment Bowman et al. (2015) and sentiment analysis Eisner et al. (2016).

Our best result in the test setup with BiCond is the second highest reported result on the Twitter Stance Detection corpus, however the first, third and fourth best approaches achieved their results by automatically labelling *Donald Trump* training data. BiCond for the unseen target setting outperforms the third and fourth best approaches by a large margin (5 and 7 points in Macro F1, respectively), as can be seen in Table 7. Results for weakly supervised stance detection are discussed in Section 6.

Pre-Training Table 4 shows the effect of unsupervised pre-training of word embeddings with a word2vec skip-gram model, and furthermore, the results of sharing of these representations between the tweets and targets, on the development set. The first set of results is with a uniformly Random embedding initialisation in $[-0.1, 0.1]$. PreFixed uses the pre-trained skip-gram word embeddings, whereas PreCont initialises the word embeddings with ones from SkipGram and continues training them during LSTM training. Our results show that, in the absence of a large labelled training dataset, pre-training of word embeddings is more helpful than random initialisation of embeddings. Sing vs Sep shows the difference between using shared vs two separate embeddings matrices for looking up the word embeddings. Sing means the word representations for tweet and target vocabularies are shared, whereas Sep means they are different. Using shared embeddings performs better, which we hypothesise is because the tweets contain some mentions of targets that are tested.

Target in Tweet vs Not in Tweet Table 5 shows results on the development set for BiCond, compared to the best unidirectional encoding model, TweetCondTar and the baseline model Concat, split by tweets that contain the target and those that do not. All three models perform well when the target is mentioned in the tweet, but less so when the targets are not mentioned explicitly. In the case where the target is mentioned in the tweet, biconditional encoding outperforms unidirectional encoding and unidirectional encoding outperforms Concat. This shows that conditional encoding is able to learn useful dependencies between the tweets and the targets.

6 Weakly Supervised Stance Detection

The previous section showed the usefulness of conditional encoding for unseen target stance detection and compared results against internal baselines. The goal of experiments reported in this section is to compare against participants in the SemEval 2016 Stance Detection Task B. While we consider an *unseen target* setup, most submissions, including the three highest ranking ones for Task B, pkudblab (Wei et al., 2016), LitisMind (Zarrella and Marsh, 2016) and INF-UFRGS (Dias and Becker, 2016) considered a different experimental setup. They automatically annotated training data for the test target *Donald Trump*, thus converting the task into weakly supervised seen target stance detection. The pkudblab system uses a deep convolutional neural network that learns to make 2-way predictions on automatically labelled positive and negative training data for *Donald Trump*. The neutral class is predicted according to rules which are applied at test time.

Method	Stance	P	R	F1
BoWV	FAVOR	0.5156	0.6689	0.5824
	AGAINST	0.6266	0.3311	0.4333
	Macro			0.5078
TweetOnly	FAVOR	0.5284	0.6284	0.5741
	AGAINST	0.5774	0.4615	0.5130
	Macro			0.5435
Concat	FAVOR	0.5506	0.5878	0.5686
	AGAINST	0.5794	0.4883	0.5299
	Macro			0.5493
TarCondTweet	FAVOR	0.5636	0.6284	0.5942
	AGAINST	0.5947	0.4515	0.5133
	Macro			0.5538
TweetCondTar	FAVOR	0.5868	0.6622	0.6222
	AGAINST	0.5915	0.4649	0.5206
	Macro			0.5714
BiCond	FAVOR	0.6268	0.6014	0.6138
	AGAINST	0.6057	0.4983	0.5468
	Macro			**0.5803**

Table 6: Stance Detection test results for weakly supervised setup, trained on automatically labelled pos+neg+neutral Trump data, and reported on the official test set.

Since the best performing systems which participated in the shared task consider a weakly supervised setup, we further compare our proposed approach to the state-of-the-art using such a weakly supervised setup. Note that, even though pkudblab, LitisMind and INF-UFRGS also use regular expressions to label training data automatically, the resulting datasets were not available to us. Therefore, we had to develop our own automatic labelling method and dataset, which are publicly available from our code repository.

Weakly Supervised Test Setup For this setup, the unlabelled *Donald Trump* corpus TaskB_Unlab is annotated automatically. For this purpose we cre-

ated a small set of regular expressions[3], based on inspection of the TaskB_Unlab corpus, expressing positive and negative stance towards the target. The regular expressions for the positive stance were:

- make(?)america(?)great(?)again
- trump(?)(for|4)(?)president
- votetrump
- trumpisright
- the truth
- #trumprules

The keyphrases for negative stance were: #dumptrump, #notrump, #trumpwatch, racist, idiot, fired

A tweet is labelled as positive if one of the positive expressions is detected, else negative if a negative expressions is detected. If neither are detected, the tweet is annotated as neutral randomly with 2% chance. The resulting corpus size per stance is shown in Table 1. The same hyperparameters for the LSTM-based models are used as for the *unseen target* setup described in the previous section.

6.1 Results and Discussion

Table 6 lists our results in the weakly supervised setting. Table 7 shows all our results, including those using the unseen target setup, compared against the state-of-the-art on the stance detection corpus. Table 7 further lists baselines reported by Mohammad et al. (2016), namely a majority class baseline (Majority baseline), and a method using 1 to 3-gram bag-of-word and character n-gram features (SVM-ngrams-comb), which are extracted from the tweets and used to train a 3-way SVM classifier.

Bag-of-word baselines (BoWV, SVM-ngrams-comb) achieve results comparable to the majority baseline (F1 of 0.2972), which shows how difficult the task is. The baselines which only extract features from the tweets, SVM-ngrams-comb and TweetOnly perform worse than the baselines which also learn representations for the targets (BoWV, Concat). By training conditional encoding models on automatically labelled stance detection data we achieve state-of-the-art results. The best result (F1 of 0.5803) is achieved with the bi-directional conditional encoding model (BiCond). This shows that such models are suitable for unseen, as well as seen target stance detection.

[3]Note that "|" indiates "or", (?) indicates optional space

Method	Stance	F1
SVM-ngrams-comb (*Unseen Target*)	FAVOR	0.1842
	AGAINST	0.3845
	Macro	0.2843
Majority baseline (*Unseen Target*)	FAVOR	0.0
	AGAINST	0.5944
	Macro	0.2972
BiCond (*Unseen Target*)	FAVOR	0.3902
	AGAINST	0.5899
	Macro	**0.4901**
INF-UFRGS (*Weakly Supervised**)	FAVOR	0.3256
	AGAINST	0.5209
	Macro	0.4232
LitisMind (*Weakly Supervised**)	FAVOR	0.3004
	AGAINST	0.5928
	Macro	0.4466
pkudblab (*Weakly Supervised**)	FAVOR	0.5739
	AGAINST	0.5517
	Macro	0.5628
BiCond (*Weakly Supervised*)	FAVOR	0.6138
	AGAINST	0.5468
	Macro	**0.5803**

Table 7: Stance Detection test results, compared against the state of the art. SVM-ngrams-comb and Majority baseline are reported in Mohammad et al. (2016), pkudblab in Wei et al. (2016), LitisMind in Zarrella and Marsh (2016), INF-UFRGS in Dias and Becker (2016)

7 Related Work

Stance Detection: Previous work mostly considered target-specific stance prediction in debates (Hasan and Ng, 2013; Walker et al., 2012) or student essays (Faulkner, 2014). The task considered in this paper is more challenging than stance detection in debates because, in addition to irregular language, the Mohammad et al. (2016) dataset is offered without any context, e.g., conversational structure or tweet metadata. The targets are also not always mentioned in the tweets, which is an additional challenge (Augenstein et al., 2016) and distinguishes this task from target-dependent (Vo and Zhang, 2015; Zhang et al., 2016; Alghunaim et al., 2015) and open-domain target-dependent sentiment analysis (Mitchell et al., 2013; Zhang et al., 2015). Related work on rumour stance detection either requires training data from the same ru-

mour (Qazvinian et al., 2011), i.e., target, or is rule-based (Liu et al., 2015) and thus potentially hard to generalise. Finally, the target-dependent stance detection task tackled in this paper is different from that of Ferreira and Vlachos (2016), which while related concerned with the stance of a statement in natural language towards another statement.

Conditional Encoding: Conditional encoding has been applied to the related task of recognising textual entailment (Rocktäschel et al., 2016), using a dataset of half a million training examples (Bowman et al., 2015) and numerous different hypotheses. Our experiments here show that conditional encoding is also successful on a relatively small training set and when applied to an unseen testing target. Moreover, we augment conditional encoding with bidirectional encoding and demonstrate the added benefit of unsupervised pre-training of word embeddings on unlabelled domain data.

8 Conclusions and Future Work

This paper showed that conditional LSTM encoding is a successful approach to stance detection for unseen targets. Our unseen target bidirectional conditional encoding approach achieves the second best results reported to date on the SemEval 2016 Twitter Stance Detection corpus. In the weakly supervised seen target scenario, as considered by prior work, our approach achieves the best results to date on the SemEval Task B dataset. We further show that in the absence of large labelled corpora, unsupervised pre-training can be used to learn target representations for stance detection and improves results on the SemEval corpus. Future work will investigate further the challenge of stance detection for tweets which do not contain explicit mentions of the target.

Acknowledgments

This work was partially supported by the European Union under grant agreement No. 611233 PHEME[4] and by Microsoft Research through its PhD Scholarship Programme.

[4] http://www.pheme.eu

References

Abdulaziz Alghunaim, Mitra Mohtarami, Scott Cyphers, and Jim Glass. 2015. A Vector Space Approach for Aspect Based Sentiment Analysis. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 116–122, Denver, Colorado, June. Association for Computational Linguistics.

Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. 2016. USFD: Any-Target Stance Detection on Twitter with Autoencoders. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

Marcelo Dias and Karin Becker. 2016. INF-UFRGS-OPINION-MINING at SemEval-2016 Task 6: Automatic Generation of a Training Corpus for Unsupervised Identification of Stance in Tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning Emoji Representations from their Description. In *Proceedings of the International Workshop on Natural Language Processing for Social Media*, SocialNLP '16, Austin, Texas.

Adam Faulkner. 2014. Automated Classification of Stance in Student Essays: An Approach Using Stance Target Information and the Wikipedia Link-Based Measure. In William Eberle and Chutima Boonthum-Denecke, editors, *FLAIRS Conference*. AAAI Press.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, California, June. Association for Computational Linguistics.

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

Kazi Saidul Hasan and Vincent Ng. 2013. Stance Classification of Ideological Debates: Data, Models, Features, and Constraints. In *IJCNLP*, pages 1348–1356. Asian Federation of Natural Language Processing / ACL.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time Rumor Debunking on Twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1867–1870, New York, NY, USA. ACM.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter Under Crisis: Can We Trust What We RT? In *Proceedings of the First Workshop on Social Media Analytics (SOMA'2010)*, pages 71–79, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open Domain Targeted Sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654, Seattle, Washington, USA, October. Association for Computational Linguistics.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor Has It: Identifying Misinformation in Microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1589–1599.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *International Conference on Learning Representations (ICLR)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Duy-Tin Vo and Yue Zhang. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features. In Qiang Yang and Michael Wooldridge, editors, *IJCAI*, pages 1347–1353. AAAI Press.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance Classification using Dialogic Properties of Persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at SemEval-2016 Task 6: A Specific Convolutional Neural Network System for Effective Stance Detection. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural Networks for Open Domain Targeted Sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 612–621, Lisbon, Portugal, September. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, February. Association for the Advancement of Artificial Intelligence.

Modeling Skip-Grams for Event Detection with Convolutional Neural Networks

Thien Huu Nguyen
Computer Science Department
New York University
New York, NY 10003 USA
thien@cs.nyu.edu

Ralph Grishman
Computer Science Department
New York University
New York, NY 10003 USA
grishman@cs.nyu.edu

Abstract

Convolutional neural networks (CNN) have achieved the top performance for event detection due to their capacity to induce the underlying structures of the k-grams in the sentences. However, the current CNN-based event detectors only model the consecutive k-grams and ignore the non-consecutive k-grams that might involve important structures for event detection. In this work, we propose to improve the current CNN models for ED by introducing the non-consecutive convolution. Our systematic evaluation on both the general setting and the domain adaptation setting demonstrates the effectiveness of the non-consecutive CNN model, leading to the significant performance improvement over the current state-of-the-art systems.

1 Introduction

The goal of event detection (ED) is to locate event triggers of some specified types in text. Triggers are generally single verbs or nominalizations that evoke the events of interest. This is an important and challenging task of information extraction in natural language processing (NLP), as the same event might appear in various expressions, and an expression might express different events depending on contexts.

The current state-of-the-art systems for ED have involved the application of convolutional neural networks (CNN) (Nguyen and Grishman, 2015b; Chen et al., 2015) that automatically learn effective feature representations for ED from sentences. This has overcome the two fundamental limitations of the traditional feature-based methods for ED: (i) the complicated feature engineering for rich feature sets and (ii) the error propagation from the NLP toolkits and resources (i.e, parsers, part of speech taggers etc) that generate such features.

The prior CNN models for ED are characterized by the temporal convolution operators that linearly map the vectors for the k-grams in the sentences into the feature space. Such k-gram vectors are obtained by concatenating the vectors of the k consecutive words in the sentences (Nguyen and Grishman, 2015b; Chen et al., 2015). In other words, the previous CNN models for ED only focus on modeling the consecutive k-grams. Unfortunately, such consecutive mechanism is unable to capture the long-range and non-consecutive dependencies that are necessary to the prediction of trigger words. For instance, consider the following sentence with the trigger word "*leave*" from the ACE 2005 corpus:

*The mystery is that she took the job in the first place or didn't **leave** earlier.*

The correct event type for the trigger word "*leave*" in this case is "*End-Org*". However, the previous CNN models might not be able to detect "*leave*" as an event trigger or incorrectly predict its type as "*Movement*". This is caused by their reliance on the consecutive local k-grams such as "*leave earlier*". Consequently, we need to resort to the non-consecutive pattern "*job leave*" to correctly determine the event type of "*leave*" in this case.

Guided by this intuition, we propose to improve the previous CNN models for ED by operating the convolution on all possible non-consecutive k-grams

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 886–891,
Austin, Texas, November 1-5, 2016.

in the sentences. We aggregate the resulting convolution scores via the *max-pooling* function to unveil the most important non-consecutive k-grams for ED. The aggregation over all the possible non-consecutive k-grams is made efficient with dynamic programming.

Note that our work is related to (Lei et al., 2015) who employ the non-consecutive convolution for the sentence and news classification problems. Our work is different from (Lei et al., 2015) in that we model the relative distances of words to the trigger candidates in the sentences via position embeddings, while (Lei et al., 2015) use the absolute distances between words in the k-grams to compute the decay weights for aggregation. To the best of our knowledge, this is the first work on non-consecutive CNN for ED.

We systematically evaluate the proposed model in the general setting as well as the domain adaptation setting. The experiment results demonstrate that our model significantly outperforms the current state-of-the-art models in such settings.

2 Model

We formalize ED as a multi-class classification problem. Given a sentence, for every token in that sentence, we want to predict if the current token is an event trigger of some event in the pre-defined event set or not? The current token along with its context in the sentence constitute an event trigger candidate.

In order to make it compatible with the previous work, we follow the procedure in (Nguyen and Grishman, 2015b) to process the trigger candidates for CNN. In particular, we limit the context of the trigger candidates to a fixed window size by trimming longer sentences and padding shorter sentences with a special token when necessary. Let $2n + 1$ be the fixed window size, and $W = [w_0, w_1, \ldots, w_n, \ldots, w_{2n-1}, w_{2n}]$ be some trigger candidate where the current token is positioned in the middle of the window (token w_n). Before entering CNN, each token w_i is first transformed into a real-valued vector x_i using the concatenation of the following vectors:

1. The word embedding vector of w_i: This is obtained by looking up a pre-trained word embedding table D (Turian et al., 2010; Mikolov et al., 2013a).

2. The position embedding vector of w_i: We obtain this vector by looking up the position embedding table for the relative distance $i - n$ from the token w_i to the current token w_n. The position embedding table is initialized randomly.

3. The real-valued embedding vector for the entity type of w_i: This vector is generated by looking up the entity type embedding table (initialized randomly) for the entity type of w_i. Note that we employ the BIO annotation schema to assign entity type labels to each token in the sentences using the entity mention heads as in (Nguyen and Grishman, 2015b).

The transformation from the token w_i to the vector x_i ($x_i \in \mathbb{R}^d$) essentially converts the input candidate W into a sequence of real-valued vectors $X = (x_0, x_1, \ldots, x_{2n})$. This sequence is used as input in the following CNN models.

2.1 The Traditional CNN

Given the window size k, the traditional CNN models for ED consider the following set of $2n + 1$ consecutive k-gram vectors:

$$C = \{u_i : 0 \leq i \leq 2n\} \quad (1)$$

Vector u_i is the concatenation of the k consecutive vectors preceding position i in the sequence X: $u_i = [x_{i-k+1}, x_{i-k+2}, \ldots, x_i] \in \mathbb{R}^{dk}$ where the out-of-index vectors are simply set to all zeros.

The core of the CNN models is the convolution operation, specified by the filter vector $\mathbf{f} \in \mathbb{R}^{dk}$. In CNN, $\mathbf{f}$ can be seen as a feature extractor for the k-grams that operates via the dot product with each element in C. This produces the following convolution score set: $S(C) = \{\mathbf{f}^T u_i : 0 \leq i \leq 2n\}$.

In the next step, we aggregate the features in S with the $\max$ function, resulting in the aggregation score:

$$p_k^{\mathbf{f}} = \max S(C) = \max\{s_i : 0 \leq i \leq 2n\} \quad (2)$$

Afterward, $p_k^{\mathbf{f}}$ is often transformed by a non-linear function G[1] to generate the transformed score $G(p_k^{\mathbf{f}})$, functioning as the extracted feature for the initial trigger candidate W.

[1] The `tanh` function in this work.

We can then repeat this process for different window sizes k and filters $\mathbf{f}$, generating multiple features $G(p_k^{\mathbf{f}})$ to capture various aspects of the trigger candidate W. Finally, such features are concatenated into a single representation vector for W, to be fed into a feed-forward neural network with a softmax layer in the end to perform classification.

2.2 The Non-consecutive CNN

As mentioned in the introduction, the limitation of the previous CNN models for ED is the inability to encode the non-consecutive k-grams that might be crucial to the trigger prediction. This limitation originates from Equation 1 in which only the consecutive k-gram vectors are considered. In order to overcome such limitation, we propose to model all possible non-consecutive k-grams in the trigger candidate, leading to the following set of non-consecutive k-gram vectors:

$$N = \{v_{i_1 i_2 \ldots i_k} : 0 \leq i_1 < i_2 < \ldots < i_k \leq 2n\}$$

where: $v_{i_1 i_2 \ldots i_k} = [x_{i_1}, x_{i_2}, \ldots, x_{i_k}] \in \mathbb{R}^{dk}$ and the number of elements in N is $|N| = \binom{2n+1}{k}$.

The non-consecutive CNN model then follows the procedure of the traditional CNN model in Section 2.1 to compute the representation vector for classification. The only difference is that the computation is done on the input set N instead of C. In particular, the convolution score set in this case would be $S(N) = \{\mathbf{f}^T v : v \in N\}$, while the aggregating score would be:

$$p_k^{\mathbf{f}} = \max S(N) = \max\{s : s \in S(N)\} \quad (3)$$

2.3 Implementation

Note that the maximum operation in Equation 2 only requires $O(n)$ operations while the naive implementation of Equation 3 would need $O(|N|) = O(n^k)$ operations. In this work, we employ the dynamic programming (DP) procedure below to reduce the computation time for Equation 3.

Assuming the filter vector $\mathbf{f}$ is the concatenation of the k vectors $\mathbf{f}_1, \ldots, \mathbf{f}_k \in \mathbb{R}^d$: $\mathbf{f} = [\mathbf{f}_1, \ldots, \mathbf{f}_k]$, Equation 3 can be re-written by:

$$\begin{aligned} p_k^{\mathbf{f}} = &\max\{\mathbf{f}_1^T x_{i_1} + \ldots + \mathbf{f}_k^T x_{i_k} \\ &: 0 \leq i_1 < i_2 < \ldots < i_k \leq 2n\} \end{aligned}$$

Let D_t^j be the dynamic programming table representing the maximum convolution score for the subfilter $[\mathbf{f}_1, \ldots, \mathbf{f}_j]$ over all possible non-consecutive j-gram vectors in the subsequence $(x_0, x_1, \ldots, x_t)$ of X:

$$\begin{aligned} D_t^j = &\max\{\mathbf{f}_1^T x_{i_1} + \ldots + \mathbf{f}_j^T x_{i_j} \\ &: 0 \leq i_1 < i_2 < \ldots < i_j \leq t\} \end{aligned}$$

where $1 \leq j \leq k$, $j - 1 \leq t \leq 2n$.

Note that $p_k^{\mathbf{f}} = D_{2n}^k$.

We can solve this DP problem by the following recursive formulas[2]:

$$D_t^j = \max\{D_{t-1}^j, D_{t-1}^{j-1} + \mathbf{f}_j^T x_t\}$$

The computation time for this procedure is $O(kn)$ and remains linear in the sequence length.

2.4 Training

We train the networks using stochastic gradient descent with shuffled mini-batches, the AdaDelta update rule, back-propagation and dropout. During the training, we also optimize the embedding tables (i.e, word, position and entity type embeddings) to achieve the optimal states. Finally, we rescale the weights whose l_2-norms exceed a predefined threshold (Nguyen and Grishman (2015a)).

3 Experiments

3.1 Dataset, Parameters and Resources

We apply *the same parameters and resources* as (Nguyen and Grishman, 2015b) to ensure the compatible comparison. Specifically, we employ the window sizes in the set {2, 3, 4, 5} for the convolution operation with 150 filters for each window size. The window size of the trigger candidate is 31 while the dimensionality of the position embeddings and entity type embeddings is 50. We use `word2vec` from (Mikolov et al., 2013b) as the pretrained word embeddings. The other parameters include the dropout rate $\rho = 0.5$, the mini-batch size = 50, the predefined threshold for the l_2 norms = 3.

Following the previous studies (Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2015b), we evaluate the models on the ACE 2005 corpus

[2]We ignore the base cases as they are trivial.

with 33 event subtypes. In order to make it compatible, we use the same test set with 40 newswire articles, the same development set with 30 other documents and the same training set with the remaining 529 documents. All the data preprocessing and evaluation criteria follow those in (Nguyen and Grishman, 2015b).

3.2 The General Setting

We compares the non-consecutive CNN model (**NC-CNN**) with the state-of-the-art systems on the ACE 2005 dataset in Table 1. These systems include:

1) The feature-based systems with rich hand-designed feature sets, including: the MaxEnt model with local features in (Li et al., 2013) (**MaxEnt**); the structured perceptron model for joint beam search with local features (**Joint+Local**), and with both local and global features (**Joint+Local+Global**) in (Li et al., 2013); and the sentence-level and cross-entity models in (Hong et al., 2011).

2) The neural network models, i.e, the CNN model in (Nguyen and Grishman, 2015b) (**CNN**), the dynamic multi-pooling CNN model (**DM-CNN**) in (Chen et al., 2015) and the bidirectional recurrent neural networks (**B-RNN**) in (Nguyen et al., 2016a).

3) The probabilistic soft logic based model to capture the event-event correlation in (Liu et al., 2016).

Methods	F
Sentence-level in Hong et al (2011)	59.7
MaxEnt (Li et al., 2013)	65.9
Joint+Local (Li et al., 2013)	65.7
Joint+Local+Global (Li et al., 2013)	67.5
Cross-entity in Hong et al. (2011) †	68.3
Probabilistic soft logic (Liu et al., 2016) †	69.4
CNN (Nguyen and Grishman, 2015b)	69.0
DM-CNN (Chen et al., 2015)	69.1
B-RNN (Nguyen et al., 2016a)	69.3
NC-CNN	**71.3**

Table 1: Performance with Gold-Standard Entity Mentions and Types. † beyond sentence level.

The most important observation from the table is that the non-consecutive CNN model significantly outperforms all the compared models with large margins. In particular, *NC-CNN* is 2% better than B-RNN (Nguyen et al., 2016a), the state-of-the-art system that only relies on the context information within the sentences of the trigger candidates. In addition, although *NC-CNN* only employs the sentence-level information, it is still better than the other models that further exploit the document-level information for prediction (an improvement of 1.9% over the probabilistic soft logic based model in (Liu et al., 2016)). Finally, comparing *NC-CNN* and the CNN model in (Nguyen and Grishman, 2015b), we see that the non-consecutive mechanism significantly improves the performance of the traditional CNN model for ED (up to 2.3% in absolute F-measures with $p < 0.05$).

3.3 The Domain Adaptation Experiments

Previous studies have shown that the NLP models would suffer from a significant performance loss when domains shift (Blitzer et al., 2006; Daume III, 2007; Plank and Moschitti, 2013; Nguyen et al., 2015c). In particular, if a model is trained on some *source domain* and applied to a different domain (*the target domain*), its performance would degrade significantly. The domain adaptation (DA) studies aim to overcome this issue by developing robust techniques across domains.

The best reported system in the DA setting for ED is (Nguyen and Grishman, 2015b), which demonstrated that the CNN model outperformed the feature-based models in the cross-domain setting. In this section, we compare *NC-CNN* with the CNN model in (Nguyen and Grishman, 2015b) (as well as the other models above) in the DA setting to further investigate their effectiveness.

3.3.1 Dataset

This section also uses the ACE 2005 dataset but focuses more on the difference between domains. The ACE 2005 corpus includes 6 different domains: broadcast conversation (*bc*), broadcast news (*bn*), telephone conversation (*cts*), newswire (*nw*), usenet (*un*) and webblogs (*wl*). Following (Nguyen and Grishman, 2015b), we use ***news*** (the union of ***bn*** and ***nw***) as the source domain and ***bc***, ***cts***, ***wl*** and ***un*** as four different target domains[3]. We take half of *bc* as the development set and use the remaining data for testing. Our data split is the same as that in (Nguyen and Grishman, 2015b).

[3]Note that (Nguyen and Grishman, 2015b) does not report the performance on *un* but we include it here for completeness.

System	In-domain(bn+nw)			bc			cts			wl			un		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
MaxEnt	74.5	59.4	66.0	70.1	54.5	61.3	66.4	49.9	56.9	59.4	34.9	43.9	-	-	-
Joint+Local	73.5	62.7	67.7	70.3	57.2	63.1	64.9	50.8	57.0	59.5	38.4	46.7	-	-	-
Joint+Local+Global	72.9	63.2	67.7	68.8	57.5	62.6	64.5	52.3	57.7	56.4	38.5	45.7	-	-	-
B-RNN	71.4	63.5	67.1	70.7	62.1	66.1	70.0	54.4	61.0	52.7	38.3	44.2	66.2	46.0	54.1
DM-CNN	75.9	62.7	68.7	75.3	59.3	66.4	74.8	52.3	61.5	59.2	37.4	45.8	72.2	44.5	55.0
CNN	69.2	67.0	68.0	70.2	65.2	67.6	68.3	58.2	62.8	54.8	42.0	**47.5**	64.6	49.9	56.2
NC-CNN	74.9	66.5	**70.4**†	73.6	64.7	**68.8**†	71.7	57.3	**63.6**	57.8	40.3	47.4	71.7	49.0	**58.1**†

Table 2: Performance on the source domain and on the target domains. Cells marked with †designates that NC-CNN significantly outperforms ($p < 0.05$) all the compared methods on the specified domain.

3.3.2 Performance

Table 2 reports the performance of the systems with 5-fold cross validation. Note that we focus on the systems exploiting only the sentence level information in this section. For each system, we train a model on the training data of the source domain and evaluate this model on the test set of the source domain (in-domain performance) as well as on the four target domains *bc*, *cts*, *wl* and *un*.

We emphasize that the performance of the systems *MaxEnt*, *Joint+Local*, *Joint+Local+Global*, *B-RNN*, and *CNN* is obtained from the actual systems in the original work (Li et al., 2013; Nguyen and Grishman, 2015b; Nguyen et al., 2016a). The performance of *DM-CNN*, on the other hand, is from our re-implementation of the system in (Chen et al., 2015) using the same hyper-parameters and resources as *CNN* and *NC-CNN* for a fair comparison.

From the table, we see that *NC-CNN* is significantly better than the other models on the source domain. This is consistent with the conclusions in Section 3.2 and further confirms the effectiveness of *NC-CNN*. More importantly, *NC-CNN* outperforms *CNN* and the other models on the target domains *bc*, *cts* and *un*, and performs comparably with *CNN* on *wl*. The performance improvement is significant on *bc* and *un* ($p < 0.05$), thereby verifying the robustness of *NC-CNN* for ED across domains.

4 Related Work

There have been three major approaches to event detection in the literature. First, the pattern-based approach explores the application of patterns to identify the instances of events, in which the patterns are formed by predicates, event triggers and constraints on the syntactic context (Grishman et al., 2005; Cao et al., 2015a; Cao et al., 2015b).

Second, the feature-based approach relies on linguistic intuition to design effective feature sets for statistical models for ED, ranging from the local sentence-level representations (Ahn, 2006; Li et al., 2013), to the higher level structures such as the cross-sentence or cross-event information (Ji and Grishman, 2008; Gupta and Ji, 2009; Patwardhan and Riloff, 2009; Liao and Grishman, 2011; Hong et al., 2011; McClosky et al., 2011; Li et al., 2015). Some recent work on the feature-based approach has also investigated event trigger detection in the joint inference with event argument prediction (Riedel et al., 2009; Poon and Vanderwende, 2010; Li et al., 2013; Venugopal et al., 2014) to benefit from their inter-dependencies.

Finally, neural networks have been introduced into ED very recently with the early work on convolutional neural networks (Nguyen and Grishman, 2015b; Chen et al., 2015). The other work includes: (Nguyen et al., 2016a) who employ bidirectional recurrent neural networks to perform event trigger and argument labeling jointly, (Jagannatha and Yu, 2016) who extract event instances from health records with recurrent neural networks and (Nguyen et al., 2016b) who propose a two-stage training algorithm for event extension with neural networks.

5 Conclusion

We present a new CNN architecture for ED that exploits the non-consecutive convolution for sentences. Our evaluation of the proposed model on the general setting and the DA setting demonstrates the effectiveness of the non-consecutive mechanism. We achieve the state-of-the-art performance for ED in both settings. In the future, we plan to investigate the non-consecutive architecture on other problems such as relation extraction or slot filling.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Kai Cao, Xiang Li, and Ralph Grishman. 2015a. Improving event detection with dependency regularization. In *RANLP*.

Kai Cao, Xiang Li, Miao Fan, and Ralph Grishman. 2015b. Improving event detection with active learning. In *RANLP*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. In *ACE 2005 Evaluation Workshop*.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Abhyuday N Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *NAACL*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS)*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *AAAI*.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *BioNLP Shared Task Workshop*.

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

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

Thien Huu Nguyen and Ralph Grishman. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015c. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP)*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL-HLT*.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *BioNLP 2009 Workshop*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*.

Porting an Open Information Extraction System from English to German

Tobias Falke[†] **Gabriel Stanovsky**[‡] **Iryna Gurevych**[†] **Ido Dagan**[‡]

[†]Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt

[‡]Natural Language Processing Lab
Department of Computer Science, Bar-Ilan University

Abstract

Many downstream NLP tasks can benefit from Open Information Extraction (Open IE) as a semantic representation. While Open IE systems are available for English, many other languages lack such tools. In this paper, we present a straightforward approach for adapting PropS, a rule-based predicate-argument analysis for English, to a new language, German. With this approach, we quickly obtain an Open IE system for German covering 89% of the English rule set. It yields 1.6 n-ary extractions per sentence at 60% precision, making it comparable to systems for English and readily usable in downstream applications.[1]

1 Introduction

The goal of Open Information Extraction (Open IE) is to extract coherent propositions from a sentence, each represented as a tuple of a relation phrase and one or more argument phrases (e.g., *born in (Barack Obama; Hawaii)*). Open IE has been shown to be useful for a wide range of semantic tasks, including question answering (Fader et al., 2014), summarization (Christensen et al., 2013) and text comprehension (Stanovsky et al., 2015), and has consequently drawn consistent attention over the last years (Banko et al., 2007; Wu and Weld, 2010; Fader et al., 2011; Akbik and Löser, 2012; Mausam et al., 2012; Del Corro and Gemulla, 2013; Angeli et al., 2015).

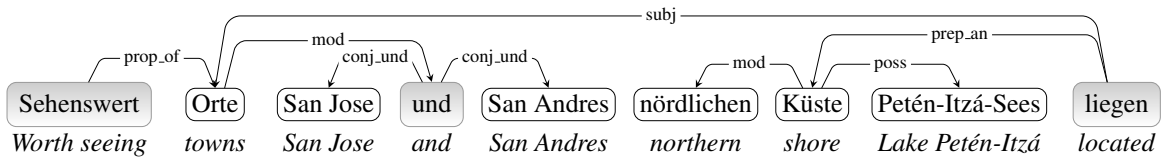
Although similar applications of Open IE in other languages are obvious, most previous work focused on English, with only a few recent exceptions (Zhila and Gelbukh, 2013; Gamallo and Garcia, 2015). For most languages, Open IE systems are still missing. While one could create them from scratch, as it was done for Spanish, this can be a very laborious process, as state-of-the-art systems make use of handcrafted, linguistically motivated rules. Instead, an alternative approach is to transfer the rule sets of available systems for English to the new language.

In this paper, we study whether an existing set of rules to extract Open IE tuples from English dependency parses can be ported to another language. We use German, a relatively close language, and the PropS system (Stanovsky et al., 2016) as examples in our analysis. Instead of creating rule sets from scratch, such a transfer approach would simplify the rule creation, making it possible to build Open IE systems for other languages with relatively low effort in a short amount of time. However, challenges we need to address are differences in syntax, dissimilarities in the corresponding dependency representations as well as language-specific phenomena. Therefore, the existing rules cannot be directly mapped to the German part-of-speech and dependency tags in a fully automatic way, but require a careful analysis as carried out in this work. Similar manual approaches to transfer rule-based systems to new languages were shown to be successful, e.g. for temporal tagging (Moriceau and Tannier, 2014), whereas fully automatic approaches led to less competitive systems (Strötgen and Gertz, 2015).

Our analysis reveals that a large fraction of the PropS rule set can be easily ported to German, requiring only small adaptations. With roughly 10%

[1]Source code and online demo available at `https://github.com/UKPLab/props-de`

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 892–898,
Austin, Texas, November 1-5, 2016.



Extraction 1: liegen (die Orte San Jose und San Andres ; an der nördlichen Küste des Petén-Itzá-Sees)

Extraction 2: sehenswert (die Orte San Jose und San Andres)

Figure 1: PropS representation for *Worth seeing are the towns San Jose and San Andres, which are located on the northern shore of Lake Petén-Itzá.* Grey boxes indicate predicates. Two Open IE tuples, one unary and one binary, are extracted from this sentence.

of the effort that went into the English system, we could build a system for German covering 89% of the rule set. As a result, we present *PropsDE*, the first Open IE system for German. In an intrinsic evaluation, we show that its performance is comparable with systems for English, yielding 1.6 extractions per sentence with an overall precision of 60%.

2 Background

Open Information Extraction Open IE was introduced as an open variant of traditional Information Extraction (Banko et al., 2007). Since its inception, several extractors were developed. The majority of them, namely ReVerb (Fader et al., 2011), KrakeN (Akbik and Löser, 2012), Exemplar (Mesquita et al., 2013) and ClausIE (Del Corro and Gemulla, 2013), successfully used rule-based strategies to extract tuples. Alternative approaches are variants of self-supervision, as in TextRunner (Banko et al., 2007), WOE (Wu and Weld, 2010) and OLLIE (Mausam et al., 2012), and semantically-oriented approaches utilizing semantic role labeling (Open IE-4[2]) or natural logic (Angeli et al., 2015). While TextRunner and ReVerb require only POS tagging as preprocessing to allow a high extraction speed, the other systems rely on dependency parsing to improve the extraction precision.

For non-English Open IE, ExtrHech has been presented for Spanish (Zhila and Gelbukh, 2013). Similar as the English systems, it uses a set of extraction rules, specifically designed for Spanish in this case. More recently, ArgOE (Gamallo and Garcia, 2015) was introduced. It manages to extract tuples in several languages with the same rule set, relying on a dependency parser that uses a common tagset for five European languages. However, an evaluation for English and Spanish revealed that this approach cannot compete with the systems specifically built for those languages. To the best of our knowledge, no work on Open IE for German exists.

Open IE with PropS Stanovsky et al. (2016) recently introduced PropS, a rule-based converter turning dependency graphs for English into typed graphs of predicates and arguments. An example is shown in Figure 1 (in German). Compared to a dependency graph, the representation masks non-core syntactic details, such as tense or determiners, unifies semantically equivalent constructions, such as active/passive, and explicates implicit propositions, such as indicated by possessives or appositions.

The resulting graph can be used to extract Open IE tuples in a straightforward way. Every non-nested predicate node $pred$ in the graph, together with its n argument-subgraphs arg_i, yields a tuple $pred(arg_1; ...; arg_n)$. With this approach, PropS is most similar to KrakeN and ClausIE, applying rules to a dependency parse. However, due to additional nodes for implicit predicates, it can also make extractions that go beyond the scope of other systems, such as *has (Michael; bicycle)* from *Michael's bicycle is red.* In line with more recent Open IE systems, this strategy extracts tuples that are not necessarily binary, but can be unary or of higher arity.

3 Analysis of Portability

Approach For each rule of the converter that transforms a dependency graph to the PropS graph, we assess its applicability for German. A rule is applied to a part of the graph if certain conditions are

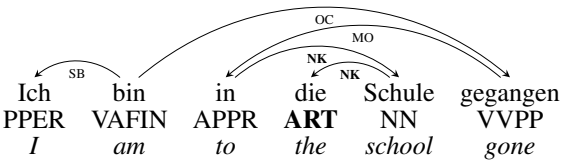
[2] https://github.com/knowitall/openie

fulfilled, expressed using dependency types, POS tags and lemmas. As we already pointed out in the introduction, several differences between the dependency and part-of-speech representations for English and German make a fully automatic translation of these rules impossible. We therefore manually analyzed the portability of each rule and report the findings in the next section.

While using Universal Dependencies (Nivre et al., 2016) could potentially simplify porting the rules, we chose not to investigate this option due to the ongoing nature of the project and focused on the established representations for now. In line with the English system, that works on collapsed Stanford dependencies (de Marneffe and Manning, 2008), we assume a similar input representation for German that can be obtained with a set of collapsing and propagation rules provided by Ruppert et al. (2015) for TIGER dependencies (Seeker and Kuhn, 2012).

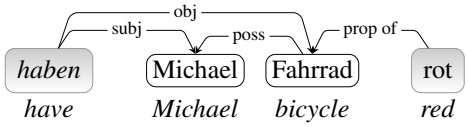
Findings Overall, we find that most rules can be used for German, mainly because syntactic differences, such as freer word order (Kübler, 2008), are already masked by the dependency representation (Seeker and Kuhn, 2012). About **38%** of the rule set can be **directly ported** to German, solely replacing dependency types, POS tags and lemmas with their German equivalents. As an example, the rule removing negation tokens looks for *neg* dependencies in the graph, for which a corresponding type *NG* exists in German. We found similar correspondences to remove punctuation and merge proper noun and number compounds. In addition, we can also handle appositions and existentials with direct mappings.

For **35%** of the English rules, **small changes** are necessary, mainly because no direct mapping to the German tag set is possible or the annotation style differs. For instance, while English has a specific type *det* to link determiners to their governor, a more generic type (*NK*) is used in German. Instead, determiners can be detected by part-of-speech:



Another type of difference exists with regard to the representation of auxiliary verb constructions. In Stanford dependencies, main verbs govern all auxiliaries, whereas in TIGER dependencies, an auxiliary heads the main verb. The above example shows this for *gone* and *am*. Therefore, all rules identifying and removing auxiliaries and modals have to be adapted to account for this difference.

With similar changes as discussed for determiners, we can also handle possessive and copular constructions. The graph for *Michael's bicycle is red*, for example, features an additional predicate *have* to explicate the implicit possessive relation, while *red* becomes an adjectival predicate, omitting *is*:



Moreover, conditional constructions can be processed with slight changes as well. Missing a counterpart for the type *mark*, we instead look for subordinating conjunctions by part-of-speech. In fact, we found conditionals to be represented more consistently across different conjunctions, making their handling in German easier than in English.

More **substantial changes** are necessary for the remaining **27%** of the rules. To represent active and passive in a uniform way, in passive clauses, PropS turns the subject into an object and a potential by-clause into the subject. For English, these cases are indicated by the presence of passive dependencies such as *nsubjpass*. For German, however, no counterparts exist. As an alternative strategy, we instead look for past participle verbs (by POS tag) that are governed by a form of the auxiliary *werden* (Schäfer, 2015). Instances of the German static passive (Zustandspassiv) are, in contrast, handled like copulas. Another deviation from the English system is necessary for relative clauses. PropS heavily relies on the Stanford dependency converter, which propagates dependencies of the relative pronoun to its referent. The German collapser does not have this feature, and we therefore implement it as an additional transformation (see *subj(liegen;Orte)* in Figure 1).

To abstract away from different tenses, PropS represents predicates with their lemma, indicating the original tense as a feature, as detected with a set of rules operating on POS tags. For German, no tense information is contained in POS tags, but instead, a morphological analysis can provide it. Determining

the overall tense of a sentence based on that requires a new set of rules, as the grammatical construction of tenses differs between German and English. PropS also tries to heuristically identify raising constructions, in which syntactic and semantic roles of arguments differ. In German, this phenomenon occurs in similar situations, such as in *Michael scheint zu lächeln* (*Michael seems to smile*), in which *Michael* is not the semantic subject of *scheinen*, though syntactically it is. To determine these cases heuristically, an empirically derived list of common raising verbs, such as done by Chrupała and van Genabith (2007) for English, needs to be created.

An **additional** step that is necessary during the lemmatization of verbs for German is to recover separated particles. For example, a verb like *ankommen* (*arrive*) can be split in a sentence such as *Er kam an* (*He arrived*), moving the particle to the end of the sentence, with a potentially large number of other tokens in between. We can reliably reattach these particles based on the dependency parse. Another addition to the rules that we consider important is to detect subjunctive forms of verbs and indicate the mood with a specific feature for the predicate. A morphological analysis provides the necessary input. Compared to English, the usage of the subjunctive is much more common, usually to indicate either unreality or indirect speech (Thieroff, 2004).

4 German Open IE System

Following our analysis, we implemented a German version of PropS, named PropsDE. It uses matetools for POS tagging, lemmatizing and parsing (Bohnet et al., 2013). Dependencies are collapsed and propagated with JoBimText (Ruppert et al., 2015). The rule set covers 89% of the English rules, lacking only the handling of raising-to-subject verbs and more advanced strategies for coordination constructions and tense detection. To assign confidence scores, PropsDE uses a logistic regression model trained to predict the correctness of extractions. Figure 1 illustrates some extracted tuples. Based on correspondence with the authors of the English system, we conclude that we were able to implement the German version with roughly 10% of the effort they reported. This shows that our approach of manually porting a rule-based system can overcome the lack of a tool for another language with reasonable effort in a short amount of time.

5 Experiments

Experimental Setup Following the common evaluation protocol for Open IE systems, we manually label extractions made by our system. For this purpose, we created a new dataset consisting of 300 German sentences, randomly sampled from three sources of different genres: news articles from TIGER (Brants et al., 2004), German web pages from CommonCrawl (Habernal et al., 2016) and featured Wikipedia articles. For the treebank part, we ran our system using both gold and parsed dependencies to analyze the impact of parsing errors.

Every tuple extracted from this set of 300 sentences was labeled independently by two annotators as correct or incorrect. In line with previous work, they were instructed to label an extraction as incorrect if it has a wrong predicate or argument, including overspecified and incomplete arguments, or if it is well-formed but not entailed by the sentence. Unresolved co-references were not marked as incorrect. We observed an inter-annotator agreement of 85% ($\kappa = 0.63$). For the evaluation, we merged the labels, considering an extraction as correct only if both annotators labeled it as such. Results are measured in terms of precision, the fraction of correct extractions, and yield, the total number of extractions. A precision-yield curve is obtained by decreasing a confidence threshold. The confidence predictor was trained on a separate development set.

Results From the whole corpus of 300 sentences, PropsDE extracted 487 tuples, yielding on average 1.6 per sentence with 2.9 arguments. 60% of them were labeled as correct. Table 1 shows that most extractions are made from Wikipedia articles, whereas the highest precision can be observed for newswire text. According to our expectations, web pages are most challenging, presumably due to noisier language. These differences between the genres can also be seen in the precision-yield curve (Figure 2).

For English, state-of-the-art systems show a similar performance. In a direct comparison of several systems carried out by Del Corro and Gemulla (2013), they observed overall precisions of 58% (Reverb), 57% (ClausIE), 43% (WOE) and 43%

Genre	Sentences	Length	Yield	Precision
News*	100	19.3	142	78.9
News	100	19.3	144	70.8
Wiki	100	21.4	178	61.8
Web	100	19.2	165	49.1
Total	300	20.0	487	60.2

Table 1: Corpus size (length in token) and system performance by genre. News* used gold trees and is not included in total.

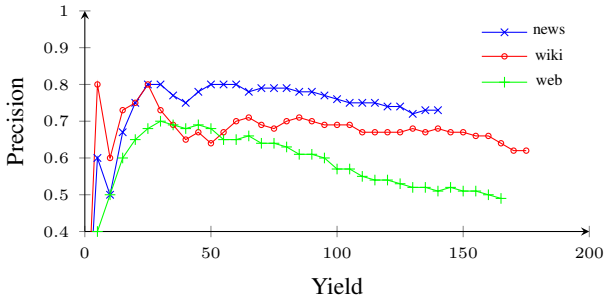


Figure 2: Extraction precision at increasing yield by genre.

(OLLIE) on datasets of similar genre. The reported yield per sentence is higher for ClausIE (4.2), OLLIE (2.6) and WOE (2.1), but smaller for Reverb (1.4). However, we note that in their evaluation, they configured all systems to output only two-argument-extractions. For example, from a sentence such as

> *The principal opposition parties boycotted the polls after accusations of vote-rigging.*

OLLIE can either make two binary extractions

> *boycotted (the principal opposition parties ; the polls)*
> *boycotted the polls after (the principal opposition parties ; accusations of vote-rigging)*

or just a single extraction with three arguments. PropS always extracts the combined tuple

> *boycotted (the principal opposition parties , the polls , after accusations of vote-rigging),*

which is in line with the default configuration of more recent Open IE systems.

For the sake of comparability, we conjecture that the yield of our system would increase if we broke down higher-arity tuples in a similar fashion: Assuming that every extraction with n arguments, $n > 2$, can be split into $n - 1$ separate extractions, our system's yield would increase from 1.6 to 3.0. That is in line with the numbers reported above for the binary configuration for English. Overall, this indicates a reasonable performance of our straightforward porting of PropS to German.

Extractions were most frequently labeled as incorrect due to false relation labels (32%), overspecified arguments (21%) and wrong word order in arguments (19%). Analyzing our system's performance on the treebank, we can see that the usage of gold dependencies increases the precision by 8 percentage points, making parsing errors responsible for about 28% of the incorrect extractions. Since the mate-tools parser is trained on the full TIGER treebank, including our experimental data, its error contribution on unseen data might be even higher.

6 Conclusion

Using PropS and German as examples, we showed that a rule-based Open IE system for English can be ported to another language in a reasonable amount of time. As a result, we presented the first Open IE system for German. In the future, studies targeting less similar languages could further evaluate the portability of PropS. Directions for future work on PropsDE are extensions of the rule set to better cover complex coordination constructions, nested sentences and nominal predicates.

Acknowledgments

This work has been supported by the DFG-funded research training group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES, GRK 1994/1), by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1) and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806.

References

Alan Akbik and Alexander Löser. 2012. KrakeN: N-ary Facts in Open Information Extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction & Web-scale Knowledge Extraction*, pages 52–56, Montreal, Canada.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 344–354, Beijing, China.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, Hyderabad, India.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint Morphological and Syntactic Analysis for Richly Inflected Languages. *Transactions of the Association for Computational Linguistics*, 1(0):415–428.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation*, 2(4):597–620.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards Coherent Multi-Document Summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, GA, USA.

Grzegorz Chrupała and Josef van Genabith. 2007. Using Very Large Corpora to Detect Raising and Control Verbs. In *Proceedings of the Lexical Functional Grammar 2007 Conference*, pages 597–620, Stanford, CA, USA.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1–8, Manchester, United Kingdom.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-Based Open Information Extraction. In *Proceedings of the 22nd International Conference on the World Wide Web*, pages 355–366, Rio de Janeiro, Brazil.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, United Kingdom.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1156–1165, New York, NY, USA.

Pablo Gamallo and Marcos Garcia. 2015. Multilingual Open Information Extraction. In *Proceedings of the 17th Portuguese Conference on Artificial Intelligence*, volume 9273 of *Lecture Notes in Computer Science*, pages 711–722, Coimbra, Portugal.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual Web-size corpus with free license. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 914–922, Portorož, Slovenia.

Sandra Kübler. 2008. The PaGe 2008 shared task on parsing German. In *Proceedings of the ACL-08: HLT Workshop on Parsing German (PaGe-08)*, pages 55–63, Columbus, OH, USA.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and Efficiency of Open Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Seattle, WA, USA.

Véronique Moriceau and Xavier Tannier. 2014. French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3239–3243, Reykjavik, Iceland.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 1659–1666, Portorož, Slovenia.

Eugen Ruppert, Jonas Klesy, Martin Riedl, and Chris Biemann. 2015. Rule-based Dependency Parse Collapsing and Propagation for German and English. In *Proceedings of the GSCL 2015*, pages 58–66, Duisburg, Germany.

Roland Schäfer. 2015. *Einführung in die grammatische Beschreibung des Deutschen*. Language Science Press, Berlin, Germany.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a Ger-

man Treebank. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an Intermediate Structure for Semantic Tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 303–308, Beijing, China.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. *Getting More Out Of Syntax with PropS*. arXiv:1603.01648.

Jannik Strötgen and Michael Gertz. 2015. A Baseline Temporal Tagger for all Languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547, Lisbon, Portugal.

Rolf Thieroff. 2004. The subjunctive mood in German and in the Germanic languages. In *Focus on Germanic Topology*, pages 315–358. Akademie Verlag, Berlin, Germany.

Fei Wu and Daniel S. Weld. 2010. Open Information Extraction Using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden.

Alisa Zhila and Alexander Gelbukh. 2013. Comparison of open information extraction for English and Spanish. In *Proceedings of the International Conference on Computational Linguistics and Intellectual Technologies (Dialogue 2013)*, pages 714–722, Bekasovo, Russia.

Named Entity Recognition for Novel Types by Transfer Learning

Lizhen Qu[1,2], **Gabriela Ferraro**[1,2], **Liyuan Zhou**[1],
Weiwei Hou[1], **Timothy Baldwin**[1,3]

[1] DATA61, Australia
[2] The Australian National University
[3] The University of Melbourne
{lizhen.qu,gabriela.ferraro,joe.zhou}@data61.csiro.au
houvivid2013@gmail.com, tb@ldwin.net

Abstract

In named entity recognition, we often don't have a large in-domain training corpus or a knowledge base with adequate coverage to train a model directly. In this paper, we propose a method where, given training data in a related domain with similar (but not identical) named entity (NE) types and a small amount of in-domain training data, we use transfer learning to learn a domain-specific NE model. That is, the novelty in the task setup is that we assume not just domain mismatch, but also label mismatch.

1 Introduction

There are two main approaches to named entity recognition (NER): (i) build sequence labelling models such as conditional random fields (CRFs) (Lafferty et al., 2001) on a large manually-labelled training corpus (Finkel et al., 2005); and (ii) exploit knowledge bases to recognise mentions of entities in text (Rizzo and Troncy, 2012; Mendes et al., 2011). For many social media-based or security-related applications, however, we cannot assume that we will have access to either of these. An alternative is to have a small amount of in-domain training data and access to large-scale annotated data in a second domain, and perform transfer learning over both the features and label set. This is the problem setting in this paper.

NER of novel named entity (NE) types poses two key challenges. First is the issue of sourcing labelled training data. Handcrafted features play a key role in supervised NER models (Turian et al., 2010), but if we have only limited training amounts of training data, we will be hampered in our ability to reliably learn feature weights. Second, the absence of target NE types in the source domain makes transfer difficult, as we cannot directly apply a model trained over the source domain to the target domain. Alvarado et al. (2015) show that even if the NE label set is identical across domains, large discrepancies in the label distribution can lead to poor performance.

Despite these difficulties, it is possible to transfer knowledge between domains, as related NE types often share lexical and context features. For example, the expressions *give lectures* and *attend tutorials* often occur near mentions of NE types PROFESSOR and STUDENT. If only PROFESSOR is observed in the source domain but we can infer that the two classes are similar, we can leverage the training data to learn an NER model for STUDENT. In practice, differences between NE classes are often more subtle than this, but if we can infer, for example, that the novel NE type STUDENT aligns with NE types PERSON and UNIVERSITY, we can compose the context features of PERSON and UNIVERSITY to induce a model for STUDENT.

In this paper, we propose a transfer learning-based approach to NER in novel domains with label mismatch over a source domain. We first train an NER model on a large source domain training corpus, and then learn the correlation between the source and target NE types. In the last step, we reuse the model parameters of the second step to initialise a linear-chain CRF and fine tune it to learn domain-specific patterns. We show that our methods achieve up to 160% improvement in F-score over a strong baseline, based on only 125 target-domain training sentences.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 899–905,
Austin, Texas, November 1-5, 2016.

2 Related work

The main scenario where transfer learning has been applied to NER is domain adaptation (Arnold et al., 2008; Maynard et al., 2001; Chiticariu et al., 2010), where it is assumed that the label set Y is the same for both the source and target corpora, and only the domain varies. In our case, however, both the domain and the label set differ across datasets.

Similar to our work, Kim et al. (2015) use transfer learning to deal with NER data sets with different label distributions. They use canonical correlation analysis (CCA) to induce label representations, and reduce the problem to one of domain adaptation. This supports two different label mappings: (i) to a coarse label set by clustering vector representations of the NE types, which are combined with mention-level predictions over the target domain to train a target domain model; and (ii) between labels based on the k nearest neighbours of each label type, and from this transferring a pre-trained model from the source to the target domain. They showed their automatic label mapping strategies attain better results than a manual mapping, with the pre-training approach achieving the best results. Similar conclusions were reached by Yosinski et al. (2014), who investigated the transferability of features from a deep neural network trained over the ImageNet data set. Sutton and McCallum (2005) investigated how the target task affects the source task, and demonstrated that decoding for transfer is better than no transfer, and joint decoding is better than cascading decoding.

Another way of dealing with a lack of annotated NER data is to use distant supervision by exploiting knowledge bases to recognise mentions of entities (Ling and Weld, 2012; Dong et al., 2015; Yosef et al., 2013; Althobaiti et al., 2015; Yaghoobzadeh and Schütze, 2015). Having a fine-grained entity typology has been shown to improve other tasks such as relation extraction (Ling and Weld, 2012) and question answering (Lee et al., 2007). Nevertheless, for many social media-based or security-related applications, we don't have access to a high-coverage knowledge base, meaning distant supervision is not appropriate.

3 Transfer Learning for NER

Our proposed approach **TransInit** consists of three steps: (1) we train a linear-chain CRF on a large source-domain corpus; (2) we learn the correlation between source NE types and target NE types using a two-layer neural network; and (3) we leverage the neural network to train a CRF for target NE types.

Given a word sequence $\mathbf{x}$ of length L, an NER system assigns each word x_i a label $y_i \in \mathcal{Y}$, where the label space $\mathcal{Y}$ includes all observed NE types and a special category O for words without any NE type. Let $(\mathbf{x}, \mathbf{y})$ be a sequence of words and their labels. A linear-chain CRF takes the form:

$$\frac{1}{Z}\prod_{l=1}^{L}\exp\Big(\mathbf{W}^f f(y_l, \mathbf{x}) + \mathbf{W}^g g(y_{l-1}, y_l)\Big), \quad (1)$$

where $f(y_l, \mathbf{x})$ is a feature function depending only on $\mathbf{x}$, and the feature function $g(y_{l-1}, y_l)$ captures co-occurrence between adjunct labels. The feature functions are weighted by model parameters $\mathbf{W}$, and Z serves as the partition function for normalisation.

The source domain model is a linear-chain CRF trained on a labelled source corpus. The co-occurrence of target domain labels is easy to learn due to the small number of parameters ($|Y|^2$). Mostly such information is domain specific so that it is unlikely that the co-occurrence of two source types can be matched to the co-occurrence of the two target types. However the feature functions $f(y_l, \mathbf{x})$ capture valuable information about the textual patterns associated with each source NE type. Without $g(y_{l-1}, y_l)$, the linear-chain CRF is reduced to a logistic regression (LR) model:

$$\sigma(y^*, \mathbf{x}_i; \mathbf{W}^f) = \frac{\exp(\mathbf{W}^f_{.y^*} f(y_i^*, \mathbf{x_i}))}{\sum_{y\in\mathcal{Y}} \exp(\mathbf{W}^f_{.y} f(y, \mathbf{x}_i))}. \quad (2)$$

In order to learn the correlation between source and target types, we formulate it as a predictive task by using the unnormalised probability of source types to predict the target types. Due to the simplification discussed above, we are able to extract a linear layer from the source domain, which takes the form $\mathbf{a}_i = \mathbf{W^s}\mathbf{x}_i$, where $\mathbf{W^s}$ denotes the parameters of $f(y_l, \mathbf{x})$ in the source domain model, and each $\mathbf{a}_i$ is the unnormalised probability for each source NE type. Taking $\mathbf{a}_i$ as input, we employ a multi-class LR classifier to predict target types, which is essentially $p(y'|\mathbf{a}) = \sigma(y', \mathbf{a}_i; \mathbf{W^t})$, where y' is the observed type. From another point of view, the whole architecture is a neural network with two linear layers.

We do not add any non-linear layers between these two linear layers because we otherwise end up with saturated activation functions. An activation function is saturated if its input values are its max/min values (Glorot and Bengio, 2010). Taking $\tanh(x)$ as an example, $\frac{\partial tanh(z)}{\partial z} = 1 - \tanh^2(z)$. If z is, for example, larger than 2, the corresponding derivative is smaller than 0.08. Assume that we have a three-layer neural network where z^i denotes the input of layer i, $\tanh(z)$ is the middle layer, and $L(z^{i-2})$ is the loss function. We then have $\frac{\partial L(z^{i-2})}{\partial z^{i-2}} = \frac{\partial L}{\partial z^{i+1}} \frac{\partial \tanh(z^{i-1})}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial z^{i-2}}$. If the $\tanh$ layer is saturated, the gradient propagated to the layers below will be small, and no learning based on back propagation will occur.

If no parameter update is required for the bottom linear layer, we will also not run into the issue of saturated activation functions. However, in our experiments, we find that parameter update is necessary for the bottom linear layer because of covariate shift (Sugiyama et al., 2007), which is caused by discrepancy in the distribution between the source and target domains. If the feature distribution differs between domains, updating parameters is a straightforward approach to adapt the model for new domains.

Although the two-layer neural network is capable of recognising target NE types, it has still two drawbacks. First, unlike a CRF, it doesn't include a label transition matrix. Second, the two-layer neural network has limited capacity if the domain discrepancy is large. If we rewrite the two-layer architecture in a compact way, we obtain:

$$p(y'|\mathbf{x}) = \sigma(y', \mathbf{x}_i; \mathbf{W^t W^s}). \quad (3)$$

As the equation suggests, if we minimize the negative log likelihood, the loss function is not convex. Thus, we could land in a non-optimal local minimum using online learning. The pre-trained parameter matrix $\mathbf{W^s}$ imposes a special constraint that the computed scores for each target type are a weighted combination of updated source type scores. If a target type shares nothing in common with source types, the pre-trained $\mathbf{W^s}$ does more harm than good.

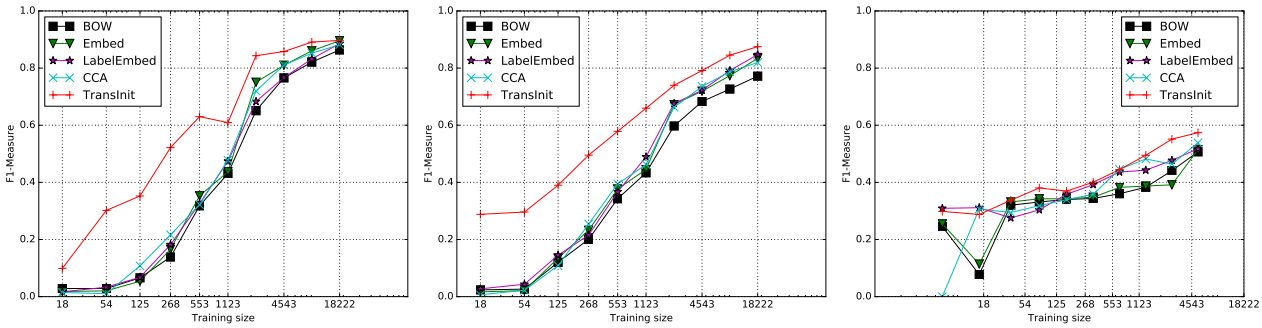
In the last step, we initialise the model parameters of a linear-chain CRF for $f(y_l, \mathbf{x})$ using the model parameters from the previous step. Based on the architecture of the NN model, we can collapse the two linear transformations into one by:

$$\mathbf{W}^f = \mathbf{W^t W^s}, \quad (4)$$

while initialising the other parameters of the CRF to zero. After this transformation, each initialised parameter vector $\mathbf{W}^f_{:y}$ is a weighted linear combination of the updated parameter vectors of the source types. Compared to the second step, the loss function we have now is convex because it is exactly a linear-chain CRF. Our previous steps have provided guided initialization of the parameters by incorporating source domain knowledge. The model also has significantly more freedom to adapt itself to the target types. In other words, collapsing the two matrices simplifies the learning task and removes the constraints imposed by the pre-trained $\mathbf{W^s}$.

Because the tokens of the class O are generally several orders of magnitude more frequent than the tokens of the NE types, and also because of covariate shift, we found that the predictions of the NN models are biased towards the class O (i.e. a non-NE). As a result, the parameters of each NE type will always include or be dominated by the parameters of O after initialisation. To ameliorate this effect, we renormalise $\mathbf{W}^t$ before applying the transformation, as in Equation (4). We do not include the parameters of the source class O when we initialise parameters of the NE types, while copying the parameters of the source class O to the target class O. In particular, let o be the index of source domain class O. For each parameter vector $\mathbf{W}^t_{i*}$ of NE type, we set $W^t_{io} = 0$. For the parameter vector for the target class O, we set only the element corresponding to the weight between source type O and target class O to 1, and other elements to 0.

Finally, we fine-tune the model over the target domain by maximising log likelihood. The training objective is convex, and thus the local optimum is also the global optimum. If we fully train the model, we will achieve the same model as if we trained from scratch over only the target domain. As the knowledge of the source domain is hidden in the initial weights, we want to keep the initial weights as long as they contribute to the predictive task. Therefore, we apply AdaGrad (Rizzo and Troncy, 2012) with early stopping based on development data, so that the knowledge of the source domain is preserved as much as possible.



(a) Target: I2B2, Source: BBN (b) Target: I2B2, Source: CoNLL (c) Target: CADEC, Source: CoNLL

Figure 1: Macro-averaged F1 results across all novel classes on different source/target domain combinations

4 Experimental Setup

4.1 Datasets

We use CADEC (Karimi et al., 2015) and I2B2 (Ben Abacha and Zweigenbaum, 2011) as target corpora with the standard training and test splits. From each training set, we hold out 10% as the development set. As source corpora, we adopt CoNLL (Tjong Kim Sang and De Meulder, 2003) and BBN (Weischedel and Brunstein, 2005).

In order to test the impact of the target domain training data size on results, we split the training set of CADEC and I2B2 into 10 partitions based on a log scale, and created 10 successively larger training sets by merging these partitions from smallest to largest (with the final merge resulting in the full training set). For all methods, we report the macro-averaged F1 over only the NE classes that are novel to the target domain.

4.2 Baselines

We compare our methods with the following two in-domain baselines, one cross-domain data-based method, and three cross-domain transfer-based benchmark methods.

BOW: an in-domain linear-chain CRF with hand-crafted features, from Qu et al. (2015).

Embed: an in-domain linear-chain CRF with hand-crafted features and pre-trained word embeddings, from Qu et al. (2015).

LabelEmbed: take the labels in the source and target domains, and determine the alignment based on the similarity between the pre-trained embeddings for each label.

CCA: the method of Kim et al. (2015), where a one-to-one mapping is generated between source and target NE classes using CCA and k-NN (see Section 2).

TransDeepCRF: A three-layer deep CRF. The bottom layer is a linear layer initialised with $\mathbf{W}^s$ from the source domain-trained CRF. The middle layer is a hard tanh function (Collobert et al., 2011). The top layer is a linear-chain CRF with all parameters initialised to zero.

TwoLayerCRF: A two-layer CRF. The bottom layer is a linear layer initialised with $\mathbf{W}^s$ from the source domain-trained CRF. The top layer is a linear-chain CRF with all parameters initialised to zero.

We compare our method with one variation, which is to freeze the parameters of the bottom linear layer and update only the parameters of the LR classifier while learning the correlation between the source and target types.

4.3 Experimental Results

Figure 1 shows the macro-averaged F1 of novel types between our method **TransInit** and the three baselines on all target corpora. The evaluation results on CADEC with BBN as the source corpus are not reported here because BBN contains all types of CADEC. From the figure we can see that **TransInit** outperforms all other methods with a wide margin on I2B2. When CoNLL is taken as the source corpus, despite not sharing any NE types with I2B2, several target types are subclasses of source types: DOCTOR and PATIENT w.r.t. PERSON, and HOS-

PITAL w.r.t. ORGANIZATION.

In order to verify if **TransInit** is able to capture semantic relatedness between source and target NE types, we inspected the parameter matrix $\mathbf{W}^t$ of the LR classifier in the step of learning type correlations. The corresponding elements in $\mathbf{W}^t$ indeed receive much higher values than the semantically-unrelated NE type pairs. When less than 300 target training sentences are used, these automatically discovered positive correlations directly lead to 10 times higher F1 scores for these types than the baseline **Embed**, which does not have a transfer learning step. Since **TransInit** is able to transfer the knowledge of multiple source types to related target types, this advantage leads to more than 10% improvement in terms of F1 score on these types compared with **LabelEmbed**, given merely 268 training sentences in I2B2. We also observe that, in case of few target training examples, **LabelEmbed** is more robust than **CCA** if the correlation of types can be inferred from their names.

We study the effects of transferring a large number of source types to target types by using BBN, which has 64 types. Here, the novel types of I2B2 w.r.t. BBN are DOCTOR, PATIENT, HOSPITAL, PHONE, and ID. For these types, **TransInit** successfully recognises PERSON as the most related type to DOCTOR, as well as CARDINAL as the most related type to ID. In contrast, **CCA** often fails to identify meaningful type alignments, especially for small training data sizes.

CADEC is definitely the most challenging task when trained on CoNLL, because there is no semantic connection between two of the target NE types (DRUG and DISEASE) and any of the source NE types. In this case, the baseline **LabelEmbed** achieves competitive results with **TransInit**. This suggests that the class names reflect semantic correlations between source and target types, and there are not many shared textual patterns between any pair of source and target NE types in the respective datasets.

Even with a complex model such as a neural network, the transfer of knowledge from the source types to the target types is not an easy task. Figure 2 shows that with a three-layer neural network, the whole model performs poorly. This is due to the fact that the hard tanh layer suffers from saturated function values. We inspected the values of the output hidden units computed by $\mathbf{W}^s\mathbf{x}$ on a random sample of target training examples before training on the target corpora. Most values are either highly positive or negative, which is challenging for online learning algorithms. This is due to the fact that these hidden units are unnormalised probabilities produced by the source domain classifier. Therefore, removing the hidden non-linear-layer layer leads to a dramatic performance improvement. Moreover, Figure 2 also shows that further performance improvement is achieved by reducing the two-layer architecture into a linear chain CRF. And updating the hidden layers leads to up to 27% higher F1 scores than not updating them in the second step of **TransInit**, which indicates that the neural networks need to update lower-level features to overcome the covariate shift problem.

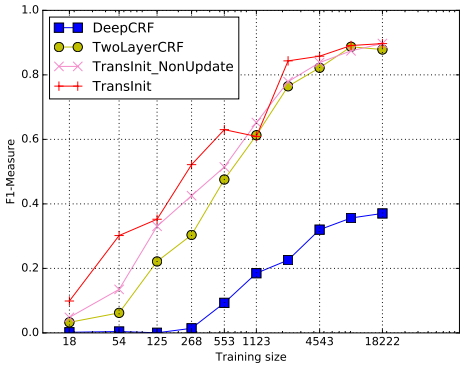


Figure 2: Difficulty of Transfer. The source model is trained on BBN.

5 Conclusion

We have proposed **TransInit**, a transfer learning-based method that supports the training of NER models across datasets where there are mismatches in domain and also possibly the label set. Our method was shown to achieve up to 160% improvement in F1 over competitive baselines, based on a handful of in-domain training instances.

Acknowledgments

This research was supported by NICTA, funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Maha Althobaiti, Udo Kruschwitz, and Massimo Poesio. 2015. Combining minimally-supervised methods for arabic named entity recognition. *Transactions of the Association for Computational Linguistics*, 3:243–255.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In *Australasian Language Technology Association Workshop 2015*.

Andrew Arnold, Ramesh Nallapati, and W. William Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In *Proceedings of ACL-08: HLT*, pages 245–253.

Asma Ben Abacha and Pierre Zweigenbaum. 2011. Medical entity recognition: A comparison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, pages 56–64.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1002–1012.

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

Li Dong, Furu Wei, Hong Tan, Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1243–1249.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370.

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

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of Biomedical Informatics*, 55:73–81.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015. New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, (ACL 2015)*, pages 473–482.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Changki Lee, Yi-Gyu Hwang, and Myung-Gil Jang. 2007. Fine-grained named entity recognition and relation extraction for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 799–800.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Diana Maynard, Valentin Tablan, Cristian Ursu, Hamish Cunningham, and Yorick Wilks. 2001. Named entity recognition from diverse text types. In *Recent Advances in Natural Language Processing 2001 Conference*.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big data small data, in domain out-of domain, known word unknown word: The impact of word representations on sequence labelling tasks. In *Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015)*, pages 83–93.

Giuseppe Rizzo and Raphaël Troncy. 2012. NERD: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. 2007. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005.

Charles Sutton and Andrew McCallum. 2005. Composition of conditional random fields for transfer learning. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 748–754.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Linguistic Data Consortium.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 715–725.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2013. HYENA-live: Fine-grained online entity type classification from natural-language text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 133–138.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, pages 3320–3328.

Extracting Subevents via an Effective Two-phase Approach

Allison Badgett and **Ruihong Huang**
Texas A&M University
{allisonbadgett, huangrh}@cse.tamu.edu

Abstract

We present our pilot research on automatically extracting subevents from a domain-specific corpus, focusing on the type of subevents that describe physical actions composing an event. We decompose the challenging problem and propose a two-phase approach that effectively captures sentential and local cues that describe subevents. We extracted a rich set of over 600 novel subevent phrases. Evaluation shows the automatically learned subevents help to discover 10% additional main events (of which the learned subevents are a part) and improve event detection performance.

1 Introduction

General and abstract event expressions and event keywords are often used for detecting events. To detect civil unrest events for example, common event expressions "took to the streets" and "staged a protest", and event keywords "rally" and "strike" are usually considered as the first option. Subevents, events that occur as a part of the main event and therefore are useful to instantiate the main event, widely exist in event descriptions but they are rarely used for detecting the main event.

In this paper, we focus on learning subevent phrases that describe physical actions composing an event. Such subevents are the evidence that an event is occurring. For example, if a person were to explain how he knew that the crowd gathered in the street was rioting, he might point to the shouting of political slogans or tires lit on fire. In this instance, the riot would be the event. The slogan-shouting and tire-burning would be the subevents. Because subevent detection requires an understanding of levels of abstraction, this task can even be difficult for humans to perform. Furthermore, subevent phrases and general event phrases often have the same grammatical structure and may share common words, which makes automatically differentiating between event phrases and subevent phrases within a document even more difficult.

Additionally, events and subevents are not disjoint classes. There are some subevents that are unambiguous. For example, "burning tires" is a concrete phrase that would not fall into the category of more abstract events. However, "gathered at site" is certainly more ambiguous. Even human analysts would not necessarily agree on the appropriate class for this phrase. In many cases, the categorization would be context-dependent. Because of this, our research focused on identifying the less ambiguous, and therefore more concrete, cases.

Instead of separating subevent phrases from general event phrases, we explicitly acquire subevent phrases by leveraging both sentential and local cues in describing subevents. We observed that subevents of the type in which we are interested, as important facts presented in news stories, are commonly mentioned in sentences that refer to the source of information or simply in quotation sentences. These sentences usually start or end with characteristic phrases such as "media reports" and "witness said". Furthermore, we observed that subevent phrases often occur in conjunction constructions as a sequence of subevent phrases, as shown in the following examples:

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 906–911,
Austin, Texas, November 1-5, 2016.

(1) *State television broadcast the event live, offering sweeping aerial views that showed the sea of people* **waving banners, blew whistles, and shouted slogans**.

(2) *They also* **set fires, stoned civilian vehicles, taunted the police and hurled stones** *at them, witnesses said.*

where the subevents are shown in bold, and sentential cues are underlined.

Inspired by these observations, we propose a novel two-phase approach to automatically extract subevents, which consists of a sentence classifier that incrementally identifies sentences mentioning subevents and a subevent extractor which looks for a sequence of subevent phrases in a conjunction structure. Our sentence classifier is trained in a weakly supervised manner and only requires a small set of subevent phrases as a guide. The classifier was initially trained with sentences containing eight provided subevent seeds, then it proceeded to label more sentences that mention new subevent phrases.

This two-phase subevent extraction approach can successfully identify 610 diverse subevent phrases from a domain-specific corpus. We evaluate our automatically learned subevent phrases by using them to detect events. Experimental results show that the learned subevent phrases can recover an additional 10% of event articles and improve event detection F-1 score by 3%.

2 Related Work

While it is generally agreed that subevents are an important type of information in event descriptions, they are seldom considered in decades of event extraction research (Appelt et al., 1993; Riloff, 1993; Soderland et al., 1995; Sudo et al., 2003; Li et al., 2005; Yu et al., 2005; Gu and Cercone, 2006; Maslennikov and Chua, 2007; S. and E., 2009; Liao and Grishman, 2010; Huang and Riloff, 2011; Chambers and Jurafsky, 2011; Huang and Riloff, 2012; Huang et al., 2016). Subevents as a theme has been discussed in the past three Event workshops (Eve, 2013), (Eve, 2014), (Eve, 2015). However, despite the great potential of using subevents to improve event detection and extraction (Hakeem and Shah, 2005), and event coreference resolution (Araki et al., 2014), there is little existing research on automatically learning subevent phrases, partially because researchers have not agreed upon the definition of subevents. Much recent research in event timeline generation (Huang and Huang, 2013) suggests the usefulness of subevents in improving quality and completeness of automatically generated event summaries. However, they often focus on a different notion of subevents that broadly covers pre-condition events and consequence events and is temporally-based.

Subevents have been studied for event tracking applications (Shen et al., 2013; Meladianos et al., 2015). However, most current research is specifically related to social media applications, like Twitter, in terms of both its definition of subevents and methodologies. For example, in previous research by (Shen et al., 2013), a subevent is defined as a topic that is discussed intensively in the Twitter stream for a short period of time before fading away. Accordingly, the subevent detection method relies on modeling the "burstiness" and "cohesiveness" properties of tweets in the stream. We instead aim to provide a more general definition of subevents as well as present a method for identifying subevent at the article level.

3 A Two-phase Approach for Subevent Extraction

As illustrated in Figure 1, We use a two-phase algorithm to identify subevent phrases from our domain-specific corpus. For the first stage, we implemented a bootstrapped artificial neural network in order to identify sentences that are likely to contain a subevent phrase. In the second stage, we identify phrases fitting a predetermined conjunction pattern within the sentences classified by the first-stage neural network.

3.1 Phase 1: Identifying Subevent Sentences

3.1.1 Domain-specific Corpus

Thanks to previous research on multi-faceted event recognition by (Huang and Riloff, 2013), we compiled our own domain-specific corpus that describes civil unrest events. Using civil unrest events as an example, (Huang and Riloff, 2013) demon-

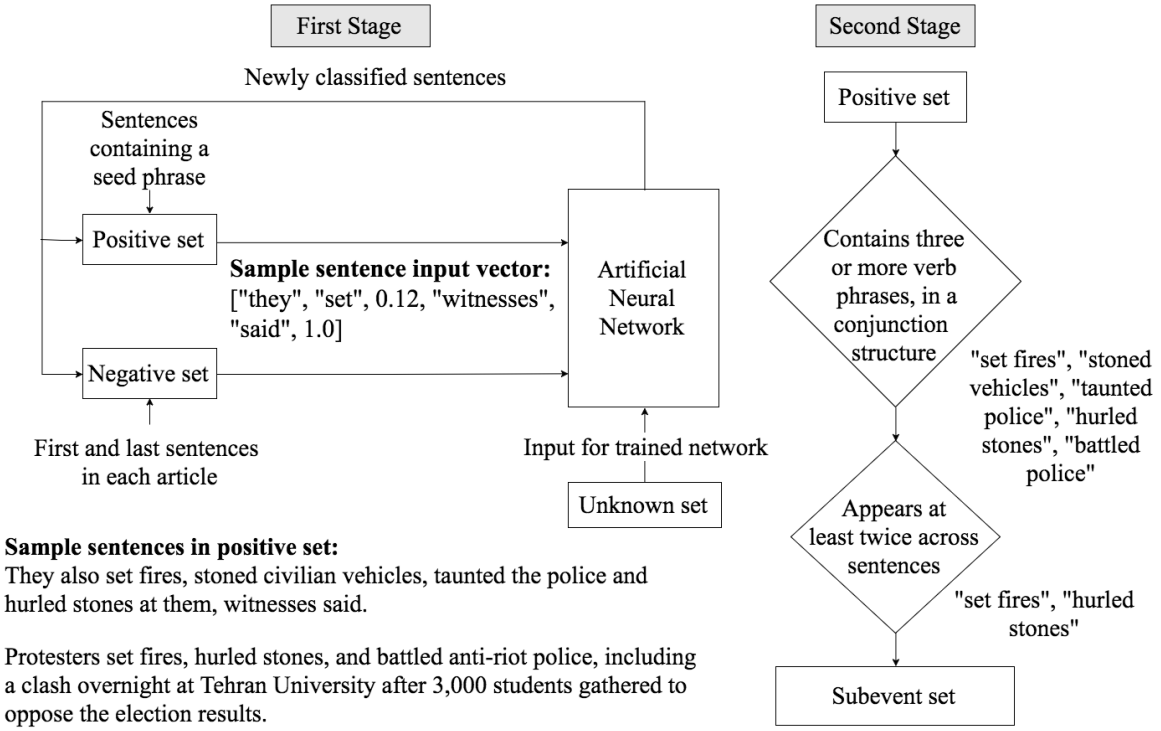


Figure 1: The Two-step Subevent Learning Paradigm

strated that we can use automatically-learned event facet phrases (event agents and purposes) and main event expressions to find event articles with a high accuracy. We first obtained their learned event facet phrases and event expressions, most of which refer to general events. Then we followed their paper and identified two types of news articles that are likely to describe a civil unrest event by matching the obtained phrases to the English Gigaword fifth edition (Parker et al., 2011)

Specifically, we first consider news articles that contain a civil unrest keyword such as "strike" and "protest"[1], then we identify an article as relevant if it contains a sentence where either two types of facet phrases or one facet phrase together with an event expression are found. In addition, we consider news articles that do not contain a civil unrest keyword; we require an article to contain a sentence where three types of event information are matched. Overall, we get a civil unrest corpus containing 232,710 news articles.

[1]We used the same set of keywords as used by (Huang and Riloff, 2013)

3.1.2 Context Feature Selection

We hypothesized that the first and last noun/verb phrases and their positions in the sentence were likely to be good indicators that the sentence might contain a subevent phrase. Because our document corpus was composed of news articles, we determined that concrete subevents would require a level of substantiation that abstract, non-specific events would not. For example, a reporter would not usually cite a source in a sentence informing the reader that a riot occurred but would likely choose to quote a source when reporting that rioters burned tires in the streets. Because of this, sentences containing subevent phrases often begin or end with phrases such as "he witnessed" or "she told the press." To represent the nouns and verbs, we used the 50-dimention Stanford GloVe (Pennington et al., 2014) word embeddings pre-trained on Wikipedia 2014 and Gigaword5 (Parker et al., 2011).

3.1.3 Seeds and Training Sentence Generation

To form a training set for the neural network, we used eight seed subevent phrases (as shown in Table 1) to identify a set of positive sentences that con-

waved banners
shouted slogans
chanted slogans
burned flag
burned flags
blocked road
clashed with police
clashed with government

Table 1: First Stage Classifier Seed Subevent Phrases

tain one of the seed phrases. In total, we obtained around 5000 positive sentences and bounded this to 3500 for use with the classifier. Finding a sufficient number of negative sentences was a more challenging task. After reviewing the corpus, we determined that the first and last sentences of an article are unlikely to contain subevent phrases. These sentences often function as general introductions and conclusions that refer to the main event of the article. We selected 7000 of these sentences to form the negative set. The rest of the sentences not classified as positive or negative remain unknown, amounting to almost 1 million.

3.1.4 Artificial Neural Networks for Sentence Classification

We implemented an artificial neural network with a single hidden layer composed of 500 nodes. In order to facilitate faster training, we used tanh as the activation function of the hidden layer. Softmax was used as the output layer activation function. In order to train the network, we provided an initial set of positive and negative vectors representing sentence data from the corpus as described in Section 3.1.3. These input vectors were then divided into a training set, validation set and testing set. The training set was comprised of 70% of the full dataset, the validation set of 20% and the testing set of 10%. The neural network was trained for 1000 epochs and used the validation set and test set to measure performance in order to avoid overfitting.

Because we began with a limited number of seed phrases to create the positive set, we chose to use a bootstrapping approach to expand our data set and improve the classifier. After training, the entire unknown data set would be classified, and sentences determined to be positive with 0.90 certainty or greater would be added to the positive set. Sentences classified as negative with 0.90 certainty or greater would be added to the negative set. However, in order to maintain the 2:1 ratio of negative to positive vectors, the number of negative vectors that could be added was capped at twice the number of positive additions for each iteration. After the new sentences were added to the positive and negative sets, the neural network was retrained with this data and classified additional previously unknown sentences. The process repeated for five iterations, then bootstrapping ended because not enough newly identified positive sentences were found ($<$3000 in the last iteration). Table 2 shows the number of sentences that were added after each bootstrapping iteration.

Iteration	Positives	Negatives
1	13223	26446
2	12611	25222
3	9411	18822
4	6076	12152
5	2842	5684

Table 2: Number of Sentences Added after Each Iteration

3.2 Phase 2: Subevent Extraction

After accumulating a large set of sentences likely containing subevents from the first phase of the system, the second step identifies the subevent phrases within these sentences. We observe that subevent phrases often occur in lists and we focus on leveraging such local cues to extract subevents. Specifically, we identify conjunction constructions that contain three or more verb phrases, each verb phrase obeys one of the following two forms: verb + direct object or verb + prepositional phrase. We extract the sequence of verb phrases, each as a subevent candidate. We only included subevents with frequency greater than or equal to two in the final evaluation. Through the two-stage extraction procedure, we identified 610 unique subevents. Table 3.2 shows some of the learned subevents.

Clearly, this second phase suffers from low recall. However, because subevents are identified at the corpus level as opposed to the document level, per-sentence recall is not a significant concern as long as a sufficient number of subevents are identi-

threw stones, hurled rocks, pounded in air
smashed through roadblocks, detained people
forced way, fired in air, threw at police
smashed windows, set fire, burned tires
threw bombs, opened fire, blocked road
pelted with stones, appeared on balcony
arrested for vandalism, threw bombs
burned cars, carried banners, lit candles
detained people, planted flag, wore masks
stoned police, converged on highway
chanted against authorities, chanted in city
broke through barricade, blocked traffic
broke windows, screamed outside palace
torched cars, ransacked office, smashed shop
shouted in unison, sang songs, planted flags
runs alongside shrines, chanted for democracy

Table 3: Subset of learned subevents

	Recall	Precision	F1-score
(Huang and Riloff, 2013)	71	**88**	79
+Subevents	**81**	83	**82**

Table 4: Event Recognition Performance Before/After Incorporating Subevents

fied across the whole corpus. As we demonstrate in the evaluation section, corpus-level recall was high enough to produce noticeable results.

4 Evaluation

We show that our acquired subevent phrases are useful to discover articles that describe the main event and therefore improve event detection performance.

For direct comparisons, we tested our subevents using the same test data and the same evaluation setting as the previous multi-faceted event recognition research by (Huang and Riloff, 2013). Specifically, they have annotated 300 new articles that each contains a civil unrest keyword and only 101 of them are actually civil unrest stories. They have shown that the multi-faceted event recognition approach can accurately identify civil unrest documents, by identifying a sentence in the documents where two types of facet phrases or one facet phrase and a main event expression were matched. The first row of Table 4 shows their multi-faceted event recognition performance.

We compared our learned subevent phrases with the event phrases learned by (Huang and Riloff, 2013) and found that 559 out of our 610 unique phrases are not in their list. We augmented their provided event phrase list with our newly acquired subevent phrases and then used the exactly same evaluation procedure. Essentially, we used a longer event phrase dictionary which is a combination of main event expressions resulted from the previous research by (Huang and Riloff, 2013) and our learned subevent phrases. Row 2 shows the event recognition performance using the extended event phrase list. We can see that after incorporating subevent phrases, additional 10% of civil unrest stories were discovered, with a small precision loss, the F1-score on event detection was improved by 3%.

5 Conclusion

We have presented a two-phase approach for identifying a specific type of "subevents", referring to physical actions composing an event. While our approach is certainly tailored to the civil unrest domain, we believe that this method is applicable to many other domains within the scope of news reports, including health, economics and even politics, where reporters overwhelmingly rely on outside opinion to present the facts of the story and provide the summary themselves. However in more casual domains where this is not necessarily the case, this approach will suffer. For instance, in sports writing, a reporter giving a play-by-play of a basketball game will not need to call upon witnesses or field experts to present concrete subevents.

Furthermore, we have shown the great potential of using subevents to improve event detection performance. In addition, distinguishing between events and subevents develops an event hierarchy and can benefit multiple applications such as text summarization and event timeline generation.

Acknowledgments

We want to thank our anonymous reviewers for providing useful comments.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. FASTUS: a Finite-state Processor for Information Extraction from Real-world Text. In *Proceedings*

of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI).

Jun Araki, Zhengzhong Liu, Eduard H Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *LREC*, pages 4553–4558.

Nathanael Chambers and Dan Jurafsky. 2011. Template-Based Information Extraction without the Templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-11)*.

2013. The 1st Workshop on EVENTS: Definition, Detection, Coreference, and Representation. In *https://sites.google.com/site/cfpwsevents/home*.

2014. The 2nd Workshop on EVENTS: Definition, Detection, Coreference, and Representation. In *https://sites.google.com/site/wsevents2014/home*.

2015. The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation. In *https://sites.google.com/site/wsevents2015/home*.

Z. Gu and N. Cercone. 2006. Segment-Based Hidden Markov Models for Information Extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 481–488, Sydney, Australia, July.

Asaad Hakeem and Mubarak Shah. 2005. Multiple agent event detection and representation in videos. In *AAAI*, pages 89–94.

Lifu Huang and Lian'en Huang. 2013. Optimized event storyline generation based on mixture-event-aspect model. In *EMNLP*, pages 726–735.

Ruihong Huang and Ellen Riloff. 2011. Peeling Back the Layers: Detecting Event Role Fillers in Secondary Contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-11)*.

Ruihong Huang and Ellen Riloff. 2012. Modeling Textual Cohesion for Event Extraction. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12)*.

Ruihong Huang and Ellen Riloff. 2013. Multi-faceted Event Recognition with Bootstrapped Dictionaries. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-13)*.

L. Huang, T. Cassidy, X. Feng, H. Ji, C. Voss, J. Han, and A. Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-16)*.

Y. Li, K. Bontcheva, and H. Cunningham. 2005. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning*, pages 72–79, Ann Arbor, MI, June.

Shasha Liao and Ralph Grishman. 2010. Using Document Level Cross-Event Inference to Improve Event Extraction. In *Proceedings of the 48st Annual Meeting on Association for Computational Linguistics (ACL-10)*.

M. Maslennikov and T. Chua. 2007. A Multi-Resolution Framework for Information Extraction from Free Text. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. 2015. Degeneracy-based real-time sub-event detection in twitter stream. In *Proceedings of the 9th AAAI international conference on web and social media (ICWSM)*, pages 248–257.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword. In *Linguistic Data Consortium*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

E. Riloff. 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence*.

Patwardhan S. and Riloff E. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In *Proceedings of 2009 the Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*.

Chao Shen, Fei Liu, Fuliang Weng, and Tao Li. 2013. A participant-based approach for event summarization using twitter streams. In *HLT-NAACL*, pages 1152–1162.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a Conceptual Dictionary. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319.

K. Sudo, S. Sekine, and R. Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*.

K. Yu, G. Guan, and M. Zhou. 2005. Resumé Information Extraction with Cascaded Hybrid Model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 499–506, Ann Arbor, MI, June.

Gaussian Visual-Linguistic Embedding for Zero-Shot Recognition

Tanmoy Mukherjee and **Timothy Hospedales**
Queen Mary University of London
School of Electronic Engineering and Computer Science
{k.m.tanmoy,t.hospedales}@qmul.ac.uk

Abstract

An exciting outcome of research at the intersection of language and vision is that of zero-shot learning (ZSL). ZSL promises to scale visual recognition by borrowing distributed semantic models learned from linguistic corpora and turning them into visual recognition models. However the popular word-vector DSM embeddings are relatively impoverished in their expressivity as they model each word as a single vector point. In this paper we explore word-*distribution* embeddings for ZSL. We present a visual-linguistic mapping for ZSL in the case where words and visual categories are both represented by distributions. Experiments show improved results on ZSL benchmarks due to this better exploiting of intra-concept variability in each modality

1 Introduction

Learning vector representations of word meaning is a topical area in computational linguistics. Based on the distributional hypothesis (Harris, 1954) – that words in similar context have similar meanings – distributed semantic models (DSM)s build vector representations based on corpus-extracted context. DSM approaches such as topic models (Blei et al., 2003), and more recently neural networks (Collobert et al., 2011; Mikolov et al., 2013) have had great success in a variety of lexical and semantic tasks (Arora et al., 2015; Schwenk, 2007).

However despite their successes, classic DSMs are severely impoverished compared to humans due to learning solely from word cooccurrence without grounding in the outside world. This has motivated a wave of recent research into multi-modal and cross-modal learning that aims to *ground* DSMs in non-linguistic modalities (Bruni et al., 2014; Kiela and Bottou, 2014; Silberer and Lapata, 2014; **?**). Such multi-modal DSMs are attractive because they learn richer representations than language-only models (e.g., that bananas are *yellow* fruit (Bruni et al., 2012b)), and thus often outperform language only models in various lexical tasks (Bruni et al., 2012a).

In this paper, we focus on a key unique and practically valuable capability enabled by cross-modal DSMs: that of zero-shot learning (ZSL). Zero-shot recognition aims to recognise visual categories in the absence of any training examples by cross-modal transfer from language. The idea is to use a limited set of training data to learn a linguistic-visual mapping and then apply the induced function to map images from novel visual categories (unseen during training) to a linguistic embedding: thus enabling recognition in the absence of visual training examples. ZSL has generated big impact (Lampert et al., 2009; Socher et al., 2013; Lazaridou et al., 2014) due to the potential of leveraging language to help visual recognition scale to many categories without labor intensive image annotation.

DSMs typically generate *vector* embeddings of words, and hence ZSL is typically realised by variants of vector-valued cross-modal regression. However, such vector representations have limited expressivity – each word is represented by a point, with no notion of intra-class variability. In this paper, we consider ZSL in the case where both visual and linguistic concepts are represented by *Gaussian distribution* embeddings. Specifically, our Gaussian-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 912–918,
Austin, Texas, November 1-5, 2016.

embedding approach to ZSL learns concept distributions in both domains: Gaussians representing individual words (as in (Vilnis and McCallum, 2015)) and Gaussians representing visual concepts. Simultaneously, it learns a cross-domain mapping that warps language-domain Gaussian concept representations into alignment with visual-domain concept Gaussians. Some existing vector DSM-based cross-modal ZSL mappings (Akata et al., 2013; Frome et al., 2013) can be seen as special cases of ours where the within-domain model is pre-fixed as vector corresponding to the Gaussian means alone, and only the cross-domain mapping is learned. Our results show that modeling linguistic and visual concepts as Gaussian distributions rather than vectors can significantly improve zero-shot recognition results.

2 Methodology

2.1 Background

Vector Word Embeddings In a typical setup for unsupervised learning of word-vectors, we observe a sequence of tokens $\{w_i\}$ and their context words $\{c(w)_i\}$. The goal is to map each word w to a d-dimensional vector e_w reflecting its distributional properties. Popular skip-gram and CBOW models (Mikolov et al., 2013), learn a matrix $W \in \mathbb{R}^{|V| \times d}$ of word embeddings for each of V vocabulary words ($e_w = W_{(w,:)}$) based on the objective of predicting words given their contexts.

Another way to formalise a word vector representation learning problem is to search for a representation W so that words w have high representational similarity with co-occuring words $c(w)$, and low similarity with representations of non-co-occurring words $\neg c(w)$. This could be expressed as optimisation of max-margin loss J; requiring that each word w's representation e_w is more similar to that of context words e_p than non-context words e_n.

$$J(W) = \sum_{w, w_p \in c(w), w_n \in \neg c(w)} \max(0, \delta - E(e_w, e_{w_p}) + E(e_w, e_{w_n})) \quad (1)$$

where similarity measure $E(\cdot, \cdot)$ is a distance in $\mathbb{R}^d$ space such as cosine or euclidean.

Gaussian Word Embeddings Vector-space models are successful, but have limited expressivity in terms of modelling the variance of a concept, or asymmetric distances between words, etc. This has motivated recent work into *distribution*-based embeddings (Vilnis and McCallum, 2015). Rather than learning word-vectors e_w, the goal here is now to learn a distribution for each word, represented by a per-word mean μ_w and covariance Σ_w.

In order to extend word representation learning approaches such as Eq. (1) to learning Gaussians, we need to replace vector similarity measure $E(\cdot, \cdot)$ with a similarity measure for Gaussians. We follow (Vilnis and McCallum, 2015) in using the inner product between distributions f and g – the probability product kernel (Jebara et al., 2004).

$$E(f, g) = \int_{x \in \mathbb{R}^n} f(x) g(x). \quad (2)$$

The probability product kernel (PPK) has a convenient closed form in the case of Gaussians:

$$\begin{aligned} E(f, g) &= \int_{x \in \mathbb{R}^n} \mathcal{N}(x; \mu_f, \Sigma_f) \mathcal{N}(x; \mu_g, \Sigma_g) dx \\ &= \mathcal{N}(0; \mu_f - \mu_g, \Sigma_f + \Sigma_g) \end{aligned} \quad (3)$$

where μ_f, μ_g are the means and Σ_f, Σ_g are the covariances of the probability distribution f and g.

2.2 Cross-Modal Distribution Mapping

Gaussian models of words can be learned as in the previous section, and that Gaussian models of image categories can be trivially obtained by maximum likelihood. The central task is therefore to establish a mapping between word-and image-Gaussians, which will be of different dimensions d_w and d_x.

We aim to find a projection matrix $A \in \mathbb{R}^{d_x \times d_w}$ such that a word w generates an image vector as $e_x = A e_w$. Working with distributions, this implies that we have $\mu_x = A \mu_w$ and $\Sigma_x = A \Sigma_w A^T$. We can now evaluate the similarity of concept distributions across modalities. The similarity between image-and text-domain Gaussians f and g is:

$$E(f, g) = \mathcal{N}(0; \mu_f - A\mu_g, \Sigma_f + A \Sigma_g A^T) \quad (4)$$

Using this metric, we can train our cross-modal projection A via the cross-domain loss:

$$J(A) = \sum_{f, g \in P,\ h, k \in N} \max(0, \delta - E(f, g) + E(h, k)) \quad (5)$$

where P is the set of matching pairs that should be aligned (e.g., the word Gaussian 'plane' and the Gaussian of plane images) and N is the set of mismatching pairs that should be separated (e.g., 'plane' and images of dogs). This can be optimised with SGD using the gradient:

$$\begin{aligned}\frac{\partial E}{\partial A} &= \frac{1}{2}((\Sigma_f + A\Sigma_g A^T)^{-1}A(\Sigma_g + \Sigma_g^T)) \\ &+ ((\mu_g^T(\Sigma_f + A\Sigma_g A^T)^{-1}(\mu_f - A\mu_g) \\ &+ (\mu_f - A\mu_g)^T(\Sigma_f + A\Sigma_g A^T)^{-1}\mu_g^T \\ &+ (\mu_f - A\mu_g)^T(\Sigma_f + A\Sigma_g A^T)^{-1} \\ &A^T(\Sigma_g + \Sigma_j^T)(\Sigma_f + A\Sigma_g A^T)^{-1}(\mu_f - A\mu_g))\end{aligned}$$

2.3 Joint Representation and Mapping

The cross-domain mapping A can be learned (Eq. 5) for fixed within-domain representations (word and image Gaussians). It is also possible to simultaneously learn the text and image-domain gaussians ($\{\mu_i, \Sigma_i\}^{text}, \{\mu_j, \Sigma_j\}^{img}$) by optimising the sum of three coupled losses: Eq. 1 with Eq. 3, Eq. 5 and max-margin image-classification using Gaussians. We found jointly learning the image-classification Gaussians did not bring much benefit over the MLE Gaussians, so we only jointly learn the text Gaussians and cross-domain mapping.

2.4 Application to Zero-Shot Recognition

Once the text-domain Gaussians and cross-domain mapping have been trained for a set of known words/classes, we can use the learned model to recognise any novel/unseen but name-able visual category w as follows: 1. Get the word-Gaussians of target categories w, $\mathcal{N}(\mu_w, \Sigma_w)$. 2. Project those Gaussians to image modality, $\mathcal{N}(A\mu_w, A\Sigma_w A^T)$. 3. Classify a test image x by evaluating its likelihood under each Gaussian, and picking the most likely Gaussian: $p(w|x) \propto \mathcal{N}(x|A\mu_w, A\Sigma_w A^T)$.

2.5 Contextual Query

To illustrate our approach, we also experiment with a new variant of the ZSL setting. In conventional ZSL, a novel word can be matched against images by projecting it into image space, and sorting images by their distance to the word (vector), or likelihood under the word (Gaussian). However, results may be unreliable when used with polysemous words, or words with large appearance variability. In this case we may wish to enrich the query with contextual words that disambiguate the visual meaning of the query. With regular vector-based queries, the typical approach is to sum the word-vectors. For example: For contextual disambiguation of polysemy, we may hope that vec('bank')+vec('river') may retrieve a very different set of images than vec('bank')+vec('finance'). For specification of a specific subcategory or variant, we may hope that vec('plane')+vec('military') retrieves a different set of images than vec('plane')+vec('passenger'). By using distributions rather than vectors, our framework provides a richer means to make such queries that accounts for the intra-class variability of each concept. When each word is represented by a Gaussian, a two-word query can be represented by their product, which is the new Gaussian $\mathcal{N}(\frac{\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2}{\Sigma_1^{-1}+\Sigma_2^{-1}}, (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1})$.

3 Experiments

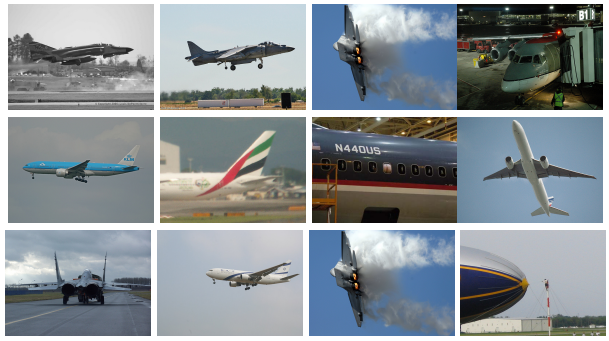
3.1 Datasets and Settings

Datasets: We evaluate our method [1] using the main Animals with Attributes (**AWA**) and **ImageNet1K** benchmarks. To extract visual features we use the VGG-16 CNN (Simonyan and Zisserman, 2015) to extract a $d_x = 4096$ dimensional feature for each image. To train the word Gaussian representation, we use a combination of UkWAC (Ferraresi et al., 2008) and Wikipedia corpus of 25 million tokens, and learn a $d_w = 100$ dimensional Gaussian representation. We set our margin parameter to $\Delta = 1$.

Settings: Our zero-shot setting involves training a visual recogniser (i.e., our mapping A) on a subset of classes, and evaluating it on a disjoint subset. For AWA, we use the standard 40/10 class split (Lampert et al., 2009), and for ImageNet we use a standard 800/200 class split (Mensink et al., 2012).

Competitors: We implement a set of representative alternatives for direct comparison with ours on the same visual features and text corpus. These include: cross-modal linear regression (LinReg, (Dinu et al., 2015)), non-linear regression (NLinReg, (Lazaridou et al., 2014; Socher et al., 2013)),

[1] Code and datasets kept at `http://bit.ly/2cI64Zf`



(a) Top: 'Military'+'Plane' (Gaussian), Middle: 'Passenger'+'Plane' (Gaussian), Bottom: 'Passenger'+'Plane' (Vector)

(b) Top: 'White'+'Horse' (Gaussian), Middle: 'Black'+'Horse' (Gaussian), Bottom: 'Black'+'Horse' (Vector)

Figure 1: Qualitative visualisation of zero-shot query with context words.

	Vector space models				Ours
Dataset	LinReg	NLinReg	CME	ES-ZSL	Gaussian
AWA	44.0	48.4	43.1	58.2	65.4

Table 1: Zero-shot recognition results on AWA (% accuracy).

ImageNet	
ConSE (Norouzi et al., 2014)	28.5%
DeVISE (Frome et al., 2013)	31.8%
Large Scale Metric. (Mensink et al., 2012)	35.7%
Semantic Manifold. (Fu et al., 2015)	41.0%
Gaussian Embedding	45.7%
AwA	
DAP (CNN feat) (Lampert et al., 2009)	53.2%
ALE (Akata et al., 2013)	43.5%
TMV-BLP (Fu et al., 2014)	47.1%
ES-ZSL (Romera-Paredes and Torr, 2015)	49.3%
Gaussian Embedding	65.4%

Table 2: Comparison of our ZSL results with state of the art.

ES-ZSL (Romera-Paredes and Torr, 2015), and a max-margin cross-modal energy function method (CME, (Akata et al., 2013; Frome et al., 2013)). Note that the CME strategy is the most closely related to ours in that it also trains a $d_x \times d_w$ matrix with max-margin loss, but uses it in a bilinear energy function with vectors $E(x, y) = x^T Ay$; while our energy function operates on Gaussians.

3.2 Results

Table 1 compares our results on the AWA benchmark against alternatives using the same visual features, and word vectors trained on the same corpus. We observe that: (i) Our Gaussian-embedding obtains the best performance overall. (ii) Our method outperforms CME which shares an objective function and optimisation strategy with ours, but operates on vectors rather than Gaussians. This suggests that our new distribution rather than vector-embedding does indeed bring significant benefit.

A comparison to published results obtained by other studies on the same ZSL splits is given in Table 2, where we see that our results are competitive despite exploitation of supervised embeddings such as attributes (Fu et al., 2014), or combinations of embeddings (Akata et al., 2013) by other methods.

We next demonstrate our approach qualitatively by means of the contextual query idea introduced in Sec 2.5. Fig. 1 shows examples of how the top retrieved images differ intuitively when querying ImageNet for zero-shot categories 'plane' and 'horse' with different context words. To ease interpretation, we constrain the retrieval to the true target class, and focus on the effect of the context word. Our learned Gaussian method retrieves more relevant images than the word-vector sum baseline. E.g., with the Gaussian model all of the top-4 retrieved images for Passenger+Plane are relevant, while only two are relevant with the vector model. Similarly, the retrieved black horses are more clearly black.

3.3 Further Analysis

To provide insight into our contribution, we repeat the analysis of the AwA dataset and evaluate several variants of our full method. These use our features, and train the same cross-domain max-margin loss in Eq 5, but vary in the energy function and representa-

AwA	
Bilinear-WordVec	43.1%
Bilinear-MeanVec	52.2%
PPK-MeanVec	52.6%
PPK-Gaussian	65.4%

Table 3: Impact of training and testing with distribution rather than vector-based representations

tions used. Variants include: (i) Bilinear-WordVec: Max-margin training on word vector representations of words and images with a bilinear energy function. (ii) Bilinear-MeanVec: As before, but using our Gaussian means as vector representations in image and text domains. (iii) PPK-MeanVec: Train the max-margin model with Gaussian representation and PPK energy function as in our full model, but treat the resulting means as point estimates for conventional vector-based ZSL matching at testing-time. (v) PPK-Gaussian: Our full model with Gaussian PPK training and testing by Gaussian matching.

From the results in Table 3, we make the observations: (i) Bilinear-MeanVec outperforming Bilinear-WordVec shows that cross-modal (Sec 2.3) training of word Gaussians learns better point estimates of words than conventional word-vector training, since these only differ in the choice of vector representation of class names. (ii) PPK-Gaussian outperforming PPK-MeanVec shows that having a model of intra-class variability (as provided by the word-Gaussians) allows better zero-shot recognition, since these differ only in whether covariance is used at testing time.

3.4 Related Work and Discussion

Our approach models intra-class variability in both images and text. For example, the variability in visual appearance of military versus passenger 'plane's, and the variability in context according to whether a the word 'plane' is being used in a military or civilian sense. Given distribution-based representations in each domain, we find a cross-modal map that warps the two distributions into alignment.

Concurrently with our work, Ren et al (2016) present a related study on distribution-based visual-text embeddings. Methodologically, they benefit from end-to-end learning of deep features as well as cross-modal mapping, but they only discriminatively train word covariances, rather than jointly training both means and covariances as we do.

With regards to efficiency, our model is fast to train if fixing pre-trained word-Gaussians and optimising only the cross-modal mapping A. However, training the mapping jointly with the word-Gaussians comes at the cost of updating the representations of all words in the dictionary, and is thus much slower.

In terms of future work, an immediate improvement would be to generalise our of Gaussian embeddings to model concepts as mixtures of Gaussians or other exponential family distributions (Rudolph et al., 2016; Chen et al., 2015). This would for example, allow polysemy to be represented more cleanly as a mixture, rather than as a wide-covariance Gaussian as happens now. We would also like to explore distribution-based embeddings of sentences/paragraphs for class description (rather than class name) based zero-shot recognition (Reed et al., 2016). Finally, besides end-to-end deep learning of visual features, training non-linear cross-modal mappings is also of interest.

4 Conclusion

In this paper, we advocate using distribution-based embeddings of text and images when bridging the gap between vision and text modalities. This is in contrast to the common practice of point vector-based embeddings. Our distribution-based approach provides a representation of intra-class variability that improves zero-shot recognition, allows more meaningful retrieval by multiple keywords, and also produces better point-estimates of word vectors.

References

[Akata et al.2013] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. 2013. Label-embedding for attribute-based classification. In *Computer Vision and Pattern Recognition*.

[Arora et al.2015] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *CoRR*, abs/1502.03520.

[Blei et al.2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

[Bruni et al.2012a] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012a. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 136–145.

[Bruni et al.2012b] Elia Bruni, Jasper Uijlings, Marco Baroni, and Nicu Sebe. 2012b. Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *ACM Multimedia*.

[Bruni et al.2014] Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47, January.

[Chen et al.2015] Xinchi Chen, Xipeng Qiu, Jingxiang Jiang, and Xuanjing Huang. 2015. Gaussian mixture embeddings for multiple word prototypes. *arXiv preprint arXiv:1511.06246*.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

[Dinu et al.2015] Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *ICLR Workshop Paper*.

[Ferraresi et al.2008] Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *In Proceedings of the 4th Web as Corpus Workshop (WAC-4*.

[Frome et al.2013] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, MarcAurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Neural Information Processing Systems (NIPS)*.

[Fu et al.2014] Yanwei Fu, Timothy Hospedales, Tony Xiang, Zhenyong Fu, and Shaogang Gong. 2014. Transductive multi-view embedding for zero-shot recognition and annotation. In *European Conference on Computer Vision*.

[Fu et al.2015] Z. Fu, T. A. Xiang, E. Kodirov, and S. Gong. 2015. Zero-shot object recognition by semantic manifold distance. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2635–2644, June.

[Harris1954] Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

[Jebara et al.2004] T. Jebara, R. Kondor, and A. Howard. 2004. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844.

[Kiela and Bottou2014] Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

[Lampert et al.2009] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition*.

[Lazaridou et al.2014] Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, June.

[Mensink et al.2012] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. 2012. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

[Norouzi et al.2014] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. 2014. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*.

[Reed et al.2016] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

[Ren et al.2016] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. 2016. Joint image-text representation by gaussian visual semantic embedding. In *Proceeding of ACM International Conference on Multimedia (ACM MM)*.

[Romera-Paredes and Torr2015] Bernardino Romera-Paredes and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *ICML*.

[Rudolph et al.2016] Maja R. Rudolph, Francisco J. R. Ruiz, Stephan Mandt, and David M. Blei. 2016. Exponential Family Embeddings, August.

[Schwenk2007] Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21.

[Silberer and Lapata2014] Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*.

[Simonyan and Zisserman2015] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[Socher et al.2013] Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero Shot Learning Through Cross-Modal Transfer. In *Advances in Neural Information Processing Systems 26*.

[Vilnis and McCallum2015] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *ICLR*.

Question Relevance in VQA: Identifying Non-Visual And False-Premise Questions

Arijit Ray[1], **Gordon Christie**[1], **Mohit Bansal**[2], **Dhruv Batra**[3,1], **Devi Parikh**[3,1]
[1]Virginia Tech [2]UNC Chapel Hill [3]Georgia Institute of Technology
{ray93,gordonac,dbatra,parikh}@vt.edu
mbansal@cs.unc.edu

Abstract

Visual Question Answering (VQA) is the task of answering natural-language questions about images. We introduce the novel problem of determining the *relevance of questions to images* in VQA. Current VQA models do not reason about whether a question is even related to the given image (*e.g., What is the capital of Argentina?*) or if it requires information from external resources to answer correctly. This can break the continuity of a dialogue in human-machine interaction. Our approaches for determining relevance are composed of two stages. Given an image and a question, (1) we first determine whether the question is visual or not, (2) if visual, we determine whether the question is relevant to the given image or not. Our approaches, based on LSTM-RNNs, VQA model uncertainty, and caption-question similarity, are able to outperform strong baselines on both relevance tasks. We also present human studies showing that VQA models augmented with such question relevance reasoning are perceived as more intelligent, reasonable, and human-like.

1 Introduction

Visual Question Answering (VQA) is the task of predicting a suitable answer given an image and a question about it. VQA models (*e.g.,* (Antol et al., 2015; Ren et al., 2015)) are typically discriminative models that take in image and question representations and output one of a set of possible answers.

Our work is motivated by the following key observation – all current VQA systems always output an answer *regardless of whether the input question makes any sense for the given image or not*. Fig. 1 shows examples of relevant and irrelevant questions. When VQA systems are fed irrelevant questions as input, they understandably produce nonsensical answers (Q: *"What is the capital of Argentina?"* A: *"fire hydrant"*). Humans, on the other hand, are unlikely to provide such nonsensical answers and will instead answer that this is irrelevant or use another knowledge source to answer correctly, when possible. We argue that this implicit assumption by all VQA systems – that an input question is always relevant for the input image – is simply untenable as VQA systems move beyond standard academic datasets to interacting with real users, who may be unfamiliar, or malicious. The goal of this work is to make VQA systems more human-like by providing them the capability to identify relevant questions.

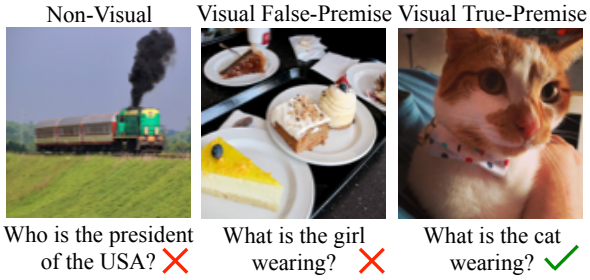


Figure 1: Example irrelevant (non-visual, false-premise) and relevant (visual true-premise) questions in VQA.

While existing work has reasoned about cross-modal similarity, being able to identify whether a question is relevant to a given image is a novel problem with real-world applications. In human-robot interaction, being able to identify questions that are dissociated from the perception data available is important. The robot must decide whether to process the scene it perceives or query external world knowledge resources to provide a response.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 919–924,
Austin, Texas, November 1-5, 2016.

As shown in Fig. 1, we study three types of question-image pairs: **Non-Visual**. These questions are not questions about images at all – they do not require information from *any* image to be answered (*e.g., "What is the capital of Argentina?"*). **Visual False-Premise**. While visual, these questions do not apply to the given image. For instance, the question *"What is the girl wearing?"* makes sense only for images that contain a girl in them. **Visual True-Premise**. These questions are relevant to (*i.e.,* have a premise which is true) the image at hand.

We introduce datasets and train models to recognize both non-visual and false-premise question-image (QI) pairs in the context of VQA. First, we identify whether a question is visual or non-visual; if visual, we identify whether the question has a true-premise for the given image. For visual vs. non-visual question detection, we use a Long Short-Term Memory (LSTM) recurrent neural network (RNN) trained on part of speech (POS) tags to capture visual-specific linguistic structure. For true vs. false-premise question detection, we present one set of approaches that use the uncertainty of a VQA model, and another set that use pre-trained captioning models to generate relevant captions (or questions) for the given image and then compare them to the given question to determine relevance.

Our proposed models achieve accuracies of 92% for detecting non-visual, and 74% for detecting false-premise questions, which significantly outperform strong baselines. We also show through human studies that a VQA system that reasons about question relevance is picked *significantly more often* as being more intelligent, human-like and reasonable than a baseline VQA system which does not. Our code and datasets are publicly available on the authors' webpages.

2 Related Work

There is a large body of existing work that reasons about cross-modal similarity: how well an image matches a query tag (Liu et al., 2009) in text-based image retrieval, how well an image matches a caption (Feng and Lapata, 2013; Xu et al., 2015; Ordonez et al., 2011; Karpathy and Fei-Fei, 2015; Fang et al., 2015), and how well a video matches a description (Donahue et al., 2015; Lin et al., 2014a).

In our work, if a question is deemed irrelevant, the VQA model says so, as opposed to answering the question anyway. This is related to perception systems that do not respond to an input where the system is likely to fail. Such failure prediction systems have been explored in vision (Zhang et al., 2014; Devarakota et al., 2007) and speech (Zhao et al., 2012; Sarma and Palmer, 2004; Choularton, 2009; Voll et al., 2008). Others attempt to provide the most meaningful answer instead of suppressing the output of a model that is expected to fail for a given input. One idea is to avoid a highly specific prediction if there is a chance of being wrong, and instead make a more generic prediction that is more likely to be right (Deng et al., 2012). Malinowski and Fritz (2014) use semantic segmentations in their approach to question answering, where they reason that objects not present in the segmentations should not be part of the answer.

To the best of our knowledge, our work is the first to study the relevance of questions in VQA. Chen et al. (2012) classify users' intention of questions for community question answering services. Most related to our work is Dodge et al. (2012). They extract visual text from within Flickr photo captions to be used as supervisory signals for training image captioning systems. Our motivation is to endow VQA systems the ability to detect non-visual questions to respond in a human-like fashion. Moreover, we also detect a more fine-grained notion of question relevance via true- and false-premise.

3 Datasets

For the task of detecting visual vs. non-visual questions, we assume all questions in the VQA dataset (Antol et al., 2015) are visual, since the Amazon Mechanical Turk (AMT) workers were specifically instructed to ask questions about a displayed image while creating it. We also collected non-visual philosophical and general knowledge questions from the internet (see supplementary material). Combining the two, we have 121,512 visual questions from the validation set of VQA and 9,952[1] generic non-visual questions collected from the internet. We call this dataset Visual vs. Non-

[1]High accuracies on this task in our experiments indicate that this suffices to learn the corresponding linguistic structure.

Visual Questions (VNQ).

We also collect a dataset of true- vs. false-premise questions by showing AMT workers images paired with random questions from the VQA dataset and asking them to annotate whether they are applicable or not. We had three workers annotate each QI pair. We take the majority vote as the final ground truth label.[2] We have 10,793 QI pairs on 1,500 unique images out of which 79% are non-applicable (false-premise). We refer to this visual true- vs. false-premise questions dataset as VTFQ.

Since there is a class imbalance in both of these datasets, we report the average per-class (*i.e.,* normalized) accuracy for all approaches. All datasets are publicly available.

4 Approach

Here we present our approaches for detecting (1) visual vs. non-visual QI pairs, and (2) true- vs. false-premise QI pairs.

4.1 Visual vs. Non-Visual Detection

Recall that the task here is to detect visual questions from non-visual ones. Non-visual questions, such as *"Do dogs fly?"* or *"Who is the president of the USA?"*, often tend to have a difference in the linguistic structure from that of visual questions, such as *"Does this bird fly?"* or *"What is this man doing?"*. We compare our approach (**LSTM**) with a baseline (**RULE-BASED**):

1. **RULE-BASED**. A rule-based approach to detect non-visual questions based on the part of speech (POS)[3] tags and dependencies of the words in the question. *E.g.,* if a question has a plural noun with no determiner before it and is followed by a singular verb (*"Do dogs fly?"*), it is a non-visual question.[4]

2. **LSTM**. We train an LSTM with 100-dim hidden vectors to embed the question into a vector and predict visual vs. not. Instead of feeding question words (['what', 'is', 'the', 'man', 'doing', '?']), the input to our LSTM is embeddings of POS tags of the words (['pronoun', 'verb', 'determiner', 'noun', 'verb']). Embeddings of the POS tags are learnt end-to-end. This captures the structure of image-grounded questions, rather than visual vs. non-visual topics. The latter are less likely to generalize across domains.

[2] 78% of the time all three votes agree.

[3] We use spaCy POS tagger (Honnibal and Johnson, 2015).

[4] See supplement for examples of such hand-crafted rules.

4.2 True- vs. False-Premise Detection

Our second task is to detect whether a question Q entails a false-premise for an image I. We present two families of approaches to measure this QI 'compatibility': (i) using uncertainty in VQA models, and (ii) using pre-trained captioning models.

Using VQA Uncertainty. Here we work with the hypothesis that if a VQA model is uncertain about the answer to a QI pair, the question may be irrelevant for the given image since the uncertainty may mean it has not seen similar QI pairs in the training data. We test two approaches:

1. **ENTROPY**. We compute the entropy of the softmax output from a state-of-the art VQA model (Antol et al., 2015; Lu et al., 2015) for a given QI pair and train a three-layer multilayer perceptron (MLP) on top with 3 nodes in the hidden layer.

2. **VQA-MLP**. We feed in the softmax output to a three-layer MLP with 100 nodes in the hidden layer, and train it as a binary classifier to predict whether a question has a true- or false-premise for the given image.

Using Pre-trained Captioning Models. Here we utilize (a) an image captioning model, and (b) an image question-generation model – to measure QI compatibility. Note that both these models generate natural language capturing the semantics of an image – one in the form of statement, the other in the form of a question. Our hypothesis is that a given question is relevant to the given image if it is similar to the language generated by these models for that image. Specifically:

1. Question-Caption Similarity (**Q-C SIM**). We use NeuralTalk2 (Karpathy and Fei-Fei, 2015) pre-trained on the MSCOCO dataset (Lin et al., 2014b) (images and associated captions) to generate a caption C for the given image, and then compute a learned similarity between Q and C (details below).

2. Question-Question Similarity (**Q-Q' SIM**). We use NeuralTalk2 re-trained (from scratch) on the questions in the VQA dataset to generate a question Q' for the image. Then, we compute a learned similarity between Q and Q'.

Visual vs. Non-Visual		True- vs. False-Premise				
RULE-BASED	LSTM	ENTROPY	VQA-MLP	Q-GEN SCORE	Q-C SIM	Q-Q' SIM
75.68	**92.27**	59.66	64.19	57.41	74.48	**74.58**

Table 1: Normalized accuracy results (averaged over 40 random train/test splits) for visual vs. non-visual detection and true- vs. false-premise detection. **RULE-BASED** and **Q-GEN SCORE** were not averaged because they are deterministic.

We now describe our learned Q-C similarity function (the Q-Q' similarity is analogous). Our Q-C similarity model is a 2-channel LSTM+MLP (one channel for Q, another for C). Each channel sequentially reads word2vec embeddings of the corresponding language via an LSTM. The last hidden state vectors (40-dim) from the 2 LSTMs are concatenated and fed as inputs to the MLP, which outputs a 2-class (relevant vs. not) softmax. The entire model is learned end-to-end on the VTFQ dataset. We also experimented with other representations (*e.g.,* bag of words) for Q, Q', C, which are included in the supplement for completeness.

Finally, we also compare our proposed models above to a simpler baseline (**Q-GEN SCORE**), where we compute the probability of the input question Q under the learned question-generation model. The intuition here is that since the question generation model has been trained only on relevant questions (from the VQA dataset), it will assign a high probability to Q if it is relevant.

5 Experiments and Results

The results for both experiments are presented in Table 1. We present results averaged over 40 random train/test splits. **RULE-BASED** and **Q-GEN SCORE** were not averaged because they are deterministic.

Visual vs. Non-Visual Detection. We use a random set of 100,000 questions from the VNQ dataset for training, and the remaining 31,464 for testing. We see that **LSTM** performs 16.59% (21.92% relative) better than **RULE-BASED**.

True- vs. False-Premise Detection. We use a random set of 7,195 (67%) QI pairs from the VTFQ dataset to train and the remaining 3,597 (33%) to test. While the VQA model uncertainty based approaches (**ENTROPY**, **VQA-MLP**) perform reasonably well (with the MLP helping over raw entropy), the learned similarity approaches perform much better (10.39% gain in normalized accuracy). High uncertainty of the model may suggest that a similar QI pair was not seen during training; however, that does not seem to translate to detecting irrelevance. The language generation models (**Q-C SIM**, **Q-Q' SIM**) seem to work significantly better at modeling the semantic interaction between the question and the image. The generative approach (**Q-GEN SCORE**) is outperformed by the discriminative approaches (**VQA-MLP**, **Q-C SIM**, **Q-Q' SIM**) that are trained explicitly for the task at hand. We show qualitative examples of **Q-Q' SIM** for true- vs. false-premise detection in Fig. 2.

6 Human Qualitative Evaluation

We also perform human studies where we compare two agents: (1) **AGENT-BASELINE**– always answers every question. (2) **AGENT-OURS**– reasons about question relevance before responding. If question is classified as visual true-premise, **AGENT-OURS** answers the question using the same VQA model as **AGENT-BASELINE** (using (Lu et al., 2015)). Otherwise, it responds with a prompt indicating that the question does not seem meaningful for the image.

A total of 120 questions (18.33% relevant, 81.67% irrelevant, mimicking the distribution of the VTFQ dataset) were used. Of the relevant questions, 54% were answered correctly by the VQA model. Human subjects on AMT were shown the response of both agents and asked to pick the agent that sounded more intelligent, more reasonable, and more human-like after every observed QI pair. Each QI pair was assessed by 5 different subjects. Not all pairs were rated by the same 5 subjects. In total, 28 unique AMT workers participated in the study.

AGENT-OURS was picked 65.8% of the time as the winner, **AGENT-BASELINE** was picked only 1.6% of the time, and both considered equally (un)reasonable in the remaining cases. We also measure the percentage of times each robot gets picked

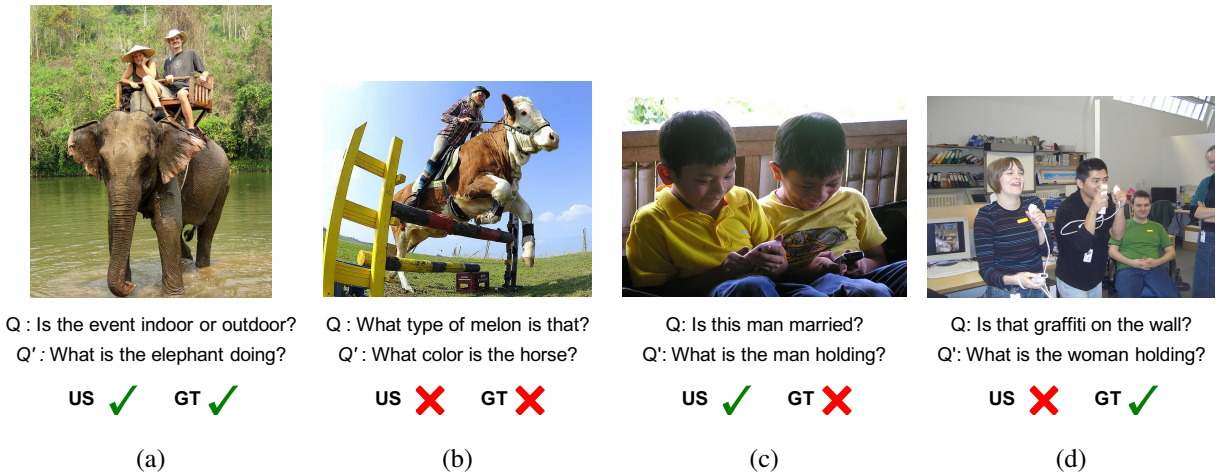


Figure 2: Qualitative examples for **Q-Q' SIM**. (a) and (b) show success cases, and (c) and (d) show failure cases. Our model predicts true-premise in (a) and (c), and false-premise in (b) and (d). In all examples we show the original question Q and the generated question Q'.

by the workers for true-premise, false-premise, and non-visual questions. These percentages are shown in Table 2.

	True-Premise	False-Premise	Non-Visual
AGENT-OURS	22.7	78.2	65.0
AGENT-BASELINE	04.7	01.4	00.0
Both	27.2	03.8	10.0
None	45.4	16.6	25.0

Table 2: Percentage of times each robot gets picked by AMT workers as being more intelligent, more reasonable, and more human-like for true-premise, false-premise, and non-visual questions.

Interestingly, humans often prefer **AGENT-OURS** over **AGENT-BASELINE** even when *both models are wrong* – **AGENT-BASELINE** answers the question incorrectly and **AGENT-OURS** incorrectly predicts that the question is irrelevant and refuses to answer a legitimate question. Users seem more tolerant to mistakes in relevance prediction than VQA.

7 Conclusion

We introduced the novel problem of identifying irrelevant (*i.e.,* non-visual or visual false-premise) questions for VQA. Our proposed models significantly outperform strong baselines on both tasks. A VQA agent that utilizes our detector and refuses to answer certain questions significantly outperforms a baseline (that answers all questions) in human studies. Such an agent is perceived as more intelligent, reasonable, and human-like.

There are several directions for future work. One possibility includes identifying the premise entailed in a question, as opposed to just stating true- or false-premise. Another is determining what external knowledge is needed to answer non-visual questions.

Our system can be further augmented to communicate to users what the assumed premise of the question is that is not satisfied by the image, *e.g.,* respond to *"What is the woman wearing?"* for an image of a cat by saying *"There is no woman."*

Acknowledgements

We thank Lucy Vanderwende for helpful suggestions and discussions. We also thank the anonymous reviewers for their helpful comments. This work was supported in part by the following: National Science Foundation CAREER awards to DB and DP, Alfred P. Sloan Fellowship, Army Research Office YIP awards to DB and DP, ICTAS Junior Faculty awards to DB and DP, Army Research Lab grant W911NF-15-2-0080 to DP and DB, Office of Naval Research grant N00014-14-1- 0679 to DB, Paul G. Allen Family Foundation Allen Distinguished Investigator award to DP, Google Faculty Research award to DP and DB, AWS in Education Research grant to DB, and NVIDIA GPU donation to DB.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Long Chen, Dell Zhang, and Levene Mark. 2012. Understanding User Intent in Community Question Answering. In *WWW*.

Stephen Choularton. 2009. *Early Stage Detection of Speech Recognition Errors*. Ph.D. thesis, Macquarie University.

Jia Deng, Jonathan Krause, Alexander C Berg, and Li Fei-Fei. 2012. Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition. In *CVPR*.

Pandu R Devarakota, Bruno Mirbach, and Björn Ottersten. 2007. Confidence estimation in classification decision: A method for detecting unseen patterns. In *ICAPR*.

Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Karl Stratos, Kota Yamaguchi, Yejin Choi, Hal Daumé, III, Alexander C. Berg, and Tamara L. Berg. 2012. Detecting Visual Text. In *NAACL HLT*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From Captions to Visual Concepts and Back. In *CVPR*.

Yansong Feng and Mirella Lapata. 2013. Automatic Caption Generation for News Images. *PAMI*, 35(4).

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *EMNLP*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *CVPR*.

Dahua Lin, Sanja Fidler, Chen Kong, and Raquel Urtasun. 2014a. Visual Semantic Search: Retrieving Videos via Complex Textual Queries. In *CVPR*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014b. Microsoft COCO: Common objects in context. In *ECCV*.

Dong Liu, Xian-Sheng Hua, Meng Wang, and HongJiang Zhang. 2009. Boost Search Relevance for Tag-based Social Image Retrieval. In *ICME*.

Jiasen Lu, Xiao Lin, Dhruv Batra, and Devi Parikh. 2015. Deeper LSTM and normalized CNN Visual Question Answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN`.

Mateusz Malinowski and Mario Fritz. 2014. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. In *NIPS*.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2Text: Describing Images Using 1 Million Captioned Photographs. In *NIPS*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *NIPS*.

Arup Sarma and David D Palmer. 2004. Context-based Speech Recognition Error Detection and Correction. In *NAACL HLT*.

Kimberly Voll, Stella Atkins, and Bruce Forster. 2008. Improving the Utility of Speech Recognition Through Error Detection. *Journal of Digital Imaging*, 21(4).

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*.

Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. 2014. Predicting Failures of Vision Systems. In *CVPR*.

Tongmu Zhao, Akemi Hoshino, Masayuki Suzuki, Nobuaki Minematsu, and Keikichi Hirose. 2012. Automatic Chinese Pronunciation Error Detection Using SVM Trained with Structural Features. In *Spoken Language Technology Workshop*.

Sort Story: Sorting Jumbled Images and Captions into Stories

Harsh Agrawal[*,1] **Arjun Chandrasekaran**[*,1,†]
Dhruv Batra[3,1] **Devi Parikh**[3,1] **Mohit Bansal**[4,2]
[1]Virginia Tech [2]TTI-Chicago [3]Georgia Institute of Technology [4]UNC Chapel Hill
{harsh92, carjun, dbatra, parikh}@vt.edu, mbansal@cs.unc.edu

Abstract

Temporal common sense has applications in AI tasks such as QA, multi-document summarization, and human-AI communication. We propose the task of *sequencing* – given a jumbled set of aligned image-caption pairs that belong to a story, the task is to sort them such that the output sequence forms a coherent story. We present multiple approaches, via unary (position) and pairwise (order) predictions, and their ensemble-based combinations, achieving strong results on this task. We use both text-based and image-based features, which depict complementary improvements. Using qualitative examples, we demonstrate that our models have learnt interesting aspects of temporal common sense.

1 Introduction

Sequencing is a task for children that is aimed at improving understanding of the temporal occurrence of a sequence of events. The task is, given a jumbled set of images (and maybe captions) that belong to a single story, sort them into the correct order so that they form a coherent story. Our motivation in this work is to enable AI systems to better understand and predict the temporal nature of events in the world. To this end, we train machine learning models to perform the task of "sequencing".

Temporal reasoning has a number of applications such as multi-document summarization of multiple sources of, say, news information where the relative order of events can be useful to accurately merge information in a temporally consistent manner. In question answering tasks (Richardson et al., 2013; Fader et al., 2014; Weston et al., 2015; Ren et al., 2015), answering questions related to when an event occurs, or what events occurred prior to a particular event require temporal reasoning. A good temporal model of events in everyday life, i.e., a "temporal common sense", could also improve the quality of communication between AI systems and humans.

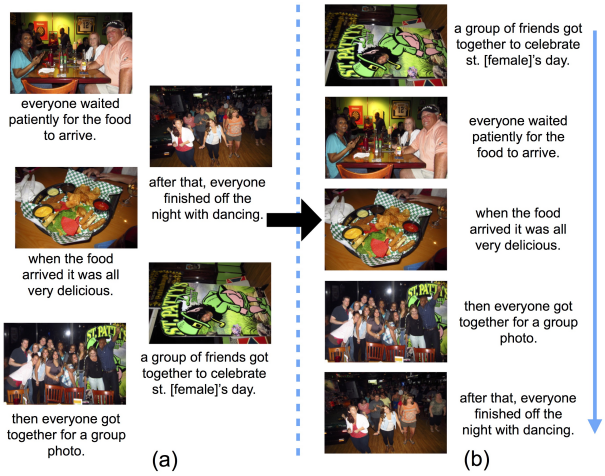


Figure 1: (a) The input is a jumbled set of aligned image-caption pairs. (b) Actual output of the system – an ordered sequence of image-caption pairs that form a coherent story.

Stories are a form of narrative sequences that have an inherent temporal common sense structure. We propose the use of visual stories depicting personal events to learn temporal common sense. We use stories from the Sequential Image Narrative Dataset (SIND) (Ting-Hao Huang, 2016) in which a set of 5 aligned image-caption pairs together form a coherent story. Given an input story that is jumbled (Fig. 1(a)), we train machine learning models to sort them into a coherent story (Fig. 1(b)).[1]

[*]Denotes equal contribution.
[†]Part of this work was done during an internship at TTIC.

[1]Note that 'jumbled' here refers to the loss of temporal ordering; image-caption pairs are still aligned.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 925–931,
Austin, Texas, November 1-5, 2016.

Our contributions are as follows:
– We propose the task of visual story sequencing.
– We implement two approaches to solve the task: one based on individual story elements to predict position, and the other based on pairwise story elements to predict relative order of story elements. We also combine these approaches in a voting scheme that outperforms the individual methods.
– As features, we represent a story element as both text-based features from the caption and image-based features, and show that they provide complementary improvements. For text-based features, we use both sentence context and relative order based distributed representations.
– We show qualitative examples of our models learning temporal common sense.

2 Related Work

Temporal ordering has a rich history in NLP research. Scripts (Schank and Abelson, 2013), and more recently, narrative chains (Chambers and Jurafsky, 2008) contain information about the participants and causal relationships between events that enable the understanding of stories. A number of works (Mani and Schiffman, 2005; Mani et al., 2006; Boguraev and Ando, 2005) learn temporal relations and properties of news events from the dense, expert-annotated TimeBank corpus (Pustejovsky et al., 2003). In our work, however, we use multi-modal story data that has no temporal annotations.

A number of works also reason about temporal ordering by using manually defined linguistic cues (Webber, 1988; Passonneau, 1988; Lapata and Lascarides, 2006; Hitzeman et al., 1995; Kehler, 2000). Our approach uses neural networks to avoid feature design for learning temporal ordering.

Recent works (Modi and Titov, 2014; Modi, 2016) learn distributed representations for predicates in a sentence for the tasks of event ordering and cloze evaluation. Unlike their work, our approach makes use of multi-modal data with *free-form* natural language text to learn event embeddings. Further, our models are trained end-to-end while their pipelined approach involves parsing and extracting verb frames from each sentence, where errors may propagate from one module to the next (as discussed in Section 4.3).

Chen et al. (2009) use a generalized Mallows model for modeling sequences for coherence within single documents. Their approach may also be applicable to our task. Recently, Mostafazadeh et al. (2016) presented the "ROCStories" dataset of 5-sentence stories with stereotypical causal and temporal relations between events. In our work though, we make use of a multi-modal story-dataset that contains *both* images and associated story-like captions.

Some works in vision (Pickup et al., 2014; Basha et al., 2012) also temporally order images; typically by finding correspondences between multiple images of the same scene using geometry-based approaches. Similarly, Choi et al. (2016) compose a story out of multiple short video clips. They define metrics based on scene dynamics and coherence, and use dense optical flow and patch-matching. In contrast, our work deals with stories containing potentially visually dissimilar but *semantically* coherent set of images and captions.

A few other recent works (Kim et al., 2015; Kim et al., 2014; Kim and Xing, 2014; Sigurdsson et al., 2016; Bosselut et al., 2016; Wang et al., 2016) summarize hundreds of individual streams of information (images, text, videos) from the web that deal with a single concept or event, to learn a common theme or *storyline* or for *timeline summarization*. Our task, however, is to predict the correct sorting of a given story, which is different from summarization or retrieval. Ramanathan et al. (2015) attempt to learn temporal embeddings of video frames in complex events. While their motivation is similar to ours, they deal with sampled frames from a video while we attempt to learn temporal common sense from *multi-modal* stories consisting of a sequence of aligned image-caption pairs.

3 Approach

In this section, we first describe the two components in our approach: unary scores that do not use context, and pairwise scores that encode relative orderings of elements. Next, we describe how we combine these scores through a voting scheme.

3.1 Unary Models

Let $\sigma \in \Sigma_n$ denote a permutation of n elements (image-caption pairs). We use σ_i to denote the position of element i in the permutation σ. A unary score

$S_u(\sigma)$ captures the appropriateness of each story element i in position σ_i:

$$S_u(\sigma) = \sum_{i=1}^{n} P(\sigma_i|i) \quad (1)$$

where $P(\sigma_i|i)$ denotes the probability of the element i being present in position σ_i, which is the output from an n-way softmax layer in a deep neural network. We experiment with 2 networks –
(1) A language-alone unary model (Skip-Thought+MLP) that uses a Gated Recurrent Unit (GRU) proposed by Cho et al. (2014) to embed a caption into a vector space. We use the Skip-Thought (Kiros et al., 2015) GRU, which is trained on the BookCorpus (Zhu et al., 2015) to predict the context (preceding and following sentences) of a given sentence. These embeddings are fed as input into a Multi-Layer Perceptron (MLP).
(2) A language+vision unary model (Skip-Thought+CNN+MLP) that embeds the caption as above and embeds the image via a Convolutional Neural Network (CNN). We use the activations from the penultimate layer of the 19-layer VGG-net (Simonyan and Zisserman, 2014), which have been shown to generalize well. Both embeddings are concatenated and fed as input to an MLP.

In both cases, the best ordering of the story elements (optimal permutation) $\sigma^* = \arg\max_{\sigma \in \Sigma_n} S_u(\sigma)$ can be found efficiently in $O(n^3)$ time with the Hungarian algorithm (Munkres, 1957). Since these unary scores are not influenced by other elements in the story, they capture the semantics and linguistic structures associated with specific positions of stories *e.g.*, the beginning, the middle, and the end.

3.2 Pairwise Models

Similar to learning to rank approaches (Hang, 2011), we develop pairwise scoring models that given a pair of elements (i, j), learn to assign a score:
$S([\![\sigma_i < \sigma_j]\!] \mid i, j)$ indicating whether element i should be placed before element j in the permutation σ. Here, $[\![\cdot]\!]$ indicates the Iverson bracket (which is 1 if the input argument is true and 0 otherwise). We develop and experiment with the following 3 pairwise models:
(1) A language-alone pairwise model (Skip-Thought+MLP) that takes as input a pair of Skip-Thought embeddings and trains an MLP (with hinge-loss) that outputs $S([\![\sigma_i < \sigma_j]\!] \mid i, j)$, the score for placing i before j.
(2) A language+vision pairwise model (Skip-Thought+CNN+MLP) that concatenates the Skip-Thought and CNN embeddings for i and j and trains a similar MLP as above.
(3) A language-alone neural position embedding (NPE) model. Instead of using frozen Skip-Thought embeddings, we learn a task-aware ordered distributed embedding for sentences. Specifically, each sentence in the story is embedded $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{x}_i \in \mathbb{R}^d_+$, via an LSTM (Hochreiter and Schmidhuber, 1997) with ReLU non-linearities. Similar to the max-margin loss that is applied to negative examples by Vendrov et al. (2016), we use an asymmetric penalty that encourages sentences appearing early in the story to be placed closer to the origin than sentences appearing later in the story.

$$\begin{aligned} L_{ij} &= \Big\| \max(0, \alpha - (\mathbf{x}_j - \mathbf{x}_i)) \Big\|^2 \\ Loss &= \sum_{1<=i<j=n} L_{ij} \end{aligned} \quad (2)$$

At train time, the parameters of the LSTM are learned end-to-end to minimize this asymmetric ordered loss (as measured over the gold-standard sequences). At test time, we use $S([\![\sigma_i < \sigma_j]\!] \mid i, j) = L_{ij}$. Thus, as we move away from the origin in the embedding space, we traverse through the sentences in a story. Each of these three pairwise approaches assigns a score $S(\sigma_i, \sigma_j | i, j)$ to an ordered pair of elements (i,j), which is used to construct a pairwise scoring model:

$$S_p(\sigma) = \sum_{1<=i<j<=n} \Big\{ S([\![\sigma_i < \sigma_j]\!]) - S([\![\sigma_j < \sigma_i]\!]) \Big\}, \quad (3)$$

by summing over the scores for all possible ordered pairs in the permutation. This pairwise score captures local contextual information in stories. Finding the best permutation $\sigma^* = \arg\max_{\sigma \in \Sigma_n} S_p(\sigma)$ under this pairwise model is NP-hard so approximations will be required. In our experiments, we study short sequences ($n = 5$), where the space of permutations is easily enumerable ($5! = 120$). For longer sequences, we can utilize integer programming methods or well-studied spectral relaxations for this problem.

3.3 Voting-based Ensemble

To combine the complementary information captured by the unary (S_u) and pairwise models (S_p), we use a voting-based ensemble. For each method in the ensemble, we find the top three permutations. Each of these permutations (σ^k) then vote for a particular element to be placed at a particular position. Let V be a vote matrix such that V_{ij} stores the number of votes for i^{th} element to occur at j^{th} position, *i.e.* $V_{ij} = \sum_k [\![\sigma_i^k == j]\!]$). We use the Hungarian algorithm to find the optimal permutation that maximizes the votes assigned, *i.e.* $\sigma^*_{\text{vote}} = \arg\max_{\sigma \in \Sigma_n} \sum_{i=1}^{n} \sum_{j=1}^{n} V_{ij} \cdot [\![\sigma_i == j]\!]$. We experimented with a number of model voting combinations and found the combination of pairwise Skip-Thought+CNN+MLP and neural position embeddings to work best (based on a validation set).

4 Experiments

4.1 Data

We train and evaluate our model on personal multimodal stories from the SIND (Sequential Image Narrative Dataset) (Ting-Hao Huang, 2016), where each story is a sequence of 5 images and corresponding story-like captions. The narrative captions in this dataset, e.g., "friends having a good time" (as opposed to "people sitting next to each other") capture a sequential, conversational language, which is characteristic of stories. We use 40,155 stories for training, 4990 for validation and 5055 stories for testing.

4.2 Metrics

We evaluate the performance of our model at correctly ordering a jumbled set of story elements using the following 3 metrics:

Spearman's rank correlation (Sp.) (Spearman, 1904) measures if the ranking of story elements in the predicted and ground truth orders are monotonically related (higher is better).

Pairwise accuracy (Pairw.) measures the fraction of pairs of elements whose predicted relative ordering is the same as the ground truth order (higher is better).

Average Distance (Dist.) measures the average change in position of all elements in the predicted story from their respective positions in the ground truth story (lower is better).

Method	Features	Sp.	Pairw.	Dist.
Random Order		0.000	0.500	1.601
Unary	SkipThought	0.508	0.718	1.373
	SkipThought + Image	0.532	0.729	1.352
Pairwise	SkipThought	0.546	0.732	0.923
	SkipThought + Image	0.565	0.740	0.897
Pairwise Order	NPE	0.480	0.704	1.010
Voting	SkipThought + Image (Pairwise) + NPE	**0.675**	**0.799**	**0.724**

Table 1: Performance of our different models and features at the sequencing task.

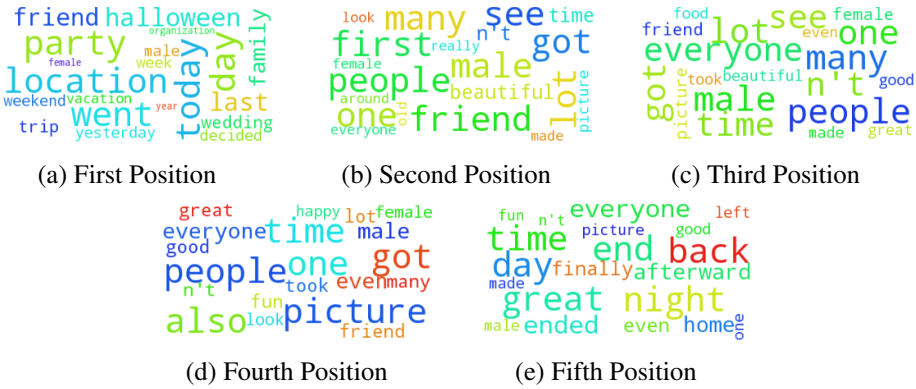
4.3 Results

Pairwise Models vs Unary Models As shown in Table 1, the pairwise models based on Skip-Thought features outperform the unary models in our task. However, the Pairwise Order Model performs worse than the unary Skip-Thought model, suggesting that the Skip-Thought features, which encode context of a sentence, also provide a crucial signal for temporal ordering of story sentences.

Contribution of Image Features Augmenting the text features with image features results in a visible performance improvement of both the model trained with unary features and the model trained with pairwise features. While image features by themselves result in poor performance on this task, they seem to capture temporal information that is complementary to the text features.

Ensemble Voting To exploit the fact that unary and pairwise models, as well as text and image features, capture different aspects of the story, we combine them using a voting ensemble. Based on the validation set, we found that combining the Pairwise Order model and the Pairwise model with both Skip-Thought and Image (CNN) features performs the best. This voting based method achieves the best performance on all three metrics. This shows that our different approaches indeed capture complementary information regarding feasible orderings of caption-image pairs to form a coherent story.

For comparison to existing related work, we tried



(a) First Position (b) Second Position (c) Third Position

(d) Fourth Position (e) Fifth Position

Figure 2: Word cloud corresponding to most discriminative words for each position.

to duplicate the pipelined approach of Modi and Titov (2014). For this, we first parse our story sentences to extract SVO (subject, verb, object) tuples (using the Stanford Parser (Chen and Manning, 2014)). However, this step succeeds for only 60% of our test data. Now even if we consider a *perfect* downstream algorithm that *always* makes the correct position prediction given SVO tuples, the overall performance is still a Spearman correlation of just 0.473, i.e., the *upper bound* performance of this pipelined approach is *lower* than the performance of our text-only end-to-end model (correlation of 0.546) in Table 1.

4.4 Qualitative Analysis

Visualizations of position predictions from our model demonstrate that it has learnt the *three act structure* (Trottier, 1998) in stories – the setup, the middle and the climax. We also present success and failure examples of our sorting model's predictions. See the supplementary for more details and figures.

We visualize our model's *temporal common sense*, in Fig. 2. The word clouds show *discriminative words* – the words that the model believes are indicative of sentence positions in a story. The size of a word is proportional to the ratio of its frequency of occurring in that position to other positions. Some words like 'party', 'wedding', etc., probably because our model believes that the start the story describes the setup – the occasion or event. People often tend to describe meeting friends or family members which probably results in the discriminative words such as 'people', 'friend', 'everyone' in the second and the third sentences. Moreover, the model believes that people tend to conclude the stories using words like 'finally', 'afterwards', tend to talk about 'great day', group 'pictures' with everyone, etc.

5 Conclusion

We propose the task of "sequencing" in a set of image-caption pairs, with the motivation of learning temporal common sense. We implement multiple neural network models based on individual and pairwise element-based predictions (and their ensemble), and utilize both image and text features, to achieve strong performance on the task. Our best system, on average, predicts the ordering of sentences to within a distance error of 0.8 (out of 5) positions. We also analyze our predictions and show qualitative examples that demonstrate temporal common sense.

Acknowledgements

We thank Ramakrishna Vedantam and the anonymous reviewers for their helpful suggestions. This work was supported by: NSF CAREER awards to DB and DP, ARO YIP awards to DB and DP, ICTAS Junior Faculty awards to DB and DP, Google Faculty Research award to DP and DB, ARL grant W911NF-15-2-0080 to DP and DB, ONR grant N00014-14-1-0679 to DB and N00014-16-1-2713 to DP, ONR YIP award to DP, Paul G. Allen Family Foundation Allen Distinguished Investigator award to DP, Alfred P. Sloan Fellowship to DP, AWS in Education Research grant to DB, NVIDIA GPU donations to DB and MB, an IBM Faculty Award and Bloomberg Data Science Research Grant to MB.

References

[Basha et al.2012] Tali Basha, Yael Moses, and Shai Avidan. 2012. Photo sequencing. In *ECCV*. 2

[Boguraev and Ando2005] Branimir Boguraev and Rie Kubota Ando. 2005. Timeml-compliant text analysis for temporal reasoning. In *IJCAI*. 2

[Bosselut et al.2016] Antoine Bosselut, Jianfu Chen, David Warren, Hannaneh Hajishirzi, and Yejin Choi. 2016. Learning prototypical event structure from photo albums. In *ACL*. 2

[Chambers and Jurafsky2008] Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*. Citeseer. 2

[Chen and Manning2014] Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*. 5

[Chen et al.2009] Harr Chen, SRK Branavan, Regina Barzilay, David R Karger, et al. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

[Choi et al.2016] Jinsoo Choi, Tae-Hyun Oh, and In So Kweon. 2016. Video-story composition via plot analysis. In *CVPR*.

[Fader et al.2014] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *ACM SIGKDD*. 1

[Hang2011] LI Hang. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*. 3

[Hitzeman et al.1995] Janet Hitzeman, Marc Moens, and Claire Grover. 1995. Algorithms for analysing the temporal structure of discourse. In *EACL*. 2

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*. 3

[Kehler2000] Andrew Kehler. 2000. Coherence and the resolution of ellipsis. *Linguistics and Philosophy*. 2

[Kim and Xing2014] Gunhee Kim and Eric Xing. 2014. Reconstructing storyline graphs for image recommendation from web community photos. In *CVPR*. 2

[Kim et al.2014] Gunhee Kim, Leonid Sigal, and Eric Xing. 2014. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *CVPR*. 2

[Kim et al.2015] Gunhee Kim, Seungwhan Moon, and Leonid Sigal. 2015. Joint photo stream and blog post summarization and exploration. In *CVPR*. 2

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*. 3

[Lapata and Lascarides2006] Mirella Lapata and Alex Lascarides. 2006. Learning sentence-internal temporal relations. *Journal of Artificial Intelligence Research*. 2

[Mani and Schiffman2005] Inderjeet Mani and Barry Schiffman. 2005. Temporally anchoring and ordering events in news. *Time and Event Recognition in Natural Language. John Benjamins*. 2

[Mani et al.2006] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *COLING-ACL*. 2

[Modi and Titov2014] Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *CoNLL*. 2

[Modi2016] Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *CoNLL*. 2

[Mostafazadeh et al.2016] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*.

[Munkres1957] James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*. 3

[Passonneau1988] Rebecca J Passonneau. 1988. A computational model of the semantics of tense and aspect. *Computational Linguistics*. 2

[Pickup et al.2014] Lyndsey Pickup, Zheng Pan, Donglai Wei, YiChang Shih, Changshui Zhang, Andrew Zisserman, Bernhard Scholkopf, and William Freeman. 2014. Seeing the arrow of time. In *CVPR*. 2

[Pustejovsky et al.2003] James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*. 2

[Ramanathan et al.2015] Vignesh Ramanathan, Kevin Tang, Greg Mori, and Li Fei-Fei. 2015. Learning temporal embeddings for complex video analysis. In *CVPR*.

[Ren et al.2015] Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *NIPS*. 1

[Richardson et al.2013] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*. 1

[Schank and Abelson2013] Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press. 2

[Sigurdsson et al.2016] Gunnar A Sigurdsson, Xinlei Chen, and Abhinav Gupta. 2016. Learning visual storylines with skipping recurrent neural networks. In *ECCV*. 2

[Simonyan and Zisserman2014] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 3

[Spearman1904] Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*. 4

[Ting-Hao Huang2016] Nasrin Mostafazadeh Ishan Misra Aishwarya Agrawal Jacob Devlin Ross Girshick Xiaodong He Pushmeet Kohli Dhruv Batra C. Lawrence Zitnick Devi Parikh Lucy Vanderwende Michel Galley Margaret Mitchell Ting-Hao Huang, Francis Ferraro. 2016. Visual storytelling. In *NAACL*. 1, 4

[Trottier1998] David Trottier. 1998. *The screenwriter's bible: A complete guide to writing, formatting, and selling your script*. Silman-James Press. 5

[Vendrov et al.2016] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

[Wang et al.2016] William Yang Wang, Yashar Mehdad, Dragomir R Radev, and Amanda Stent. 2016. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. In *NAACL*. 2

[Webber1988] Bonnie Lynn Webber. 1988. Tense as discourse anaphor. *Computational Linguistics*. 2

[Weston et al.2015] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*. 1

[Zhu et al.2015] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *CVPR*. 3

Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions?

Abhishek Das[1*] **Harsh Agrawal**[1*] **C. Lawrence Zitnick**[2] **Devi Parikh**[1,3] **Dhruv Batra**[1,3]
[1]Virginia Tech [2]Facebook AI Research [3]Georgia Institute of Technology
{abhshkdz, harsh92, parikh, dbatra}@vt.edu, zitnick@fb.com

Abstract

We conduct large-scale studies on 'human attention' in Visual Question Answering (VQA) to understand where humans choose to look to answer questions about images. We design and test multiple game-inspired novel attention-annotation interfaces that require the subject to sharpen regions of a blurred image to answer a question. Thus, we introduce the VQA-HAT (Human ATtention) dataset. We evaluate attention maps generated by state-of-the-art VQA models against human attention both qualitatively (via visualizations) and quantitatively (via rank-order correlation). Overall, our experiments show that current VQA attention models do not seem to be looking at the same regions as humans.

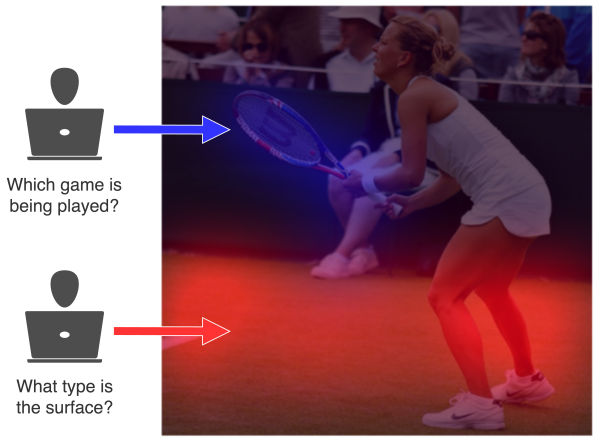


Figure 1: Different human attention regions based on question. (best viewed in color)

1 Introduction

It helps to pay attention. Humans have the ability to quickly perceive a scene by selectively attending to parts of the image instead of processing the whole scene in its entirety (Rensink, 2000). Inspired by human attention, a recent trend in computer vision and deep learning is to build computational models of attention. Given an input signal, these models learn to attend to parts of it for further processing and have been successfully applied in machine translation (Bahdanau et al., 2015; Firat et al., 2016), object recognition (Ba et al., 2015; Mnih et al., 2014; Sermanet et al., 2014), image captioning (Xu et al., 2015; Cho et al., 2015) and visual question answering (Yang et al., 2016; Lu et al., 2016; Xu and Saenko, 2015; Xiong et al., 2016).

In this work, we study attention for the task of Visual Question Answering (VQA). Unlike image captioning, where a coarse understanding of an image is often sufficient for producing generic descriptions (Devlin et al., 2015), visual questions selectively target different areas of an image including background details and underlying context. This suggests that a VQA model may benefit from an explicit or implicit attention mechanism to answer a question correctly. In this work, we are interested in the following questions: 1) Which image regions do humans choose to look at in order to answer questions about images? 2) Do deep VQA models with attention mechanisms attend to the same regions as humans?

We design and conduct studies to collect "human attention maps". Figure 1 shows human attention maps on the same image for two different questions. When asked 'What type is the surface?', humans choose to look at the floor, while attention for 'Which game is being played?' is concentrated around the player and racket.

These human attention maps can be used both for evaluating machine-generated attention maps and for explicitly training attention-based models.

[*]Denotes equal contribution.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 932–937,
Austin, Texas, November 1-5, 2016.

(a) (b) (c)

Figure 2: (a-c): Column 1 shows deblurred image, and column 2 shows human attention map.

Contributions. First, we design game-inspired novel interfaces for collecting human attention maps of where humans choose to look to answer questions from the large-scale VQA dataset (Antol et al., 2015); this VQA-HAT (Human ATtention) dataset is publicly available at our project webpage[1] Second, we perform qualitative and quantitative comparison of the maps generated by state-of-the-art attention-based VQA models (Yang et al., 2016; Lu et al., 2016) and a task-independent saliency baseline (Judd et al., 2009) against our human attention maps through visualizations and rank-order correlation. We find that machine-generated attention maps from the most accurate VQA model have a mean rank-correlation of 0.26 with human attention maps, which is worse than task-independent saliency maps that have a mean rank-correlation of 0.49. It is well understood that task-independent saliency maps have a 'center bias' (Tatler, 2007; Judd et al., 2009). After we control for this center bias, we find that the correlation of task-independent saliency is poor (as expected), while trends for machine-generated VQA-attention maps remain the same, which confirms our key finding that current VQA attention models do not seem to be looking at the same regions as humans.

2 Related Work

Our work draws on recent work in attention-based VQA and human studies in saliency prediction. We work with the free-form and open-ended VQA dataset released by (Antol et al., 2015).

VQA Models. Attention-based models for VQA typically use convolutional neural networks to highlight relevant regions of image given a question. Stacked Attention Networks (SAN) proposed in (Yang et al., 2016) use LSTM encodings of question words to produce a spatial attention distribution over the convolutional layer features of the image. Hierarchical Co-Attention Network (Lu et al., 2016) generates multiple levels of image attention based on words, phrases and complete questions, and is the top entry on the VQA Challenge[2] as of the time of this submission. Another interesting approach uses question parsing to compose the neural network from modules, attention being one of the sub-tasks addressed by these modules (Andreas et al., 2016). Note that all these works are *unsupervised* attention models, where "attention" is simply an intermediate variable (a spatial distribution) that is produced by the model to optimize downstream loss (VQA cross-entropy). The fact that some (it's unclear how many) of these spatial distributions end up being interpretable is simply fortuitous. In contrast, we study where humans choose to look to answer visual questions. These human attention maps can be used to evaluate unsupervised maps.

Human Studies. There's a rich history of work in collecting eye tracking data from human subjects to gain an understanding of image saliency and visual perception (Jiang et al., 2014; Judd et al., 2009; Fei-Fei et al., 2007; Yarbus, 1967). Eye tracking data to study natural visual exploration (Jiang et al., 2014; Judd et al., 2009) is useful but difficult and expensive to collect on a large scale. (Jiang et al., 2015) established mouse tracking as an accurate alternative to eye tracking for collecting attention maps. They collected large-scale attention annotations for MS COCO (Lin et al., 2014) on Ama-

[1] http://computing.ece.vt.edu/~abhshkdz/vqa-hat

[2] http://visualqa.org/challenge.html

zon Mechanical Turk (AMT). While (Jiang et al., 2015) studies natural exploration and collects task-independent human annotations by asking subjects to freely move the mouse cursor to anywhere they wanted to look on a blurred image, our approach is task-driven. (Jia Deng and Jonathan Krause and Li Fei-Fei, 2013; Deng et al., 2015) leverage crowdsourcing to help computers select discriminative features for fine-grained recognition. They introduce a novel gamified setting where the humans can reveal regions with certain penalty which ensures discriminative regions with assured quality. Related to this is the work of (von Ahn and Dabbish, 2004) who explore gamification to locate objects in an image. To the best of our knowledge, this is the first work to collect human attention maps for VQA.

Specifically, as described in Section 3, we collect ground truth attention annotations by instructing subjects to sharpen parts of a blurred image that are important for answering the questions accurately. Section 4 covers evaluation of unsupervised attention maps generated by VQA models against our human attention maps.

3 VQA-HAT (Human ATtention) Dataset

We design and test multiple game-inspired novel interfaces for conducting large-scale human studies on AMT. Our basic interface design consists of a "deblurring" exercise for answering visual questions. Specifically, we present subjects with a blurred image and a question about the image, and ask subjects to sharpen regions of the image that will help them answer the question correctly, in a smooth, click-and-drag, 'coloring' motion with the mouse. The sharpening is gradual: successively scrubbing the same region progressively sharpens it.

We experiment with multiple variants of the data collection interface. Analysis of the interfaces as well as details of the human evaluation studies conducted to converge on the final interface used for results in this main document have been included in the supplement. The human evaluation studies consisted of showing these attention-sharpened images to humans and asking them to answer the question. Based on these human studies, we pick the "Blurred Image with Answer" interface, where subjects were shown the correct answer in addition to the question and blurred image, and asked to deblur as few regions as possible such that someone can answer the question just by looking at the sharpened regions. Since the payment structure on AMT encourage completing tasks as quickly as possible, this implicitly incentivizes subjects to deblur as few regions as possible. Our followup human studies on these collected maps show that other subjects are able to answer questions based on these collected maps (details in supplement). Thus, overall we achieve a balance between highlighting too little or too much.

Note that the "Blurred Image with Answer" interface used to collect attention maps is a verification task as opposed to actual question answering. We show subjects an answer and ask them to sharpen regions that will help them answer the question correctly, as opposed to showing them just the question and asking them for the answer as well as relevant sharpened regions in the image ("Blurred Image without Answer" interface). Attention maps collected via this verification task "Blurred Image with Answer" are more informative (in terms of human VQA accuracy) than those collected for "Blurred Image without Answer" – 78.7% vs. 75.2%.

We collected human attention maps for 58475 train (out of 248349 total) and 1374 val (out of 121512 total) question-image pairs from the VQA dataset. This dataset is publicly available[1]. Overall, we conducted approximately 20000 Human Intelligence Tasks (HITs) on AMT, among 800 unique workers. Figure 2 shows examples of collected human attention maps.

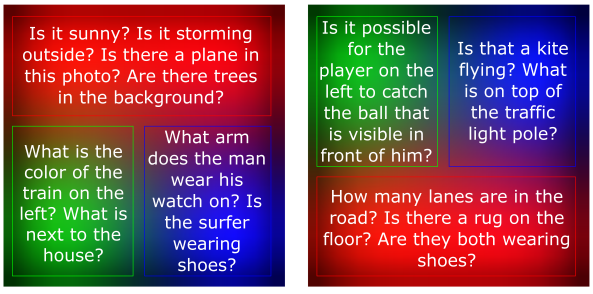


Figure 3

To visualize the collected dataset, we cluster the human attention maps and visualize the average attention map and example questions falling in each of them for 6 selected clusters in Figure 3.

4 Human Attention Maps vs Unsupervised Attention Models

Now that we have collected these human attention maps, we can ask the following question – do unsupervised attention models learn to predict attention maps that are similar to human attention maps? To rephrase, *do neural networks look at the same regions as humans to answer a visual question?*

VQA Attention Models. We evaluate maps generated by the following unsupervised models:

- Stacked Attention Network (SAN) (Yang et al., 2016) with two attention layers (SAN-2)[3].
- Hierarchical Co-Attention Network (HieCoAtt) (Lu et al., 2016) with word-level (HieCoAtt-W), phrase-level (HieCoAtt-P) and question-level (HieCoAtt-Q) attention maps; we evaluate all three maps[4].

Comparison Metric: Rank Correlation. We first scale both the machine-generated and human attention maps to 14x14, rank the pixels according to their spatial attention and then compute correlation between these two ranked lists. We choose an order-based metric so as to make the evaluation invariant to absolute spatial probability values which can be made peaky or diffuse by tweaking a 'temperature' parameter.

Table 1 shows rank-order correlation averaged over all image-question pairs on the validation set. We compare with random attention maps and task-independent saliency maps generated by a model trained to predict human eye fixation locations where subjects are asked to freely view an image for 3 seconds (Judd et al., 2009). Both SAN-2 and HieCoAtt attention maps are positively correlated with human attention maps, but not as strongly as task-independent Judd saliency maps. Our findings lead to two take-away messages with significant potential impact on future research in this active field. First, current VQA attention models do not seem to be 'looking' at the same regions as humans to produce an answer. Second, as attention-based VQA models become more accurate (58.9% SAN $\rightarrow$ 62.1% HieCoAtt), they seem to be (slightly) better correlated with humans in terms of where they look. Our dataset will allow for a more thorough validation of this observation as future attention-based VQA models are proposed. Figure 4 shows examples of human and machine-generated attention maps with their rank-correlation coefficients.

Model	Rank-correlation
SAN-2 (Yang et al., 2016)	0.249 ± 0.004
HieCoAtt-W (Lu et al., 2016)	0.246 ± 0.004
HieCoAtt-P (Lu et al., 2016)	0.256 ± 0.004
HieCoAtt-Q (Lu et al., 2016)	0.264 ± 0.004
Random	0.000 ± 0.001
Judd et al. (Judd et al., 2009)	0.497 ± 0.004
Human	0.623 ± 0.003

Table 1: Mean rank-correlation coefficients (higher is better); error bars show standard error of means. We can see that both SAN-2 and HieCoAtt attention maps are positively correlated with human attention maps, but not as strongly as task-independent Judd saliency maps.

Model	Rank-correlation
SAN-2 (Yang et al., 2016)	0.038 ± 0.011
HieCoAtt-W (Lu et al., 2016)	0.062 ± 0.012
HieCoAtt-P (Lu et al., 2016)	0.048 ± 0.010
HieCoAtt-Q (Lu et al., 2016)	0.114 ± 0.012
Judd et al. (Judd et al., 2009)	-0.063 ± 0.009

Table 2: Correlation on the reduced set without center bias goes down significantly for Judd saliency since they have a strong center bias. Relative trends among SAN-2 & HieCoAtt are similar to those over the whole validation set (reported in Table 1).

To put these numbers in perspective, we computed inter-human agreement on the validation set by collecting 3 human attention maps per image-question pair and computing mean rank-correlation, which is 0.623. Lastly, all reported correlations are averaged over 3 trials by adding random noise (order of 10^{-14}) to human attention maps to account for ranking variations in case of uniformly weighted regions.

Center Bias. Judd saliency maps aim to predict human eye fixations during natural visual exploration. These tend to have a strong center bias (Tatler, 2007; Judd et al., 2009). Although our human attention maps dataset is not an eye tracking study, the cen-

[3] https://github.com/zcyang/imageqa-san

[4] https://github.com/jiasenlu/HieCoAttenVQA

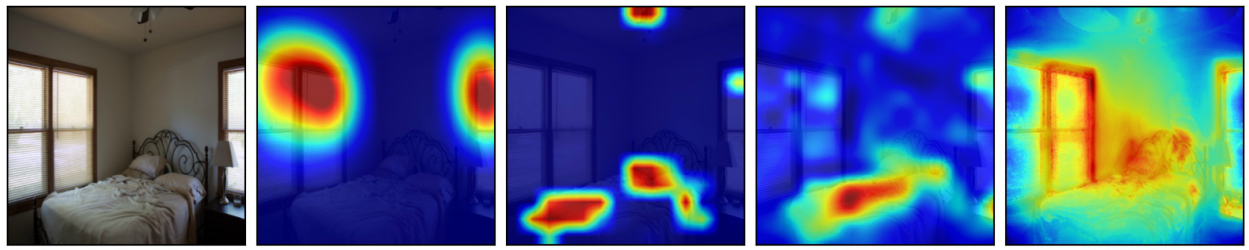


Figure 4: Random example of human attention (column 2) v/s machine-generated attention (columns 3-5)

ter bias still exists albeit not as severely as in eye-tracking. A potential source of center bias is the fact that the VQA dataset was human-generated by subjects looking at images. Thus, salient objects in the center of the image are likely to be potential subjects of questions. We compute rank-correlation of a synthetically generated central attention map with Judd saliency and human attention maps. Judd saliency maps have a mean rank-correlation of 0.877 and human attention maps have a mean rank-correlation of 0.458 on the validation set.

To eliminate the effect of center bias in this evaluation, we removed human attention maps that have positive rank-correlation with the center attention map. We compute rank-correlation of machine-generated attention with human attention on this reduced set. See Table 2. Mean correlation goes down significantly for Judd saliency maps since they have a strong center bias. Relative trends among SAN-2 & HieCoAtt are similar to those over the whole validation set (reported in Table 1). HieCoAtt-Q now has higher correlation with human attention maps than Judd saliency. Thus discounting the center bias, VQA-specific machine attention maps correlate better with VQA-specific human attention maps than task-independent machine saliency maps.

5 Conclusion & Discussion

We introduce and release the VQA-HAT dataset[1]. This dataset can be used to evaluate attention maps generated in an unsupervised manner by attention-based VQA models, or to explicitly train models with attention supervision for VQA. We quantify whether current attention-based VQA models are 'looking' at the same regions of the image as humans do to produce an answer.

Necessary vs Sufficient Maps. Are human attention maps 'necessary' and/or 'sufficient'? If regions highlighted by the human attention maps are sufficient to answer the question accurately, then so is any region that is a superset. For example, if attention mass is concentrated on a 'cat' for 'What animal is present in the picture?', then an attention map that assigns weights to any arbitrary-sized region that includes the 'cat' is sufficient as well. On the contrary, a *necessary* and sufficient attention map would be the smallest visual region sufficient for answering the question accurately. It is an ill-posed problem to define a necessary attention map in the space of pixels; random pixels can be blacked out and chances are that humans would still be able to answer the question given the resulting subset attention map. Our work thus poses an interesting question for future work – what is the right *semantic* space in which it is meaningful to talk about necessary and sufficient attention maps for humans?

Acknowledgements

We thank Jiasen Lu and Rama Vedantam for helpful suggestions. This work was supported in part by the National Science Foundation CAREER awards to DB & DP, Army Research Office YIP awards to DB & DP, ICTAS Junior Faculty awards at VT to DB & DP, Army Research Lab grant W911NF-15-2-0080 to DP & DB, Office of Naval Research (ONR) YIP award to DP, ONR grant N00014-14-1-0679 to DB, Alfred P. Sloan Fellowship to DP, Paul G. Allen Family Foundation Allen Distinguished Investigator award to DP, Google Faculty Research award to DP & DB, AWS in Education Research grant to DB, and NVIDIA GPU donation to DB. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Government or any sponsor.

References

[Andreas et al.2016] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *NAACL HLT*. 2

[Antol et al.2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*. 2

[Ba et al.2015] Jimmy Lei Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple Object Recognition With Visual Attention. In *ICLR*. 1

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*. 1

[Cho et al.2015] KyungHyun Cho, Aaron C. Courville, and Yoshua Bengio. 2015. Describing Multimedia Content using Attention-based Encoder-Decoder Networks. volume abs/1507.01053. 1

[Deng et al.2015] Jia Deng, Jonathan Krause, Michael Stark, and Li Fei-Fei. 2015. Leveraging the Wisdom of the Crowd for Fine-Grained Recognition. *PAMI*. 3

[Devlin et al.2015] Jacob Devlin, Saurabh Gupta, Ross B. Girshick, Margaret Mitchell, and C. Lawrence Zitnick. 2015. Exploring nearest neighbor approaches for image captioning. volume abs/1505.04467. 1

[Fei-Fei et al.2007] Li Fei-Fei, Asha Iyer, Christof Koch, and Pietro Perona. 2007. What do we perceive in a glance of a real-world scene? *Journal of Vision*, 7(1):10. 2

[Firat et al.2016] Orhan Firat, KyungHyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. volume abs/1601.01073. 1

[Jia Deng and Jonathan Krause and Li Fei-Fei2013] Jia Deng and Jonathan Krause and Li Fei-Fei. 2013. Fine-Grained Crowdsourcing for Fine-Grained Recognition. In *CVPR*. 3

[Jiang et al.2014] Ming Jiang, Juan Xu, and Qi Zhao. 2014. Saliency in Crowd. In *ECCV*. 2

[Jiang et al.2015] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. 2015. Salicon: Saliency in context. In *CVPR*. 2, 3

[Judd et al.2009] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. 2009. Learning to predict where humans look. In *ICCV*. 2, 4

[Lin et al.2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*. 2

[Lu et al.2016] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical Question-Image Co-Attention for Visual Question Answering. In *NIPS*. 1, 2, 4

[Mnih et al.2014] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *NIPS*. 1

[Rensink2000] Ronald A. Rensink. 2000. The dynamic representation of scenes. *Visual Cognition*, 7(1-3):17–42. 1

[Sermanet et al.2014] Pierre Sermanet, Andrea Frome, and Esteban Real. 2014. Attention for Fine-Grained Categorization. volume abs/1412.7054. 1

[Tatler2007] Benjamin W. Tatler. 2007. The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions. *Journal of Vision*, 7(14):4. 2, 4

[von Ahn and Dabbish2004] Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *CHI*. 3

[Xiong et al.2016] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. 1

[Xu and Saenko2015] Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. volume abs/1511.05234. 1

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*. 1

[Yang et al.2016] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2016. Stacked Attention Networks for Image Question Answering. In *CVPR*. 1, 2, 4

[Yarbus1967] A. L. Yarbus. 1967. *Eye Movements and Vision*. Plenum. New York. 2

Recurrent Residual Learning for Sequence Classification

Yiren Wang*
University of Illinois at Urbana-Champaign
yiren@illinois.edu

Fei Tian
Microsoft Research
fetia@microsoft.com

Abstract

In this paper, we explore the possibility of leveraging Residual Networks (ResNet), a powerful structure in constructing extremely deep neural network for image understanding, to improve recurrent neural networks (RNN) for modeling sequential data. We show that for sequence classification tasks, incorporating residual connections into recurrent structures yields similar accuracy to Long Short Term Memory (LSTM) RNN with much fewer model parameters. In addition, we propose two novel models which combine the best of both residual learning and LSTM. Experiments show that the new models significantly outperform LSTM.

1 Introduction

Recurrent Neural Networks (RNNs) are powerful tools to model sequential data. Among various RNN models, Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is one of the most effective structures. In LSTM, gating mechanism is used to control the information flow such that gradient vanishing problem in vanilla RNN is better handled, and long range dependency is better captured. However, as empirically verified by previous works and our own experiments, to obtain fairly good results, training LSTM RNN needs carefully designed optimization procedure (Hochreiter et al., 2001; Pascanu et al., 2013; Dai and Le, 2015; Laurent et al., 2015; He et al., 2016; Arjovsky et al., 2015), especially when faced with unfolded very deep architectures for fairly long sequences (Dai and Le, 2015).

From another perspective, for constructing very deep neural networks, recently Residual Networks (ResNet) (He et al., 2015) have shown their effectiveness in quite a few computer vision tasks. By learning a residual mapping between layers with identity skip connections (Jaeger et al., 2007), ResNet ensures a fluent information flow, leading to efficient optimization for very deep structures (e.g., with hundreds of layers). In this paper, we explore the possibilities of leveraging residual learning to improve the performances of recurrent structures, in particular, LSTM RNN, in modeling fairly long sequences (i.e., whose lengths exceed 100). To summarize, our main contributions include:

1. We introduce residual connecting mechanism into the recurrent structure and propose recurrent residual networks for sequence learning. Our model achieves similar performances to LSTM in text classification tasks, whereas the number of model parameters is greatly reduced.

2. We present in-depth analysis of the strengths and limitations of LSTM and ResNet in respect of sequence learning.

3. Based on such analysis, we further propose two novel models that incorporate the strengths of the mechanisms behind LSTM and ResNet. We demonstrate that our models outperform LSTM in many sequence classification tasks.

*This work was done when the author was visiting Microsoft Research Asia.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 938–943,
Austin, Texas, November 1-5, 2016.

2 Background

RNN models sequences by taking sequential input $x = \{x_1, \cdots, x_T\}$ and generating T step hidden states $h = \{h_1, \cdots, h_T\}$. At each time step t, RNN takes the input vector $x_t \in \mathcal{R}^n$ and the previous hidden state vector $h_{t-1} \in \mathcal{R}^m$ to produce the next hidden state h_t.

Based on this basic structure, LSTM avoids gradient vanishing in RNN training and thus copes better with long range dependencies, by further augmenting vanilla RNN with a memory cell vector $c_t \in \mathcal{R}^m$ and multiplicative gate units that regulate the information flow. To be more specific, at each time step t, an LSTM unit takes x_t, c_{t-1}, h_{t-1} as input, generates the input, output and forget gate signals (denoted as i_t, o_t and f_t respectively), and produces the next cell state c_t and hidden state h_t:

$$\begin{aligned} \tilde{c}_t &= tanh(W_c[h_{t-1}, x_t] + b_c) \\ i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ c_t &= f_t \otimes C_{t-1} + i_t \otimes \tilde{c}_t \\ h_t &= o_t \otimes tanh(c_t) \end{aligned} \tag{1}$$

where $\otimes$ refers to element-wise product. $\sigma(x)$ is the sigmoid function $\sigma(x) = 1/(1+\exp(-x))$. $W_j (j \in \{i, o, f, c\})$ are LSTM parameters. In the following part, such functions generating h_t and c_t are denoted as $h_t, c_t = LSTM(x_t, h_{t-1}, c_{t-1})$.

Residual Networks (ResNet) are among the pioneering works (Szegedy et al., 2015; Srivastava et al., 2015) that utilize extra identity connections to enhance information flow such that very deep neural networks can be effectively optimized. ResNet (He et al., 2015) is composed of several stacked residual units, in which the l^{th} unit takes the following transformation:

$$h_{l+1} = f(g(h_l) + \mathcal{F}(h_l; W_l)) \tag{2}$$

where h_l and h_{l+1} are the input and output for the l^{th} unit respectively. $\mathcal{F}$ is the residual function with weight parameters W_l. f is typically the ReLU function (Nair and Hinton, 2010). g is set as identity function, i.e., $g(h_l) = h_l$. Such an identity connection guarantees the direct propagation of signals among different layers, thereby avoids gradient vanishing. The recent paper (Liao and Poggio, 2016) talks about the possibility of using shared weights in ResNet, similar to what RNN does.

3 Recurrent Residual Learning

The basic idea of recurrent residual learning is to force a direct information flow in different time steps of RNNs by identity (skip) connections. In this section, we introduce how to leverage residual learning to 1) directly construct recurrent neural network in subsection 3.1; 2) improve LSTM in subsection 3.2.

3.1 Recurrent Residual Networks (RRN)

The basic architecture of Recurrent Residual Network (**RRN** for short) is illustrated in Figure 1, in which orange arrows indicate the identity connections from each h_{t-1} to h_t, and blue arrows represent the recurrent transformations taking both h_t and x_t as input. Similar to equation (2), the recurrent transformation in RRN takes the following form (denoted as $h_t = RRN(x_t, h_{t-1})$ in the following sections):

$$h_t = f(g(h_{t-1}) + \mathcal{F}(h_{t-1}, x_t; W)), \tag{3}$$

where g is still the identity function s.t. $g(h_{t-1}) = h_{t-1}$, corresponding to the orange arrows in Figure 1. f is typically set as *tanh*. For function $\mathcal{F}$ with weight parameters W (corresponding to the blue arrows in Figure 1), inspired by the observation that higher recurrent depth tends to lead to better performances (Zhang et al., 2016), we impose K deep transformations in $\mathcal{F}$:

$$\begin{aligned} y_1^t &= \sigma(x_t W_1 + h_{t-1} U_1 + b_1) \\ y_2^t &= \sigma(x_t W_2 + y_1^t U_2 + b_2) \\ &\cdots \\ y_K^t &= \sigma(x_t W_K + y_{K-1}^t U_K + b_K) \\ \mathcal{F}(h_{t-1}, x_t) &= y_K^t \end{aligned} \tag{4}$$

where x_t is taken at every layer such that the input information is better captured, which works similarly to the mechanism of highway network (Srivastava et al., 2015). K is the recurrent depth defined in (Zhang et al., 2016). The weights W_m ($m \in \{1, \cdots, K\}$) are shared across different time steps t.

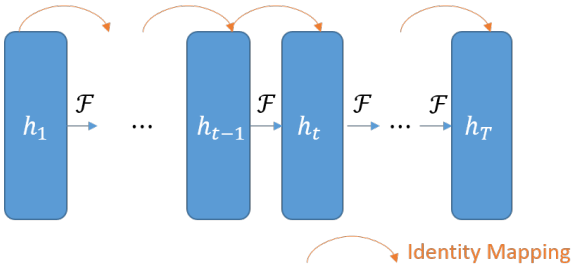


Figure 1: The basic structure of Recurrent Residual Networks.

RRN forces the direct propagation of hidden state signals between every two consecutive time steps with identity connections g. In addition, the multiple non-linear transformations in $\mathcal{F}$ guarantees its capability in modelling complicated recurrent relationship. In practice, we found that $K = 2$ yields fairly good performances, meanwhile leads to half of LSTM parameter size when model dimensions are the same.

3.2 Gated Residual RNN

Identity connections in ResNet are important for propagating the single input image information to higher layers of CNN. However, when it comes to sequence classification, the scenario is quite different in that there is a new input at every time step. Therefore, a forgetting mechanism to "forget" less critical historical information, as is employed in LSTM (controlled by the forget gate f_t), becomes necessary. On the other hand, while LSTM benefits from the flexible gating mechanism, its parametric nature brings optimization difficulties to cope with fairly long sequences, whose long range information dependencies could be better captured by identity connections.

Inspired by the success of the gating mechanism of LSTM and the residual connecting mechanism with enhanced information flow of ResNet, we further propose two Gated Residual Recurrent models leveraging the strengths of the two mechanisms.

3.2.1 Model 1: Skip-Connected LSTM (SC-LSTM)

Skip-Connected LSTM (**SC-LSTM** for short) introduces identity connections into standard LSTM to enhance the information flow. Note that in Figure 1, a straightforward approach is to replace $\mathcal{F}$ with an LSTM unit. However, our preliminary experiments do not achieve satisfactory results. Our conjecture is that identity connections between consecutive memory cells, which are already sufficient to maintain short-range memory, make the unregulated information flow overly strong, and thus compromise the merit of gating mechanism.

To reasonably enhance the information flow for LSTM network while keeping the advantage of gating mechanism, starting from equation (1), we propose to add skip connections between two LSTM hidden states with a wide range of distance L (e.g., $L = 20$), such that $\forall t = \{1, 1+L, 1+2L, \cdots, 1+\lfloor\frac{T-L-1}{L}\rfloor L\}$:

$$h_{t+L} = tanh(c_{t+L}) \otimes o_{t+L} + \alpha h_t \tag{5}$$

Here α is a scalar that can either be fixed as 1 (i.e., identity mapping) or be optimized during training process as a model parameter (i.e., parametric skip connection). We refer to these two variants as **SC-LSTM-I** and **SC-LSTM-P** respectively. Note that in SC-LSTM, the skip connections only exist in time steps $1, 1+L, 1+2L, \cdots, 1+\lfloor\frac{T-L-1}{L}\rfloor L$. The basic structure is shown in Figure 2.

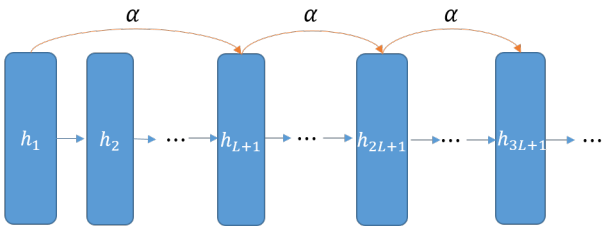


Figure 2: The basic structure of Skip-Connected LSTM.

3.2.2 Model 2: Hybrid Residual LSTM (HRL)

Since LSTM generates sequence representations out of flexible gating mechanism, and RRN generates representations with enhanced residual historical information, it is a natural extension to combine the two representations to form a signal that benefits from both mechanisms. We denote this model as Hybrid Residual LSTM (**HRL** for short).

In HRL, two independent signals, h_t^{LSTM} generated by LSTM (equation (1)) and h_t^{RRN} generated by RRN (equation (3)), are propagated through LSTM and RRN respectively:

$$\begin{aligned} h_t^{LSTM}, c_t &= LSTM(x_t, h_{t-1}^{LSTM}, c_{t-1}) \\ h_t^{RRN} &= RRN(x_t, h_{t-1}^{RRN}) \end{aligned} \tag{6}$$

The final representation h_T^{HRL} is obtained by the mean pooling of the two "sub" hidden states:

$$h_T^{HRL} = \frac{1}{2}(h_T^{LSTM} + h_T^{RRN}) \quad (7)$$

h_T^{HRL} is then used for higher level tasks such as predicting the sequence label. Acting in this way, h_T^{HRL} contains both the *statically* forced and *dynamically* adjusted historical signals, which are respectively conveyed by h_t^{RRN} and h_t^{LSTM}.

4 Experiments

We conduct comprehensive empirical analysis on sequence classification tasks. Listed in the ascending order of average sequence lengths, several public datasets we use include:

1. **AG's news corpus**[1],a news article corpus with categorized articles from more than $2,000$ news sources. We use the dataset with 4 largest classes constructed in (Zhang et al., 2015).

2. **IMDB movie review dataset**[2], a binary sentiment classification dataset consisting of movie review comments with positive/negative sentiment labels (Maas et al., 2011).

3. **20 Newsgroups** (**20NG** for short), an email collection dataset categorized into 20 news groups. Simiar to (Dai and Le, 2015), we use the post-processed version[3], in which attachments, PGP keys and some duplicates are removed.

4. **Permuted-MNIST** (**P-MNIST** for short). Following (Le et al., 2015; Arjovsky et al., 2015), we shuffle pixels of each MNIST image (LeCun et al., 1998) with a fixed random permutation, and feed all pixels sequentially into recurrent network to predict the image label. Permuted-MNIST is assumed to be a good testbed for measuring the ability of modeling very long range dependencies (Arjovsky et al., 2015).

[1] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[2] http://ai.stanford.edu/~amaas/data/sentiment/

[3] http://ana.cachopo.org/datasets-for-single-label-text-categorization

Detailed statistics of each dataset are listed in Table 1. For all the text datasets, we take every word as input and feed word embedding vectors pre-trained by Word2Vec (Mikolov et al., 2013) on Wikipedia into the recurrent neural network. The top most frequent words with 95% total frequency coverage are kept, while others are replaced by the token "UNK". We use the standard training/test split along with all these datasets and randomly pick 15% of training set as dev set, based on which we perform early stopping and for all models tune hyper-parameters such as dropout ratio (on non-recurrent layers) (Zaremba et al., 2014), gradient clipping value (Pascanu et al., 2013) and the skip connection length L for SC-LSTM (cf. equation (5)). The last hidden states of recurrent networks are put into logistic regression classifiers for label predictions. We use Adadelta (Zeiler, 2012) to perform parameter optimization. All our implementations are based on Theano (Theano Development Team, 2016) and run on one K40 GPU. All the source codes and datasets can be downloaded at `https://publish.illinois.edu/yirenwang/emnlp16source/`.

We compare our proposed models mainly with the state-of-art standard LSTM RNN. In addition, to fully demonstrate the effects of residual learning in our HRL model, we employ another hybrid model as baseline, which combines LSTM and GRU (Cho et al., 2014), another state-of-art RNN variant, in a similar way as HRL. We use **LSTM+GRU** to denote such a baseline. The model sizes (word embedding size $\times$ hidden state size) configurations used for each dataset are listed in Table 2. In Table 2, "Non-Hybrid" refers to LSTM, RRN and SC-LSTM models, while "Hybrid" refers to two methods that combines two basic models: HRL and LSTM+GRU. The model sizes of all hybrid models are smaller than the standard LSTM. All models have only one recurrent layer.

4.1 Experimental Results

All the classification accuracy numbers are listed in Table 3. From this table, we have the following observations and analysis:

1. RRN achieves similar performances to standard LSTM in all classification tasks with only

Dataset	Ave. Len	Max Len	#Classes	#Train : #Test
AG's News	34	211	4	$120,000 : 7,600$
IMDB	281	$2,956$	2	$25,000 : 25,000$
20NG	267	$11,924$	20	$11,293 : 7,528$
P-MNIST	784	784	10	$60,000 : 10,000$

Table 1: Classification Datasets.

	AG's News	*IMDB*	*20NG*	*P-MNIST*
Non-Hybird	256×512	256×512	500×768	1×100
Hybrid	256×384	256×384	256×512	1×80

Table 2: Model Sizes on Different Dataset.

Model/Task	*AG's News*	*IMDB*	*20NG*	*P-MNIST*
LSTM	91.76%	88.88%	79.21%	90.64%
RRN	91.19%	89.13%	79.76%	88.63%
SC-LSTM-P	92.01%	90.74%	**82.98%**	94.46%
SC-LSTM-I	**92.05%**	90.67%	81.85%	**94.80%**
LSTM+GRU	91.05%	89.23%	80.12%	90.28%
HRL	91.90%	**90.92%**	81.73%	90.33%

Table 3: Classification Results (Test Accuracy).

half of the model parameters, indicating that residual network structure, with connecting mechanism to enhance the information flow, is also an effective approach for sequence learning. However, the fact that it fails to significantly outperform other models (as it does in image classification) implies that forgetting mechanism is desired in recurrent structures to handle multiple inputs.

2. Skip-Connected LSTM performs much better than standard LSTM. For tasks with shorter sequences such as AG's News, the improvement is limited. However, the improvements get more significant with the growth of sequence lengths among different datasets[4], and the performance is particularly good in P-MNIST with very long sequences. This reveals the importance of skip connections in carrying on historical information through a long range of time steps, and demonstrates the effectiveness of our approach that adopts the residual connecting mechanism to improve LSTM's capability of handling long-term dependency. Furthermore, SC-LSTM is robust with different hyperparameter values: we test $L = 10, 20, 50, 75$ in P-MNIST and find the performance is not sensitive w.r.t. these L values.

3. HRL also outperforms standard LSTM with fewer model parameters[5]. In comparison, the hybrid model of LSTM+GRU cannot achieve such accuracy as HRL. As we expected, the additional long range historical information propagated by RRN is proved to be good assistance to standard LSTM.

5 Conclusion

In this paper, we explore the possibility of leveraging residual network to improve the performance of LSTM RNN. We show that direct adaptation of ResNet performs well in sequence classification. In addition, when combined with the gating mechanism in LSTM, residual learning significantly improve LSTM's performance. As to future work, we plan to apply residual learning to other sequence tasks such as language modeling, and RNN based neural machine translation (Sutskever et al., 2014) (Cho et al., 2014).

[4]t-test on SC-LSTM-P and SC-LSTM-I with $p\_value < 0.001$.

[5]t-test on HRL with $p\_value < 0.001$.

References

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2015. Unitary evolution recurrent neural networks. *arXiv preprint arXiv:1511.06464*.

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

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3061–3069.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

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

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jrgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. 2007. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352.

César Laurent, Gabriel Pereyra, Philémon Brakel, Ying Zhang, and Yoshua Bengio. 2015. Batch normalized recurrent neural networks. *arXiv preprint arXiv:1510.01378*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Qianli Liao and Tomaso Poggio. 2016. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June. Association for Computational Linguistics.

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

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2368–2376.

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

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan Salakhutdinov, and Yoshua Bengio. 2016. Architectural complexity measures of recurrent neural networks. *arXiv preprint arXiv:1602.08210*.

Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling

Ehsan Shareghi,[♣] Trevor Cohn[♠] and Gholamreza Haffari[♣]
[♣] Faculty of Information Technology, Monash University
[♠] Computing and Information Systems, The University of Melbourne
first.last@{monash.edu,unimelb.edu.au}

Abstract

In this work we present a generalisation of the Modified Kneser-Ney interpolative smoothing for richer smoothing via additional discount parameters. We provide mathematical underpinning for the estimator of the new discount parameters, and showcase the utility of our rich MKN language models on several European languages. We further explore the interdependency among the training data size, language model order, and number of discount parameters. Our empirical results illustrate that larger number of discount parameters, i) allows for better allocation of mass in the smoothing process, particularly on small data regime where statistical sparsity is severe, and ii) leads to significant reduction in perplexity, particularly for out-of-domain test sets which introduce higher ratio of out-of-vocabulary words.[1]

1 Introduction

Probabilistic language models (LMs) are the core of many natural language processing tasks, such as machine translation and automatic speech recognition. m-gram models, the corner stone of language modeling, decompose the probability of an utterance into conditional probabilities of words given a fixed-length context. Due to sparsity of the events in natural language, smoothing techniques are critical for generalisation beyond the training text when estimating the parameters of m-gram LMs. This is particularly important when the training text is small, e.g. building language models for translation or speech recognition in low-resource languages.

A widely used and successful smoothing method is interpolated Modified Kneser-Ney (MKN) (Chen and Goodman, 1999). This method uses a linear interpolation of higher and lower order m-gram probabilities by preserving probability mass via absolute discounting. In this paper, we extend MKN by introducing additional discount parameters, leading to a richer smoothing scheme. This is particularly important when statistical sparsity is more severe, i.e., in building high-order LMs on small data, or when out-of-domain test sets are used.

Previous research in MKN language modeling, and more generally m-gram models, has mainly dedicated efforts to make them faster and more compact (Stolcke et al., 2011; Heafield, 2011; Shareghi et al., 2015) using advanced data structures such as succinct suffix trees. An exception is Hierarchical Pitman-Yor Process LMs (Teh, 2006a; Teh, 2006b) providing a rich Bayesian smoothing scheme, for which Kneser-Ney smoothing corresponds to an approximate inference method. Inspired by this work, we directly enrich MKN smoothing realising some of the reductions while remaining more efficient in learning and inference.

We provide estimators for our additional discount parameters by extending the discount bounds in MKN. We empirically analyze our enriched MKN LMs on several European languages in in- and out-of-domain settings. The results show that our discounting mechanism significantly improves the perplexity compared to MKN and offers a more elegant

[1] For the implementation see: https://github.com/eehsan/cstlm

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 944–949,
Austin, Texas, November 1-5, 2016.

way of dealing with out-of-vocabulary (OOV) words and domain mismatch.

2 Enriched Modified Kneser-Ney

Interpolative Modified Kneser-Ney (MKN) (Chen and Goodman, 1999) smoothing is widely accepted as a state-of-the-art technique and is implemented in leading LM toolkits, e.g., SRILM (Stolcke, 2002) and KenLM (Heafield, 2011).

MKN uses lower order k-gram probabilities to smooth higher order probabilities. $P(w|\mathbf{u})$ is defined as,

$$\frac{c(\mathbf{u}w) - \mathbb{D}^m(c(\mathbf{u}w))}{c(\mathbf{u})} + \frac{\gamma(\mathbf{u})}{c(\mathbf{u})} \times \bar{P}(w|\pi(\mathbf{u}))$$

where $c(\mathbf{u})$ is the frequency of the pattern $\mathbf{u}$, $\gamma(.)$ is a constant ensuring the distribution sums to one, and $\bar{P}(w|\pi(\mathbf{u}))$ is the smoothed probability computed recursively based on a similar formula[2] conditioned on the suffix of the pattern $\mathbf{u}$ denoted by $\pi(\mathbf{u})$. Of particular interest are the discount parameters $\mathbb{D}^m(.)$ which remove some probability mass from the maximum likelihood estimate for each event which is redistributed over the smoothing distribution. The discounts are estimated as

$$\mathbb{D}^m(i) = \begin{cases} 0, & \text{if } i = 0 \\ 1 - 2\frac{n_2[m]}{n_1[m]}\frac{n_1[m]}{n_1[m]+2n_2[m]}, & \text{if } i = 1 \\ 2 - 3\frac{n_3[m]}{n_2[m]}\frac{n_1[m]}{n_1[m]+2n_2[m]}, & \text{if } i = 2 \\ 3 - 4\frac{n_4[m]}{n_3[m]}\cdot\frac{n_1[m]}{n_1[m]+2n_2[m]}, & \text{if } i \geq 3 \end{cases}$$

where $n_i(m)$ is the number of unique m-grams[3] of frequency i. This effectively leads to three discount parameters $\{\mathbb{D}^m(1), \mathbb{D}^m(2), \mathbb{D}^m(3+)\}$ for the distributions on a particular context length, m.

2.1 Generalised MKN

Ney et al. (1994) characterized the data sparsity using the following empirical inequalities,

$$3n_3[m] < 2n_2[m] < n_1[m] \quad \text{for} \quad m \leq 3$$

It can be shown (see Appendix A) that these empirical inequalities can be extended to higher frequencies and larger contexts $m > 3$,

$$\begin{aligned}(N-m)n_{N-m}[m] < ... < 2n_2[m] \\ < n_1[m] < \sum_{i>0} n_i[m] \ll n_0[m] < \sigma^m\end{aligned}$$

where σ^m is the possible number of m-grams over a vocabulary of size σ, $n_0[m]$ is the number of m-grams that never occurred, and $\sum_{i>0} n_i[m]$ is the number of m-grams observed in the training data.

We use these inequalities to extend the discount depth of MKN, resulting in new discount parameters. The additional discount parameters increase the flexibility of the model in altering a wider range of raw counts, resulting in a more elegant way of assigning the mass in the smoothing process. In our experiments, we set the number of discounts to 10 for all the levels of the hierarchy, (compare this to these in MKN).[4] This results in the following estimators for the discounts,

$$\mathbb{D}^m(i) = \begin{cases} 0, & \text{if } i = 0 \\ i - (i+1)\frac{n_{i+1}[m]}{n_i[m]}\frac{n_1[m]}{n_1[m]+2n_2[m]}, & \text{if } i < 10 \\ 10 - 11\frac{n_{11}[m]}{n_{10}[m]}\cdot\frac{n_1[m]}{n_1[m]+2n_2[m]}, & \text{if } i \geq 10 \end{cases}$$

It can be shown that the above estimators for our discount parameters are derived by maximizing a lower bound on the leave-one-out likelihood of the training set, following (Ney et al., 1994; Chen and Goodman, 1999) (see Appendix B for the proof sketch).

3 Experiments

We compare the effect of using different numbers of discount parameters on perplexity using the Finnish (FI), Spanish (ES), German (DE), English (EN) portions of the Europarl v7 (Koehn, 2005) corpus. For each language we excluded the first 10K sentences and used it as the in-domain test set (denoted as EU), skipped the second 10K sentences, and used the rest as the training set. The data was tokenized, sentence split, and the XML markup discarded. We tested the effect of domain mismatch, under two settings for out-of-domain test sets: i) *mild* using the Spanish section of news-test 2013, the German, English sections of news-test 2014, and the Finnish section

[2]Note that in all but the top layer of the hierarchy, *continuation counts*, which count the number of unique contexts, are used in place of the frequency counts (Chen and Goodman, 1999).

[3]Continuation counts are used for the lower layers.

[4]We have selected the value of 10 arbitrarily; however our approach can be used with larger number of discount parameters, with the caveat that we would need to handle sparse counts in the higher orders.

Training	size (M) tokens	sents	Test	size (K) tokens	sents	OOV%	Perplexity MKN ($D_{1...3}$) $m=2$	$m=5$	$m=10$	MKN ($D_{[1...4]}$) $m=2$	$m=5$	$m=10$	MKN ($D_{[1...10]}$) $m=2$	$m=5$	$m=10$
			NT	19.8	3	9.2	6536.6	5900.3	5897.3	6451.3	5827.6	5824.6	6154.4	5575.0	5572.5
FI	46.5	2.2	EU	197.3	10	6.1	390.7	287.4	286.8	390.7	287.3	286.6	390.4	287.3	286.8
			TW	10.9	1.3	52.1	57 825.1	51 744.1	51 740.1	55 550.2	49 884.2	49 881.3	47 696.2	43 277.3	43 275.5
			NT	70.7	3	9.1	565.6	431.5	429.4	560.0	425.5	423.5	541.5	409.0	407.3
ES	68.0	2.2	EU	281.5	10	2.4	92.7	51.5	51.1	92.8	51.5	51.1	92.8	51.4	51.0
			TW	3141.3	293	78.5	17 804.2	14 062.7	14 027.1	17 121.4	13 487.4	13 454.1	14 915.7	11 832.1	11 807.2
			NT	64.5	3	18.7	2190.7	1784.6	1781.8	2158.9	1755.8	1753.2	2065.3	1680.6	1678.3
DE	61.2	2.3	EU	244.0	10	4.6	156.9	91.7	91.2	156.9	91.6	91.2	156.4	91.7	91.2
			MED	317.7	10	59.8	5135.7	4232.4	4226.7	5007.5	4123.0	4117.5	4636.0	3831.2	3826.6
			NT	69.5	3	5.5	1089.2	875.0	872.2	1071.1	857.2	854.4	1011.5	806.7	804.4
EN	67.5	2.2	EU	274.9	10	1.7	90.1	48.4	48.1	90.1	48.3	48.0	90.5	48.3	48.0
			MED	405.9	10	44.1	2319.7	1947.9	1942.5	2261.6	1893.3	1888.2	2071.9	1734.9	1730.8

Table 1: Perplexity for various m-gram orders $m \in 2, 3, 10$ and training languages from Europarl, using different numbers of discount parameters for MKN. **MKN ($D_{[1...3]}$)**, **MKN ($D_{[1...4]}$)**, **MKN ($D_{[1...10]}$)** represent vanilla MKN, MKN with 1 more discounts, and MKN with 7 more discount parameters, respectively. Test sets sources EU, NT, TW, MED are Europarl, news-test, Twitter, and medical patent descriptions, respectively. OOV is reported as the ratio $\frac{|\{OOV \in \text{test-set}\}|}{|\{w \in \text{test-set}\}|}$.

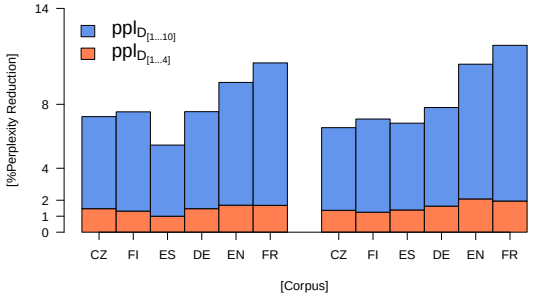


Figure 1: Percentage of perplexity reduction for pplx$_{D_{[1...4]}}$ and pplx$_{D_{[1...10]}}$ compared with pplx$_{D_{[1...3]}}$ on different training corpora (Europarl CZ, FI, ES, DE, EN, FR) and on news-test sets (NT) for $m = 2$ (left), and $m = 10$ (right).

of news-test 2015 (all denoted as NT)[5], and ii) *extreme* using a 24 hour period of streamed Finnish, and Spanish tweets[6] (denoted as TW), and the German and English sections of the patent description of medical translation task[7] (denoted as MED). See Table 1 for statistics of the training and test sets.

3.1 Perplexity

Table 1 shows substantial reduction in perplexity on all languages for out-of-domain test sets when expanding the number of discount parameters from 3 in vanilla MKN to 4 and 10. Consider the English news-test (NT), in which even for a 2-gram language model a single extra discount parameter ($m = 2$, $D_{[1...4]}$) improves the perplexity by 18 points and this improvement quadruples to 77 points when using 10 discounts ($m = 2$, $D_{[1...10]}$). This effect is consistent across the Europarl corpora, and for all LM orders. We observe a substantial improvements even for $m = 10$-gram models (see Figure 1). On the medical test set which has 9 times higher OOV ratio, the perplexity reduction shows a similar trend. However, these reductions vanish when an in-domain test set is used. Note that we use the same treatment of OOV words for computing the perplexities which is used in KenLM (Heafield, 2013).

3.2 Analysis

Out-of-domain and Out-of-vocabulary We selected the Finnish language for which the number and ratio of OOVs are close on its out-of-domain and in-domain test sets (NT and EU), while showing substantial reduction in perplexity on out-of-domain test set, see FI bars on Figure 1. Figure 2 (left), shows the full perplexity results for Finnish for vanilla MKN, and our extensions when tested on in-domain (EU) and out-of-domain (NT) test sets. The discount plot, Figure 2 (middle) illustrates the behaviour of the various discount parameters. We also measured the average hit length for queries by varying m on in-domain and out-of-domain test sets. As illustrated in Figure 2 (right) the in-domain test set allows for longer matches to the training data as

[5] http://www.statmt.org/{wmt13,14,15}/test.tgz
[6] Streamed via Twitter API on 17/05/2016.
[7] http://www.statmt.org/wmt14/medical-task/

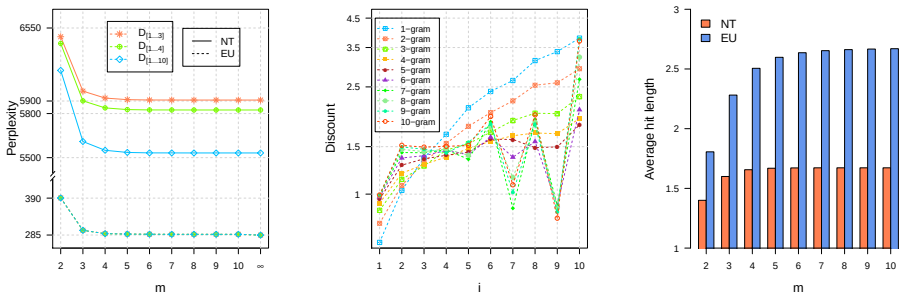


Figure 2: Statistics for the Finnish section of Europarl. The left plot illustrates the perplexity when tested on an out-of-domain (NT) and in-domain (EU) test sets varying LM order, m. The middle plot shows the discount parameters $D_{i\in[1...10]}$ for different m-gram orders. The right plot correspond to average hit length on EU and NT test sets.

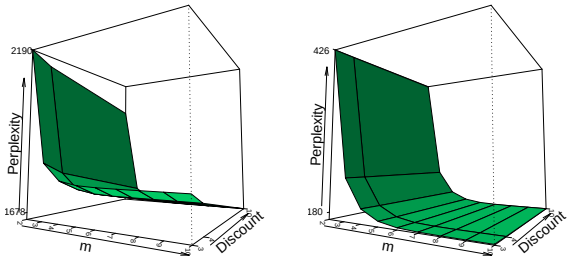


Figure 3: Perplexity (z-axis) vs. $m \in [2...10]$ (x-axis) vs. number of discounts $D_{i\in3,4,10}$ (y-axis) for German language trained on Europarl (left), and CommonCrawl2014 (right) and tested on news-test. Arrows show the direction of the increase.

m grows. This indicates that having more discount parameters is not only useful for test sets with extremely high number of OOV, but also allows for a more elegant way of assigning mass in the smoothing process when there is a domain mismatch.

Interdependency of m, data size, and discounts To explore the correlation between these factors we selected the German and investigated this correlation on two different training data sizes: Europarl (61M words), and CommonCrawl 2014 (984M words). Figure 3 illustrates the correlation between these factors using the same test set but with small and large training sets. Considering the slopes of the surfaces indicates that the small training data regime (left) which has higher sparsity, and more OOV in the test time benefits substantially from the more accurate discounting compared to the large training set (right) in which the gain from discounting is slight.[8]

[8]Nonetheless, the improvement in perplexity consistently grows with introducing more discount parameters even under the large training data regime, which suggests that more discount parameters, e.g., up to D_{30}, may be required for larger training corpus to reflect the fact that even an event with frequency of 30 might be considered rare in a corpus of nearly 1 billion words.

4 Conclusions

In this work we proposed a generalisation of Modified Kneser-Ney interpolative language modeling by introducing new discount parameters. We provide the mathematical proof for the discount bounds used in Modified Kneser-Ney and extend it further and illustrate the impact of our extension empirically on different Europarl languages using in-domain and out-of-domain test sets.

The empirical results on various training and test sets show that our proposed approach allows for a more elegant way of treating OOVs and mass assignments in interpolative smoothing. In future work, we will integrate our language model into the Moses machine translation pipeline to intrinsically measure its impact on translation qualities, which is of particular use for out-of-domain scenario.

Acknowledgements

This research was supported by the National ICT Australia (NICTA) and Australian Research Council Future Fellowship (project number FT130101105). This work was done when Ehsan Shareghi was an intern at IBM Research Australia.

A. Inequalities

We prove that these inequalities hold in expectation by making the reasonable assumption that events in

the natural language follow the power law (Clauset et al., 2009), $p\big(C(\mathbf{u}) = f\big) \propto f^{-1-\frac{1}{s_m}}$, where s_m is the parameter of the distribution, and $C(\mathbf{u})$ is the random variable denoting the frequency of the m-grams pattern $\mathbf{u}$. We now compute the expected number of unique patterns having a specific frequency $E[n_i[m]]$. Corresponding to each m-grams pattern $\mathbf{u}$, let us define a random variable $X_\mathbf{u}$ which is 1 if the frequency of $\mathbf{u}$ is i and zero otherwise. It is not hard to see that $n_i[m] = \sum_\mathbf{u} X_\mathbf{u}$, and

$$\begin{aligned}E\big[n_i[m]\big] &= E\big[\sum_\mathbf{u} X_\mathbf{u}\big] = \sum_\mathbf{u} E[X_\mathbf{u}] = \sigma^m E[X_\mathbf{u}] \\ &= \sigma^m\Big(p\big(C(\mathbf{u}) = i\big) \times 1 + p\big(C(\mathbf{u}) \neq i\big) \times 0\Big) \\ &\propto \sigma^m i^{-1-\frac{1}{s_m}}.\end{aligned}$$

We can verify that

$$\begin{aligned}(i+1)E\big[n_{i+1}[m]\big] < iE\big[n_i[m]\big] \Leftrightarrow \\ (i+1)\sigma^m(i+1)^{-1-\frac{1}{s_m}} < i\sigma^m i^{-1-\frac{1}{s_m}} \Leftrightarrow \\ i^{\frac{1}{s_m}} < (i+1)^{\frac{1}{s_m}}.\end{aligned}$$

which completes the proof of the inequalities.

B. Discount bounds proof sketch

The leave-one-out (leaving those m-grams which occurred only once) log-likelihood function of the interpolative smoothing is lower bounded by backoff model's (Ney et al., 1994), hence the estimated discounts for later can be considered as an approximation for the discounts of the former. Consider a backoff model with absolute discounting parameter D, were $P(w_i|w_{i-m+1}^{i-1})$ is defined as:

$$\begin{cases} \frac{c(w_{i-m+1}^i)-D}{c(w_{i-m+1}^{i-1})} & \text{if } c(w_{i-m+1}^i) > 0 \\ \frac{Dn_{1+}(w_{i-m+1}^{i-1}\,\bullet)}{c(w_{i-m+1}^{i-1})}\bar{P}(w^i|w_{i-m+2}^{i-1}) & \text{if } c(w_{i-m+1}^i) = 0 \end{cases}$$

where $n_{1+}(w_{i-m+1}^{i-1}\,\bullet)$ is the number of unique right contexts for the w_{i-m+1}^{i-1} pattern. Assume that for any choice of $0 < D < 1$ we can define $\bar{P}$ such that $P(w_i|w_{i_m+1}^{i-1})$ sums to 1. For readability we use the $\lambda(w_{i-m+1}^{i-1}) = \frac{n_{1+}(w_{i-m+1}^{i-1}\,\bullet)}{c(w_{i-m+1}^{i-1})-1}$ replacement. Following (Chen and Goodman, 1999), rewriting the leave-one-out log-likelihood for KN (Ney et al., 1994) to include more discounts (in this proof up to D_4), results in:

$$\begin{aligned}&\sum_{\substack{w_{i-m+1}^i \\ c(w_{i-m+1}^i)>4}} c(w_{i-m+1}^i)\log\frac{c(w_{i-m+1}^i)-1-D_4}{c(w_{i-m+1}^{i-1})-1} + \\ &\sum_{j=2}^{4}\Big(\sum_{\substack{w_{i-m+1}^i \\ c(w_{i-m+1}^i)=j}} c(w_{i-m+1}^i)\log\frac{c(w_{i-m+1}^i)-1-D_{j-1}}{c(w_{i-m+1}^{i-1})-1}\Big) + \\ &\sum_{\substack{w_{i-m+1}^i \\ c(w_{i-m+1}^i)=1}}\Big(c(w_{i-m+1}^i)\log(\sum_{j=1}^{4} n_j[m]D_j)\lambda(w_{i-m+1}^{i-1})\bar{P}\Big)\end{aligned}$$

which can be simplified to,

$$\begin{aligned}&\sum_{\substack{w_{i-m+1}^i \\ c(w_{i-m+1}^i)>4}} c(w_{i-m+1}^i)\log(c(w_{i-m+1}^i)-1-D_4) + \\ &\sum_{j=2}^{4}\Big(jn_j[m]\log(j-1-D_{j-1})\Big) + \\ &n_1[m]\log(\sum_{j=1}^{4} n_j[m]D_j) + \text{const}\end{aligned}$$

To find the optimal D_1, D_2, D_3, D_4 we set the partial derivatives to zero. For D_3,

$$\begin{aligned}&\frac{\partial}{\partial D_3} = n_1[m]\frac{n_3[m]}{\sum_{j=1}^4 n_j[m]D_j} - \frac{4n_4[m]}{3-D_3} = 0 \Rightarrow \\ &n_1[m]n_3[m](3-D_3) = 4n_4[m]\sum_{j=1}^{4} n_j[m]D_j \Rightarrow \\ &3n_1[m]n_3[m] - D_3 n_1[m]n_3[m] - 4n_4[m]n_1[m]D_1 > 0 \\ &\Rightarrow 3 - 4\frac{n_4[m]}{n_3[m]}D_1 > D_3 \quad \blacksquare\end{aligned}$$

And after taking $c(w_{i-m+1}^i) = 5$ out of the summation, for D_4:

$$\begin{aligned}&\frac{\partial}{\partial D_4} = \sum_{c(w_{i-m+1}^i)>5}\frac{-c(w_{i-m+1}^i)}{c(w_{i-m+1}^i)-1-D} - \frac{5n_5[m]}{4-D_4} \\ &+ n_1[m]\frac{n_4[m]}{\sum_{j=1}^4 n_j[m]D_j} = 0 \Rightarrow -\frac{5n_5[m]}{4-D_4} \\ &+ n_1[m]\frac{n_4[m]}{\sum_{j=1}^4 n_j[m]D_j} > 0 \Rightarrow n_1[m]n_4[m](4-D_4) \\ &> 5n_5[m]\sum_{j=1}^{4} n_j[m]D_j \Rightarrow 4 - 5\frac{n_5[m]}{n_4[m]}D_1 > D_4 \quad \blacksquare\end{aligned}$$

References

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Workshop on Statistical Machine Translation*.

Kenneth Heafield. 2013. *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation summit*.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38.

Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2015. Compact, efficient and unlimited capacity: Language modeling with compressed suffix trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*.

Andreas Stolcke. 2002. SRILM–an extensible language modeling toolkit. In *Proceedings of the International Conference of Spoken Language Processing*.

Yee Whye Teh. 2006a. A Bayesian interpretation of interpolated Kneser-Ney. Technical report, NUS School of Computing.

Yee Whye Teh. 2006b. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

A General Regularization Framework for Domain Adaptation

Wei Lu[1] and **Hai Leong Chieu**[2] and **Jonathan Löfgren**[3]
[1]Singapore University of Technology and Design
[2]DSO National Laboratories
[3]Uppsala University
luwei@sutd.edu.sg, chaileon@dso.org.sg, lofgren021@gmail.com

Abstract

We propose a domain adaptation framework, and formally prove that it generalizes the feature augmentation technique in (Daumé III, 2007) and the multi-task regularization framework in (Evgeniou and Pontil, 2004). We show that our framework is strictly more general than these approaches and allows practitioners to tune hyper-parameters to encourage transfer between close domains and avoid negative transfer between distant ones.

1 Introduction

Domain adaptation (DA) is an important problem that has received substantial attention in natural language processing (Blitzer et al., 2006; Daumé III, 2007; Finkel and Manning, 2009; Daumé III et al., 2010). In this paper, we propose a novel regularization framework which allows DA practitioners to tune hyper-parameters to encourage transfer between close domains, and avoid negative transfer (Rosenstein et al., 2005) between distant ones. In our framework, model parameters in multiple domains are learned jointly and constrained to remain close to one another. In the transfer learning taxonomy (Pan and Yang, 2010), our framework falls under the parameter-transfer category for multi-task inductive learning. We show that our framework generalizes the frustratingly easy domain adaptation (FEDA) in Daumé III (2007), Finkel and Manning (2009), and the regularised multi-task learning of Evgeniou and Pontil (2004). At the same time, it provides us with hyper-parameters to control the amount of transfer between domains.

2 Domain Adaptation Framework

Given labeled data from N domains, $\mathbf{D}^1, \ldots, \mathbf{D}^N$, traditional machine learning maximizes the following objective function for each domain $\mathbf{D}^i$:

$$\mathcal{O}(\mathbf{D}^i; \mathbf{w}_i) = \mathcal{L}_i(\mathbf{D}^i; \mathbf{w}_i) - \lambda_i ||\mathbf{w}_i||^2, \quad (1)$$

and we maximize $\mathcal{L}_i$ by tuning the parameter vector $\mathbf{w}_i$. For example, $\mathcal{L}_i$ can be the log-likelihood or the negative hinge loss. The term $\lambda_i ||\mathbf{w}_i||^2$ is the L_2-regularization term where λ_i is a positive scalar. In our framework, we propose to maximize

$$\sum_{i=1}^{N} \mathcal{L}_i(\mathbf{D}^i; \mathbf{w}_i) - \sum_{i=1}^{N} \eta_{0,i} ||\mathbf{w}_i||^2 - \sum\nolimits_{1 \le j < k \le N} \eta_{j,k} ||\mathbf{w}_j - \mathbf{w}_k||^2, \quad (2)$$

where $\eta_{j,k}$ are parameters controlling the transfer between domains. In the next sections, we show how our framework generalizes existing works.

2.1 Frustratingly Easy DA

The FEDA approach was introduced by Daumé III (2007) and later formalized by Finkel and Manning (2009) within a hierarchical Bayesian DA framework. While simple, the approach has often been shown to be effective. In this section, we show that our framework generalizes the FEDA approach.

The FEDA approach defines a new augmented feature space by duplicating each feature in $\mathbf{D}^i$ to a "general" domain. Therefore each parameter in $\mathbf{w}_i$ has a corresponding parameter in $\mathbf{w}_0$, and:

$$\mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i, \mathbf{w}_0) = \mathcal{L}_i(\mathbf{D}^i; \mathbf{w}_i + \mathbf{w}_0) \quad (3)$$

This directly leads to the following remark:

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 950–954,
Austin, Texas, November 1-5, 2016.

Remark For all i, for any $\mathbf{w}_i, \mathbf{w}_0, \mathbf{d} \in \mathbf{R}^m$:

$$\mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i + \mathbf{d}, \mathbf{w}_0 - \mathbf{d}) = \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i, \mathbf{w}_0)$$

The complete objective function involving N ($N \geq 2$) domains is defined as follows:

$$\mathcal{O}'(\mathbf{D}; , \mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_N) = \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i, \mathbf{w}_0) - \sum_{i=0}^{N} \lambda_i ||\mathbf{w}_i||^2$$

We first prove the following relation:

Lemma 2.1 *Assume*

$$(\mathbf{w}_0^*, ..., \mathbf{w}_N^*) = \arg\max_{\mathbf{w}_1, ..., \mathbf{w}_N, \mathbf{w}_0} \left[\sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i, \mathbf{w}_0) - \left(\lambda_0 ||\mathbf{w}_0||^2 + \sum_{i=1}^{N} \lambda_i ||\mathbf{w}_i||^2 \right) \right],$$

where $\lambda_0, \lambda_1, \ldots, \lambda_N > 0$*, then:*

$$\lambda_0 \mathbf{w}_0^* = \sum_{i=1}^{N} \lambda_i \mathbf{w}_i^* \tag{4}$$

Proof Let's introduce the vector $\mathbf{d}$ as follows:

$$\mathbf{d} = \frac{1}{\sum_{i=0}^{N} \lambda_i} \left(\lambda_0 \mathbf{w}_0^* - \sum_{i=1}^{N} \lambda_i \mathbf{w}_i^* \right) \tag{5}$$

Denote $(\mathbf{w}'_0, \ldots, \mathbf{w}'_N)$ such that $\forall\ 0 \leq i \leq N$,

$$\mathbf{w}'_i = \mathbf{w}_i^* + \mathbf{d}, \text{ and } \mathbf{w}'_0 = \mathbf{w}_0^* - \mathbf{d}.$$

Based on the remark, $\mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}'_i, \mathbf{w}'_0) = \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i^*, \mathbf{w}_0^*)$. Let $\Delta = \mathcal{O}'(\mathbf{D}; \mathbf{w}'_0, \ldots, \mathbf{w}'_N) - \mathcal{O}'(\mathbf{D}; \mathbf{w}_0^*, \ldots, \mathbf{w}_N^*)$. Since $(\mathbf{w}_0^*, \ldots, \mathbf{w}_N^*)$ is optimal, $\Delta \leq 0$. Moreover,

$$\begin{aligned}
\Delta &= \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}'_i, \mathbf{w}'_0) - \sum_{i=0}^{N} \lambda_i ||\mathbf{w}'_i||^2 \\
&\quad - \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i; \mathbf{w}_i^*, \mathbf{w}_0^*) + \sum_{i=0}^{N} \lambda_i ||\mathbf{w}_i^*||^2 \\
&= \lambda_0 ||\mathbf{w}_0^*||^2 - \lambda_0 ||\mathbf{w}_0^* - \mathbf{d}||^2 \\
&\quad + \sum_{i=1}^{N} \lambda_i ||\mathbf{w}_i^*||^2 - \sum_{i=1}^{N} \lambda_i ||\mathbf{w}_i^* + \mathbf{d}||^2 \\
&= -\left(\sum_{i=0}^{N} \lambda_i \right) ||\mathbf{d}||^2 + \\
&\quad 2\mathbf{d} \cdot \left(\lambda_0 \mathbf{w}_0^* - \sum_{i=1}^{N} \lambda_i \mathbf{w}_i^* \right) \\
&= -\left(\sum_{i=0}^{N} \lambda_i \right) ||\mathbf{d}||^2 + 2\mathbf{d} \cdot \left(\sum_{i=0}^{N} \lambda_i \right) \mathbf{d} \\
&= \left(\sum_{i=0}^{N} \lambda_i \right) ||\mathbf{d}||^2 \geq 0
\end{aligned}$$

Hence, $\Delta = 0$ implying $||\mathbf{d}|| = 0$ and so $\mathbf{d} = \mathbf{0}$. From the definition of $\mathbf{d}$, Equation 4 holds. ∎

Next we state the following lemma (see supplementary material for the proof).

Lemma 2.2 *For any vectors* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \in \mathbf{R}^m$*, any scalars* $\lambda_0, \lambda_1, \ldots, \lambda_N \in \mathbf{R}^+$*, let* $\mathbf{v}_0 = (\sum_{i=1}^{N} \lambda_i \mathbf{v}_i)/\lambda_0$*, then the following always holds:*

$$\lambda_0 ||\mathbf{v}_0||^2 + \sum_{i=1}^{N} \lambda_i ||\mathbf{v}_i||^2 = \sum_{i=1}^{N} \eta_{0,i} ||\mathbf{v}_i + \mathbf{v}_0||^2 + \sum_{1 \leq j < k \leq N} \eta_{j,k} ||\mathbf{v}_j - \mathbf{v}_k||^2,$$

$$\text{where} \quad \eta_{i,j} = \frac{\lambda_i \lambda_j}{\sum_{l=0}^{N} \lambda_l},\ \forall\ 0 \leq i < j \leq N.$$

Now we state and prove the following theorem, which shows our framework generalizes FEDA.

Theorem 2.3 *For* $\lambda_0, \lambda_1, \ldots, \lambda_N \in \mathbf{R}^+$*, define*

$$\forall 0 \leq i < j \leq N, \quad \eta_{i,j} = \frac{\lambda_i \lambda_j}{\sum_{l=0}^{N} \lambda_l},$$

the following holds:

$$\max_{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_N,\mathbf{w}_0} \left[\sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i;\mathbf{w}_i,\mathbf{w}_0) - \left(\lambda_0||\mathbf{w}_0||^2 + \sum_{i=1}^{N}\lambda_i||\mathbf{w}_i||^2\right)\right]$$
$$= \max_{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_N} \left[\sum_{i=1}^{N} \mathcal{L}_i(\mathbf{D}^i;\mathbf{w}_i) - \left(\sum_{i=1}^{N}\eta_{0,i}||\mathbf{w}_i||^2 + \sum_{1\le j<k\le N}\eta_{j,k}||\mathbf{w}_j-\mathbf{w}_k||^2\right)\right]$$

Proof Let $(\mathbf{w}_0^*,\ldots,\mathbf{w}_N^*)$ be a solution to the first optimization problem. We have:

$$\mathbf{LHS} = \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i;\mathbf{w}_i^*,\mathbf{w}_0^*) - \left(\lambda_0||\mathbf{w}_0^*||^2 + \sum_{i=1}^{N}\lambda_i||\mathbf{w}_i^*||^2\right) \quad (6)$$

Lemma 2.1 gives $\mathbf{w}_0^* = \left(\sum_{i=1}^{N}\lambda_i\mathbf{w}_i^*\right)/\lambda_0$. Introduce $\mathbf{w}'_i = \mathbf{w}_i^* + \mathbf{w}_0^*$. Using Lemma 2.2, we have:

$$\mathbf{LHS} = \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i;\mathbf{w}_i^*,\mathbf{w}_0^*) - \left(\sum_{i=1}^{N}\eta_{0,i}||\mathbf{w}_i^*+\mathbf{w}_0^*||^2 + \sum_{1\le j<k\le N}\eta_{j,k}||\mathbf{w}_j^*-\mathbf{w}_k^*||^2\right)$$
$$= \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{D}^i;\mathbf{w}'_i) - \left(\sum_{i=1}^{N}\eta_{0,i}||\mathbf{w}'_i||^2 + \sum_{1\le j<k\le N}\eta_{j,k}||\mathbf{w}'_j-\mathbf{w}'_k||^2\right) \le \mathbf{RHS}$$

Now, let $(\mathbf{w}_1^*,\mathbf{w}_2^*,\ldots,\mathbf{w}_N^*)$ be an optimal solution to the second problem. Given the relation between $\eta_{i,j}$ and $\lambda_0,\lambda_1,\ldots,\lambda_N$, let $\mathbf{w}'_0 = \left(\sum_{i=1}^{N}\lambda_i\mathbf{w}_i^*\right)/\left(\sum_{l=0}^{N}\lambda_l\right)$, and $\mathbf{w}'_i = \mathbf{w}_i^* - \mathbf{w}'_0$. We show in the supplementary material that

$$\mathbf{w}'_0 = \frac{1}{\lambda_0}\left(\sum_{i=1}^{N}\lambda_i\mathbf{w}'_i\right) \quad (7)$$

Based on these and Lemma 2.2, we have:

$$\mathbf{RHS} = \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{D}^i;\mathbf{w}_i^*) - \left(\sum_{i=1}^{N}\eta_{0,i}||\mathbf{w}_i^*||^2 + \sum_{1\le j<k\le N}\eta_{j,k}||\mathbf{w}_j^*-\mathbf{w}_k^*||^2\right)$$
$$= \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{D}^i;\mathbf{w}'_i+\mathbf{w}'_0) - \left(\sum_{i=1}^{N}\eta_{0,i}||\mathbf{w}'_i+\mathbf{w}'_0||^2 + \sum_{1\le j<k\le N}\eta_{j,k}||\mathbf{w}'_j-\mathbf{w}'_k||^2\right)$$
$$= \sum_{i=1}^{N} \mathcal{L}'_i(\mathbf{D}^i;\mathbf{w}'_i,\mathbf{w}'_0) - \left(\lambda_0||\mathbf{w}'_0||^2 + \sum_{i=1}^{N}\lambda_i||\mathbf{w}'_i||^2\right) \le \mathbf{LHS}$$

Therefore we must have $\mathbf{LHS} = \mathbf{RHS}$. ∎

This formally shows that FEDA is equivalent to solving the objective function given in Equation 2. In this new optimization problem, if we drop the terms involving $\eta_{j,k}$ for $j \neq 0$, we have:

$$\sum_{i=1}^{N}\left(\mathcal{L}_i(\mathbf{D}^i;\mathbf{w}_i) - \eta_{0,i}||\mathbf{w}_i||^2\right) \quad (8)$$

This is learning without domain adaptation. The additional regularization terms allow us keep the parameters from different domains close to one other. In the special case with two domains, if we use the same λ for all regularization terms, we have the following corollary:

Corollary 2.4 *For any $\lambda > 0$:*

$$\max_{\mathbf{w}_1,\mathbf{w}_2,\mathbf{w}_0}\left[\mathcal{L}'_1(\mathbf{D}^1;\mathbf{w}_1,\mathbf{w}_0) + \mathcal{L}'_2(\mathbf{D}^2;\mathbf{w}_2,\mathbf{w}_0) - \lambda\left(||\mathbf{w}_1||^2 + ||\mathbf{w}_2||^2 + ||\mathbf{w}_0||^2\right)\right]$$
$$= \max_{\mathbf{w}_1,\mathbf{w}_2}\left[\mathcal{L}_1(\mathbf{D}^1;\mathbf{w}_1) + \mathcal{L}_2(\mathbf{D}^2;\mathbf{w}_2) - \frac{1}{3}\lambda\left(||\mathbf{w}_1||^2 + ||\mathbf{w}_2||^2 + ||\mathbf{w}_1-\mathbf{w}_2||^2\right)\right]$$

Hence, the FEDA feature augmentation technique indirectly introduces a regularization term that pushes the source and target parameters as close

as possible. This is related to the technique of Chelba and Acero (2006) where they regularize the model parameters for the target domain using the term $\lambda||\mathbf{w} - \mathbf{w}^s||$, where $\mathbf{w}^s$ is the parameter vector learned from the source domain. The difference here is, in their work the parameters for the source domain are learned first and then fixed. The relation between their work and the feature augmentation technique was also briefly discussed in the paper of Daumé III (2007). We formally showed a precise relation here in this paper.

2.2 Regularized Multi-task Learning

Evgeniou and Pontil (2004) proposed multi-task regularized learning using support vector machines (SVM). They decomposed the model weight vector as a sum of domain-specific vectors and a general vector, in much the same way as FEDA[1]. Hence, both Lemma 2.1 and Theorem 2.3 of this paper apply, and our framework also generalizes multi-task regularized learning.

3 Experimental Results

In this section we apply our framework to both structured and un-structured tasks. For structured prediction, we use the named-entity recognition (NER) ACE-2005 dataset with 7 classes and 6 domains. We apply the linear chain CRF (Lafferty et al., 2001), and show results using standard and softmax-margin CRF (SM-CRF) (Gimpel and Smith, 2010), with features consisting of word shape features, neighboring words, previous prediction and prefixes/suffixes. The second task is sentiment classification on the Amazon review data set (Blitzer et al., 2007) from 4 domains, labeled positive or negative. We apply logistic regression (LR) and SVM using unigram and bigram features. All the models used in this section are implemented on top of a common framework, which was also used to implement various structured prediction models previously (Lu, 2015; Lu and Roth, 2015; Muis and Lu, 2016). For each task we compare:

TGT Trained only on the specific domain data,
ALL Trained on the data from all domains,

[1]They proved in Lemma 2.1 in their paper a similar relationship to Equation 4, but their proof assumes a SVM framework, and that λ_1=λ_2=. . . =λ_N.

Model	Dom.	TGT	ALL	AUG	RF
CRF	bc	71.85	75.56	75.30	**76.48**
	bn	72.06	75.02	**75.17**	75.15
	cts	85.49	85.98	86.44	**86.70**
	nw	72.55	76.52	76.27	**76.61**
	un	67.09	72.99	72.90	**73.12**
	wl	64.38	69.66	69.46	**69.90**
	avg	72.24	75.96	75.92	**76.33**
SM-CRF	bc	72.33	75.54	75.04	**76.50**
	bn	72.18	74.86	75.10	**75.44**
	cts	85.68	85.96	86.15	**86.89**
	nw	72.70	76.19	75.92	**76.50**
	un	66.83	**72.94**	72.91	72.93
	wl	64.57	69.90	69.76	**70.30**
	avg	72.38	75.90	75.81	**76.43**

Table 1: F-score on the ACE NER task. The domains are broadcast conversations (bc), broadcast news (bn), conversational telephone speech (cts), newswire (nw), usenet (un) and weblog (wl). The macro-average (avg) over the 6 domains is also shown in the table.

Model	Dom.	TGT	ALL	AUG	RF
LR	book	75.83	79.33	79.00	**80.67**
	dvd	82.17	82.83	**83.83**	**83.83**
	elec.	84.67	84.67	**84.83**	**84.83**
	kit.	83.83	86.33	86.17	**87.33**
	avg	81.63	83.29	83.46	**84.17**
SVM	book	76.83	80.67	80.33	**81.00**
	dvd	83.17	83.17	82.50	**84.00**
	elec.	85.00	**86.50**	85.83	85.67
	kit.	86.33	85.83	**88.33**	87.83
	avg	82.83	84.04	84.25	**84.63**

Table 2: Accuracies on the sentiment classification task. The domains are books (book), dvds (dvd), electronics (elec.) and kitchen (kit.). The macro-average (avg) over the four domains are also shown in the table.

AUG The FEDA approach, and
RF Our proposed regularization framework.

We use a 40/30/30 train-development-test split and report the results on the test set. The regularization parameters were tuned on the development set over a logarithmic scale between 10^{-3} to 10^3. For our framework, we used random search to tune the parameters, since an exhaustive search is too expensive (21 parameters for 6 domains). We choose the within-domain $\eta_{0,i}$ to be close to those used for the *ALL* and *AUG* model, while choosing the other $\eta_{j,k}$ to be 1-2 orders of magnitude higher. A good model could quickly be found that generally beats the baselines on the development set and also generalizes well to the test set. We show the results for NER in Table 1 and the sentiment task in Table 2.

4 Discussion

Our proof did not require any assumption about $\mathcal{L}$, as long as L_2 regularization is used. This means our result is applicable to a variety of models such as SVM, LR, and CRF (where L_2 regularization is used for the latter two models). Theoretically, we have shown the equivalence of DA optimization problems. Empirically, for non-convex objectives, different approaches may arrive at different solutions. However, for convex loss functions, our objective (Equation 2) is also convex, and all approaches should share the same solution.

We have shown that we can map the FEDA optimization problem to our framework. The converse is false: for any problem in this family (with arbitrary choices of η), we can only solve it using FEDA if there are only 2 domains, or if all regularization hyper-parameters are equal. Some parameter configurations in this family are "unreachable" by the feature augmentation technique. This is because in Theorem 2.3, the values of η's are defined based on λ's and therefore possess certain properties. For example, they must at least satisfy such constraints as $\eta_{i,k}\eta_{k,j} = \eta_{i,l}\eta_{l,j}$ for any $i \leq k, l \leq j$. We have seen that some of those unreachable problems could give us better empirical results. Can we find an alternative simple adaptation method such that all problems in this family are "reachable"? This is a question that needs to be addressed in future research.

5 Conclusion

In this paper, we presented a framework for domain adaptation that generalizes several previous works (Daumé III, 2007; Finkel and Manning, 2009; Evgeniou and Pontil, 2004). Our approach allows practitioners to specify the amount of transfer between domains via regularization hyper-parameters. These parameters could be tuned based on intuition or using held-out data. In future work we could also seek to find methods that can automatically optimize these parameters. The supplementary material of this paper is available at `http://statnlp.org/research/ml/`.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, and Zhanming Jie for his help on this work. The experiments of this work were done when Jonathan Löfgren was a visiting student at Singapore University of Technology and Design (SUTD). This work is supported by MOE Tier 1 grant SUTDT12015008.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*.

Ciprian Chelba and Alex Acero. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of 2010 Workshop on Domain Adaptation for Natural Language Processing*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of KDD*.

J. R. Finkel and C.D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of ACL*.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proceedings of HLT-NAACL*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of EMNLP*.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL/IJCNLP*.

Aldrian Obaja Muis and Wei Lu. 2016. Weak semi-markov crfs for noun phrase chunking in informal text. In *Proceedings of NAACL*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October.

Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. 2005. To transfer or not to transfer. In *In NIPS'05 Workshop, Inductive Transfer: 10 Years Later*.

Coverage Embedding Models for Neural Machine Translation

Haitao Mi Baskaran Sankaran Zhiguo Wang Abe Ittycheriah*
T.J. Watson Research Center
IBM
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{hmi, bsankara, zhigwang, abei}@us.ibm.com

Abstract

In this paper, we enhance the attention-based neural machine translation (NMT) by adding explicit coverage embedding models to alleviate issues of repeating and dropping translations in NMT. For each source word, our model starts with a *full* coverage embedding vector to track the *coverage* status, and then keeps updating it with neural networks as the translation goes. Experiments on the large-scale Chinese-to-English task show that our enhanced model improves the translation quality significantly on various test sets over the strong large vocabulary NMT system.

1 Introduction

Neural machine translation (NMT) has gained popularity in recent years (e.g. (Bahdanau et al., 2014; Jean et al., 2015; Luong et al., 2015; Mi et al., 2016b; Li et al., 2016)), especially for the attention-based models of Bahdanau et al. (2014). The attention at each time step shows which source word the model should focus on to predict the next target word. However, the attention in each step only looks at the previous hidden state and the previous target word, there is no history or coverage information typically for each source word. As a result, this kind of model suffers from issues of repeating or dropping translations.

The traditional statistical machine translation (SMT) systems (e.g. (Koehn, 2004)) address the above issues by employing a source side "coverage vector" for each sentence to indicate explicitly which words have been translated, which parts have not yet. A coverage vector starts with all zeros, meaning no word has been translated. If a source word at position j got translated, the coverage vector sets position j as 1, and they won't use this source word in future translation. This mechanism avoids the repeating or dropping translation problems.

However, it is not easy to adapt the "coverage vector" to NMT directly, as attentions are soft probabilities, not 0 or 1. And SMT approaches handle one-to-many fertilities by using phrases or hiero rules (predict several words in one step), while NMT systems only predict one word at each step.

In order to alleviate all those issues, we borrow the basic idea of "coverage vector", and introduce a coverage embedding vector for each source word. We keep updating those embedding vectors at each translation step, and use those vectors to track the *coverage* information.

Here is a brief description of our approach. At the beginning of translation, we start from a *full* coverage embedding vector for each source word. This is different from the "coverage vector" in SMT in following two aspects:

- each source word has its own coverage embedding vector, instead of 0 or 1, a scalar, in SMT,
- we start with a *full* embedding vector for each word, instead of 0 in SMT.

After we predict a translation word y_t at time step t, we need to update each coverage embedding vector accordingly based on the attentions in the current step. Our motivation is that if we observe a very high attention over x_i in this step, there is a high chance that x_i and y_t are translation equivalent. So the embedding vector of x_i should come to *empty* (a zero vector) in a one-to-one translation case, or subtract the embedding of y_t for the one-to-many translation case. An *empty* coverage embedding of a word x_i indicates this word is translated, and we can not translate x_i again in future. Empirically, we model this procedure by using neural networks (gated recurrent unit (GRU) (Cho et al., 2014) or direct subtraction).

Large-scale experiments over Chinese-to-English on various test sets show that our method improves the translation quality significantly over the large vocabulary NMT system (Section 5).

*Work done while at IBM. To contact Abe, aittycheriah@google.com.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 955–960,
Austin, Texas, November 1-5, 2016.

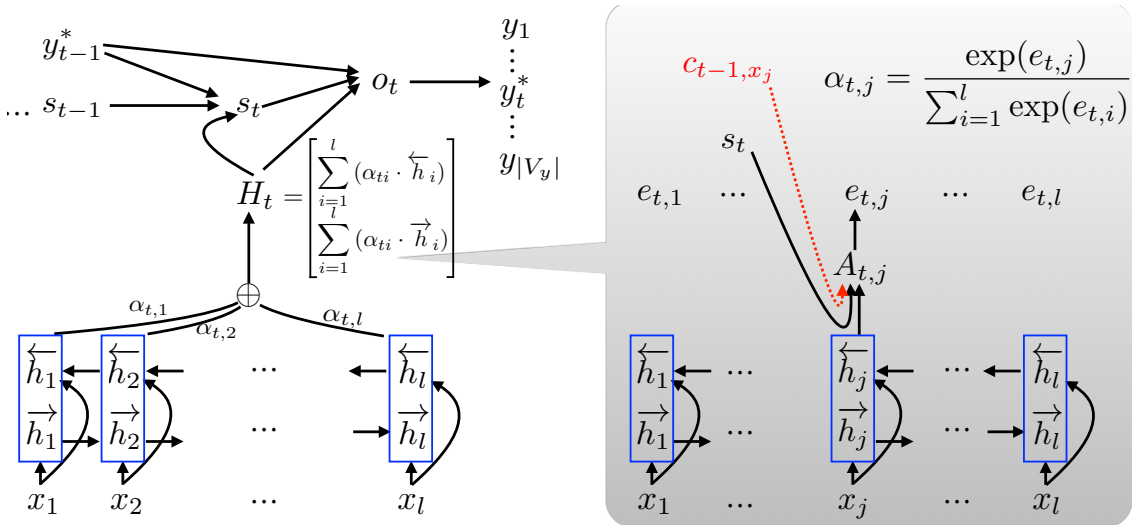


Figure 1: The architecture of attention-based NMT.The source sentence is $\mathbf{x} = (x_1, ..., x_l)$ with length l, the translation is $\mathbf{y}^* = (y_1^*, ..., y_m^*)$ with length m. $\overleftarrow{h_i}$ and $\overrightarrow{h_i}$ are bi-directional encoder states. $\alpha_{t,j}$ is the attention probability at time t, position j. H_t is the weighted sum of encoding states. s_t is a hidden state. o_t is an output state. Another one layer neural network projects o_t to the target output vocabulary, and conducts softmax to predict the probability distribution over the output vocabulary. The attention model (in right gray box) is a two layer feedforward neural network, $A_{t,j}$ is an intermediate state, then another layer converts it into a real number $e_{t,j}$, the final attention probability at position j is $\alpha_{t,j}$. We plug coverage embedding models into NMT model by adding an input c_{t-1,x_j} to $A_{t,j}$ (the red dotted line).

2 Neural Machine Translation

As shown in Figure 1, attention-based neural machine translation (Bahdanau et al., 2014) is an encoder-decoder network. the encoder employs a bi-directional recurrent neural network to encode the source sentence $\mathbf{x} = (x_1, ..., x_l)$, where l is the sentence length, into a sequence of hidden states $\mathbf{h} = (h_1, ..., h_l)$, each h_i is a concatenation of a left-to-right $\overrightarrow{h_i}$ and a right-to-left $\overleftarrow{h_i}$,

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix},$$

where $\overleftarrow{f}$ and $\overrightarrow{f}$ are two GRUs.

Given the encoded $\mathbf{h}$, the decoder predicts the target translation by maximizing the conditional log-probability of the correct translation $\mathbf{y}^* = (y_1^*, ...y_m^*)$, where m is the sentence length. At each time t, the probability of each word y_t from a target vocabulary V_y is:

$$p(y_t|\mathbf{h}, y_{t-1}^*..y_1^*) = g(s_t, y_{t-1}^*), \quad (1)$$

where g is a two layer feed-forward neural network (o_t is a intermediate state) over the embedding of the previous word y_{t-1}^*, and the hidden state s_t. The s_t is computed as:

$$s_t = q(s_{t-1}, y_{t-1}^*, H_t) \quad (2)$$

$$H_t = \begin{bmatrix} \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \quad (3)$$

where q is a GRU, H_t is a weighted sum of $\mathbf{h}$, the weights, α, are computed with a two layer feed-forward neural network r:

$$\alpha_{t,i} = \frac{\exp\{r(s_{t-1}, h_i, y_{t-1}^*)\}}{\sum_{k=1}^{l} \exp\{r(s_{t-1}, h_k, y_{t-1}^*)\}} \quad (4)$$

3 Coverage Embedding Models

Our basic idea is to introduce a coverage embedding for each source word, and keep updating this embedding at each time step. Thus, the coverage embedding for a sentence is a matrix, instead of a vector in SMT. As different words have different fertilities (one-to-one, one-to-many, or one-to-zero), similar to word embeddings, each source word has its own coverage embedding vector. For simplicity, the number of coverage embedding vectors is the same as the source word vocabulary size.

At the beginning of our translation, our coverage embedding matrix $(c_{0,x_1}, c_{0,x_2}, ...c_{0,x_l})$ is initialized with the coverage embedding vectors of all the source words.

Then we update them with neural networks (a GRU (Section 3.1.1) or a subtraction (Section 3.1.2))

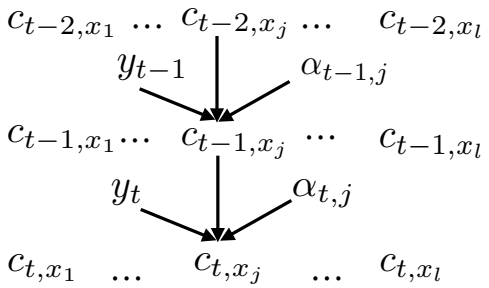


Figure 2: The coverage embedding model with a GRU at time step $t-1$ and t. $c_{0,1}$ to $c_{0,l}$ are initialized with the word coverage embedding matrix

until we translation all the source words.

In the middle of translation, some coverage embeddings should be close to *zero*, which indicate those words are covered or translated, and can not be translated in future steps. Thus, in the end of translation, the embedding matrix should be close to zero, which means all the words are covered.

In the following part, we first show two updating methods, then we list the NMT objective that takes into account the embedding models.

3.1 Updating Methods

3.1.1 Updating with a GRU

Figure 2 shows the updating method with a GRU. Then, at time step t, we feed y_t and $\alpha_{t,j}$ to the coverage model (shown in Figure 2),

$$z_{t,j} = \sigma(W^{zy} y_t + W^{z\alpha}\alpha_{t,j} + U^z c_{t-1,x_j})$$
$$r_{t,j} = \sigma(W^{ry} y_t + W^{r\alpha}\alpha_{t,j} + U^r c_{t-1,x_j})$$
$$\tilde{c}_{t,x_j} = \tanh(W y_t + W^{\alpha}\alpha_{t,j} + r_{t,j} \circ U c_{t-1,x_j})$$
$$c_{t,x_j} = z_{t,j} \circ c_{t-1,x_j} + (1 - z_{t,j}) \circ \tilde{c}_{t,x_j},$$

where, z_t is the update gate, r_t is the reset gate, $\tilde{c}_t$ is the new memory content, and c_t is the final memory. The matrix W^{zy}, $W^{z\alpha}$, U^z, W^{ry}, $W^{r\alpha}$, U^r, W^y, W^{α} and U are shared across different position j. $\circ$ is a pointwise operation.

3.1.2 Updating as Subtraction

Another updating method is to subtract the embedding of y_t directly from the coverage embedding c_{t,x_j} with a weight $\alpha_{t,j}$ as

$$c_{t,x_j} = c_{t-1,x_j} - \alpha_{t,j} \circ (W^{y\to c} y_t), \qquad (5)$$

where $W^{y\to c}$ is a matrix that coverts word embedding of y_t to the same size of our coverage embedding vector c.

3.2 Objectives

We integrate our coverage embedding models into the attention NMT (Bahdanau et al., 2014) by adding c_{t-1,x_j} to the first layer of the attention model (shown in the red dotted line in Figure 1).

Hopefully, if y_t is partial translation of x_j with a probability $\alpha_{t,j}$, we only remove partial information of c_{t-1,x_j}. In this way, we enable coverage embedding c_{0,x_j} to encode fertility information of x_j.

As we have mentioned, in the end of translation, we want all the coverage embedding vectors to be close to *zero*. So we also minimize the absolute values of embedding matrixes as

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \Bigg\{ \sum_{t=1}^{m} \log p(y_t^{*n} | \mathbf{x}^n, y_{t-1}^{*n} .. y_1^{*n}) - \lambda \sum_{i=1}^{l} ||c_{m,x_i}|| \Bigg\}, \qquad (6)$$

where λ is the coefficient of our coverage model.

As suggested by Mi et al. (2016a), we can also use some supervised alignments in our training. Then, we know exactly when each c_{t,x_j} should become close to *zero* after step t. Thus, we redefine Equation 6 as:

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \Bigg\{ \sum_{t=1}^{m} \log p(y_t^{*n} | \mathbf{x}^n, y_{t-1}^{*n} .. y_1^{*n}) - \lambda \sum_{i=1}^{l} (\sum_{j=a_{x_i}}^{m} ||c_{j,x_i}||) \Bigg\}, \qquad (7)$$

where a_{x_i} is the maximum index on the target sentence x_i can be aligned to.

4 Related Work

There are several parallel and independent related work (Tu et al., 2016; Feng et al., 2016; Cohn et al., 2016). Tu et al. (2016) is the most relevant one. In their paper, they also employ a GRU to model the coverage vector. One main difference is that our model introduces a specific coverage embedding vector for each source word, in contrast, their work initializes the word coverage vector with a scalar with a uniform distribution. Another difference lays in the fertility part, Tu et al. (2016) add an accumulate operation and a fertility function to simulate

the process of one-to-many alignments. In our approach, we add fertility information directly to coverage embeddings, as each source word has its own embedding. The last difference is that our baseline system (Mi et al., 2016b) is an extension of the large vocabulary NMT of Jean et al. (2015) with candidate list decoding and UNK replacement, a much stronger baseline system.

Cohn et al. (2016) augment the attention model with well-known features in traditional SMT, including positional bias, Markov conditioning, fertility and agreement over translation directions. This work is orthogonal to our work.

5 Experiments

5.1 Data Preparation

We run our experiments on Chinese to English task. We train our machine translation systems on two training sets. The first training corpus consists of approximately 5 million sentences available within the DARPA BOLT Chinese-English task. The second training corpus adds HK Law, HK Hansard and UN data, the total number of training sentence pairs is 11 million. The Chinese text is segmented with a segmenter trained on CTB data using conditional random fields (CRF).

Our development set is the concatenation of several tuning sets (GALE Dev, P1R6 Dev, and Dev 12) released under the DARPA GALE program. The development set is 4491 sentences in total. Our test sets are NIST MT06, MT08 news, and MT08 web.

For all NMT systems, the full vocabulary sizes for thr two training sets are $300k$ and $500k$ respectively. The coverage embedding vector size is 100. In the training procedure, we use AdaDelta (Zeiler, 2012) to update model parameters with a mini-batch size 80. Following Mi et al. (2016b), the output vocabulary for each mini-batch or sentence is a sub-set of the full vocabulary. For each source sentence, the sentence-level target vocabularies are union of top $2k$ most frequent target words and the top 10 candidates of the word-to-word/phrase translation tables learned from 'fast_align' (Dyer et al., 2013). The maximum length of a source phrase is 4. In the training time, we add the reference in order to make the translation reachable.

Following Jean et al. (2015), We dump the alignments, attentions, for each sentence, and replace UNKs with the word-to-word translation model or the aligned source word.

Our traditional SMT system is a hybrid syntax-based tree-to-string model (Zhao and Al-onaizan, 2008), a simplified version of Liu et al. (2009) and Cmejrek et al. (2013). We parse the Chinese side with Berkeley parser, and align the bilingual sentences with GIZA++. Then we extract Hiero and tree-to-string rules on the training set. Our two 5-gram language models are trained on the English side of the parallel corpus, and on monolingual corpora (around 10 billion words from Gigaword (LDC2011T07)), respectively. As suggestion by Zhang (2016), NMT systems can achieve better results with the help of those monolingual corpora. We tune our system with PRO (Hopkins and May, 2011) to minimize (TER- BLEU)/2 on the development set.

5.2 Translation Results

Table 1 shows the results of all systems on 5 million training set. The traditional syntax-based system achieves 9.45, 12.90, and 17.72 on MT06, MT08 News, and MT08 Web sets respectively, and 13.36 on average in terms of (TER- BLEU)/2. The large-vocabulary NMT (LVNMT), our baseline, achieves an average (TER- BLEU)/2 score of 15.74, which is about 2 points worse than the hybrid system.

We test four different settings for our coverage embedding models:

- $\mathbf{U}_{GRU}$: updating with a GRU;
- $\mathbf{U}_{Sub}$: updating as a subtraction;
- $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$: combination of two methods (do not share coverage embedding vectors);
- **+Obj.**: $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ plus an additional objective in Equation 6[1].

$\mathbf{U}_{GRU}$ improves the translation quality by 1.3 points on average over LVNMT. And $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ achieves the best average score of 13.14, which is about 2.6 points better than LVNMT. All the improvements of our coverage embedding models over LVNMT are statistically significant with the sign-test of Collins et al. (2005). We believe that we need to explore more hyper-parameters of **+Obj.** in order to get even better results over $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$.

[1]We use two λs for $\mathbf{U}_{GRU}$ and $\mathbf{U}_{Sub}$ separately, and we test $\lambda_{GRU} = 1 \times 10^{-4}$ and $\lambda_{Sub} = 1 \times 10^{-2}$ in our experiments.

single system		MT06			MT08						avg.
					News			Web			
		BP	BLEU	T-B	BP	BLEU	T-B	BP	BLEU	T-B	T-B
Tree-to-string		0.95	34.93	9.45	0.94	31.12	12.90	0.90	23.45	17.72	13.36
LVNMT		0.96	34.53	12.25	0.93	28.86	17.40	0.97	26.78	17.57	15.74
Ours	$\mathbf{U}_{GRU}$	0.92	35.59	10.71	0.89	30.18	15.33	0.97	27.48	16.67	14.24
	$\mathbf{U}_{Sub}$	0.91	35.90	10.29	0.88	30.49	15.23	0.96	27.63	16.12	13.88
	$\mathbf{U}_{GRU}$+$\mathbf{U}_{Sub}$	0.92	36.60	9.36	0.89	31.86	13.69	0.95	27.12	16.37	13.14
	+Obj.	0.93	36.80	9.78	0.90	31.83	14.20	0.95	28.28	15.73	13.24

Table 1: Single system results in terms of (TER-BLEU)/2 (the lower the better) on 5 million Chinese to English training set. NMT results are on a large vocabulary ($300k$) and with UNK replaced. $\mathbf{U}_{GRU}$: updating with a GRU; $\mathbf{U}_{Sub}$: updating as a subtraction; $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$: combination of two methods (do not share coverage embedding vectors); **+Obj.**: $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ with an additional objective in Equation 6, we have two λs for $\mathbf{U}_{GRU}$ and $\mathbf{U}_{Sub}$ separately, and we test $\lambda_{GRU} = 1 \times 10^{-4}$ and $\lambda_{Sub} = 1 \times 10^{-2}$.

single system	MT06		MT08				avg.
			News		Web		
	BP	T-B	BP	T-B	BP	T-B	T-B
Tree-to-string	0.90	8.70	0.84	12.65	0.84	17.00	12.78
LVNMT	0.96	9.78	0.94	14.15	0.97	15.89	13.27
$\mathbf{U}_{GRU}$	0.97	8.62	0.95	12.79	0.97	15.34	12.31

Table 2: Single system results in terms of (TER-BLEU)/2 on 11 million set. NMT results are on a large vocabulary ($500k$) and with UNK replaced. Due to the time limitation, we only have the results of $\mathbf{U}_{GRU}$ system.

Table 2 shows the results of 11 million systems, LVNMT achieves an average (TER-BLEU)/2 of 13.27, which is about 2.5 points better than 5 million LVNMT. The result of our $\mathbf{U}_{GRU}$ coverage model gives almost 1 point gain over LVNMT. Those results suggest that the more training data we use, the stronger the baseline system becomes, and the harder to get improvements. In order to get a reasonable or strong NMT system, we have to conduct experiments over a large-scale training set.

5.3 Alignment Results

Table 3 shows the F1 scores on the alignment test set (447 hand aligned sentences). The MaxEnt model is trained on $67k$ hand-aligned data, and achieves an F1 score of 75.96. For NMT systems, we dump alignment matrixes, then, for each target word we only add the highest probability link if it is higher than 0.2. Results show that our best coverage model, $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$, improves the F1 score by 2.2 points over the sorce of LVNMT.

We also check the repetition statistics of NMT outputs. We simply compute the number of repeated phrases (length longer or equal than 4 words) for each sentence. On MT06 test set, the 5 million LVNMT has 209 repeated phrases, our $\mathbf{U}_{GRU}$ system reduces it significantly to 79, $\mathbf{U}_{GRU}$+$\mathbf{U}_{Sub}$ and **+Obj.** only have 50 and 47 repeated phrases, respectively. The 11 million LVNMT gets 115 repeated phrases, and $\mathbf{U}_{GRU}$ reduces it further down to 16. Those trends hold across other test sets. Those statistics show that a larger training set or coverage embedding models alleviate the repeating problem in NMT.

system		pre.	rec.	F1
MaxEnt		74.86	77.10	75.96
LVNMT		47.88	41.06	44.21
Ours	$\mathbf{U}_{GRU}$	51.11	41.42	45.76
	$\mathbf{U}_{Sub}$	49.07	42.49	45.55
	$\mathbf{U}_{GRU}$+$\mathbf{U}_{Sub}$	49.46	43.83	46.47
	+Obj.	49.78	41.73	45.40

Table 3: Alignment F1 scores of different models.

6 Conclusion

In this paper, we propose simple, yet effective, coverage embedding models for attention-based NMT. Our model learns a special coverage embedding vector for each source word to start with, and keeps updating those coverage embeddings with neural networks as the translation goes. Experiments on the large-scale Chinese-to-English task show significant improvements over the strong LVNMT system.

Acknowledgment

We thank reviewers for their useful comments.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv e-prints*, September.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Martin Cmejrek, Haitao Mi, and Bowen Zhou. 2013. Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 545–555, Seattle, Washington, USA, October. Association for Computational Linguistics.

T. Cohn, C. D. V. Hoang, E. Vymolova, K. Yao, C. Dyer, and G. Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. *ArXiv e-prints*, January.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

S. Feng, S. Liu, M. Li, and M. Zhou. 2016. Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model. *ArXiv e-prints*, January.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*, pages 1–10, Beijing, China, July.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proceedings of IJCAI 2016*, pages 2852–2858, New York, NY, USA, July.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint decoding with multiple translation models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 576–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016a. Supervised attentions for neural machine translation. In *Proceedings of EMNLP*, Austin, USA, November.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016b. Vocabulary manipulation for neural machine translation. In *Proceedings of ACL*, Berlin, Germany, August.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. 2016. Coverage-based Neural Machine Translation. *ArXiv e-prints*, January.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*.

Jiajun Zhang. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP 2016*, Austin, Texas, USA, November.

Bing Zhao and Yaser Al-onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 572–581, Stroudsburg, PA, USA. Association for Computational Linguistics.

Neural Morphological Analysis: Encoding-Decoding Canonical Segments

Katharina Kann
Center for Information and Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Canonical morphological segmentation aims to divide words into a sequence of standardized segments. In this work, we propose a character-based neural encoder-decoder model for this task. Additionally, we extend our model to include morpheme-level and lexical information through a neural reranker. We set the new state of the art for the task improving previous results by up to 21% accuracy. Our experiments cover three languages: English, German and Indonesian.

1 Introduction

Morphological segmentation aims to divide words into morphemes, meaning-bearing sub-word units. Indeed, segmentations have found use in a diverse set of NLP applications, e.g., automatic speech recognition (Afify et al., 2006), keyword spotting (Narasimhan et al., 2014), machine translation (Clifton and Sarkar, 2011) and parsing (Seeker and Çetinoğlu, 2015). In the literature, most research has traditionally focused on *surface segmentation*, whereby a word w is segmented into a sequence of substrings whose concatenation is the entire word; see Ruokolainen et al. (2016) for a survey. In contrast, we consider *canonical segmentation*: w is divided into a sequence of standardized segments. To make the difference concrete, consider the following example: the surface segmentation of the complex English word *achievability* is *achiev+abil+ity*, whereas its canonical segmentation is *achieve+able+ity*, i.e., we restore the alterations made during word formation.

Canonical versions of morphological segmentation have been introduced multiple times in the literature (Kay, 1977; Naradowsky and Goldwater, 2009; Cotterell et al., 2016). Canonical segmentation has several representational advantages over surface segmentation, e.g., whether two words share a morpheme is no longer obfuscated by orthography. However, it also introduces a hard algorithmic challenge: in addition to segmenting a word, we must reverse orthographic changes, e.g., mapping *achievability*↦*achieveableity*.

Computationally, canonical segmentation can be seen as a sequence-to-sequence problem: we must map a word form to a canonicalized version with segmentation boundaries. Inspired by the recent success of neural encoder-decoder models (Sutskever et al., 2014) for sequence-to-sequence problems in NLP, we design a neural architecture for the task. However, a naïve application of the encoder-decoder model ignores much of the linguistic structure of canonical segmentation—it cannot directly model the individual canonical segments, e.g., it cannot easily produce segment-level embeddings. To solve this, we use a neural reranker on top of the encoder-decoder, allowing us to embed both characters and entire segments. The combined approach outperforms the state of the art by a wide margin (up to 21% accuracy) in three languages: English, German and Indonesian.

2 Neural Canonical Segmentation

We begin by formally describing the canonical segmentation task. Given a discrete alphabet Σ (e.g., the 26 letters of the English alphabet),

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 961–967,
Austin, Texas, November 1-5, 2016.

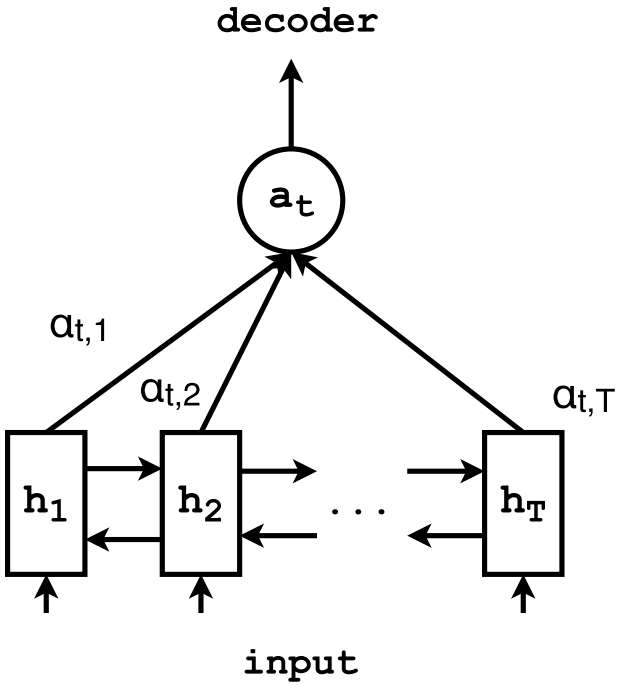


Figure 1: Detailed view of the attention mechanism of the neural encoder-decoder.

our goal is to map a word $w \in \Sigma^*$ (e.g., w=*achievability*), to a canonical segmentation $c \in \Omega^*$ (e.g., c=*achieve+able+ity*). We define $\Omega = \Sigma \cup \{+\}$, where $+$ is a distinguished separation symbol. Additionally, we will write the segmented form as $c=\sigma_1+\sigma_2+\ldots+\sigma_n$, where each segment $\sigma_i \in \Sigma^*$ and n is the number of canonical segments.

We take a probabilistic approach and, thus, attempt to learn a distribution $p(c \mid w)$. Our model consists of two parts. First, we apply an encoder-decoder recurrent neural network (RNN) (Bahdanau et al., 2014) to the sequence of characters of the input word to obtain candidate canonical segmentations. Second, we define a neural reranker that allows us to embed individual morphemes and chooses the final answer from within a set of candidates generated by the encoder-decoder.

2.1 Neural Encoder-Decoder

Our encoder-decoder is based on Bahdanau et al. (2014)'s neural machine translation model.[1] The **encoder** is a bidirectional gated RNN (GRU) (Cho et al., 2014b). Given a word $w \in \Sigma^*$, the input to the encoder is the sequence of characters of w, represented as one-hot vectors. The **decoder** defines a conditional probability distribution over $c \in \Omega^*$ given w:

$$\begin{aligned} p_{\text{ED}}(c \,|\, w) &= \prod_{t=1}^{|c|} p(c_t | c_1, \ldots, c_{t-1}, w) \\ &= \prod_{t=1}^{|c|} g(c_{t-1}, s_t, a_t) \end{aligned}$$

where g is a nonlinear activation function, s_t is the state of the decoder at t and a_t is a weighted sum of the $|w|$ states of the encoder. The state of the encoder for w_i is the concatenation of forward and backward hidden states $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ for w_i. An overview of how the attention weight and the weighted sum a_t are included in the architecture can be seen in Figure 1. The attention weights $\alpha_{t,i}$ at each timestep t are computed based on the respective encoder state and the decoder state s_t. See Bahdanau et al. (2014) for further details.

2.2 Neural Reranker

The encoder-decoder, while effective, predicts each output character in Ω sequentially. It does not use explicit representations for entire segments and is incapable of incorporating simple lexical information, e.g., does this canonical segment occur as an independent word in the lexicon? Therefore, we extend our model with a reranker.

The reranker rescores canonical segmentations from a candidate set, which in our setting is sampled from p_{ED}. Let the sample set be $\mathcal{S}_w = \{k^{(i)}\}_{i=1}^{N}$ where $k^{(i)} \sim p_{\text{ED}}(c \mid w)$. We define the neural reranker as

$$p_\theta(c \,|\, w) = \frac{\exp\Big(u^\top \tanh(W v_c) + \tau \log p_{\text{ED}}(c \,|\, w)\Big)}{Z_\theta}$$

where $v_c = \sum_{i=1}^{n} v_{\sigma_i}$ (recall $c = \sigma_1+\sigma_2+\ldots+\sigma_n$) and v_{σ_i} is a one-hot morpheme embedding of σ_i with an additional binary dimension marking if σ_i occurs independently as a word in the language.[2] The partition function is $Z_\theta(w)$ and the parameters are $\theta = \{u, W, \tau\}$. The parameters W and u

[1] github.com/mila-udem/blocks-examples/tree/master/machine_translation

[2] To determine if a canonical segment is in the lexicon, we check its occurrence in ASPELL. Alternatively, one could ask whether it occurs in a large corpus, e.g., Wikipedia.

are projection and hidden layers, respectively, of a multi-layered perceptron and τ can be seen as a temperature parameter that anneals the encoder-decoder model p_{ED} (Kirkpatrick, 1984). We define the partition function over the sample set $\mathcal{S}_w$:

$$Z_\theta = \sum_{k \in \mathcal{S}_w} \exp\left(u^\top \tanh(W v_k) + \tau \log p_{\text{ED}}(k \mid w)\right).$$

The reranking model's ability to embed morphemes is important for morphological segmentation since we often have strong corpus-level signals. The reranker also takes into account the character-level information through the score of the encoder-decoder model. Due to this combination we expect stronger performance.

3 Related Work

Various approaches to morphological segmentation have been proposed in the literature. In the unsupervised realm, most work has been based on the principle of minimum description length (Cover and Thomas, 2012), e.g., LINGUISTICA (Goldsmith, 2001; Lee and Goldsmith, 2016) or MORFESSOR (Creutz and Lagus, 2002; Creutz et al., 2007; Poon et al., 2009). MORFESSOR was later extended to a semi-supervised version by Kohonen et al. (2010). Supervised approaches have also been considered. Most notably, Ruokolainen et al. (2013) developed a supervised approach for morphological segmentation based on conditional random fields (CRFs) which they later extended to work also in a semi-supervised way (Ruokolainen et al., 2014) using letter successor variety features (Hafer and Weiss, 1974). Similarly, Cotterell et al. (2015) improved performance with a semi-Markov CRF.

More recently, Wang et al. (2016) achieved state-of-the-art results on surface morphological segmentation using a window LSTM. Even though Wang et al. (2016) also employ a recurrent neural network, we distinguish our approach, in that we focus on *canonical* morphological segmentation, rather than *surface* morphological segmentation.

Naturally, our approach is also relevant to other applications of recurrent neural network transduction models (Sutskever et al., 2014; Cho et al., 2014a). In addition to machine translation (Bahdanau et al., 2014), these models have been successfully applied to many areas of NLP, including parsing (Vinyals et al., 2015), morphological reinflection (Kann and Schütze, 2016) and automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

4 Experiments

To enable comparison to earlier work, we use a dataset that was prepared by Cotterell et al. (2016) for canonical segmentation.[3]

4.1 Languages

The dataset we work on covers 3 languages: English, German and Indonesian. English and German are West Germanic Languages, with the former being an official languages in nearly 60 different states and the latter being mainly spoken in Western Europe. Indonesian — or *Bahasa Indonesia*— is the official language of Indonesia.

Cotterell et al. (2016) report the best experimental results for Indonesian, followed by English and finally German. The high error rate for German might be caused by it being rich in orthografic changes. In contrast, Indonesian morphology is comparatively simple.

4.2 Corpora

The data for the English language was extracted from segmentations derived from the CELEX database (Baayen et al., 1993). The German data was extracted from DerivBase (Zeller et al., 2013), which provides a collection of derived forms together with the transformation rules, which were used to create the canonical segmentations. Finally, the data for Bahasa Indonesia was collected by using the output of the MORPHIND analyzer (Larasati et al., 2011), together with an open-source corpus of Indonesian. For each language we used the 10,000 forms that were selected at random by Cotterell et al. (2016) from a uniform distribution over types to form the corpus. Following them, we perform our experiments on 5 splits of the data into 8000 training forms, 1000 development forms and 1000 test forms and report averages.

[3] ryancotterell.github.io/canonical-segmentation

4.3 Training

We train an ensemble of five encoder-decoder models. The encoder and decoder RNNs each have 100 hidden units. Embedding size is 300. We use ADADELTA (Zeiler, 2012) with a minibatch size of 20. We initialize all weights (encoder, decoder, embeddings) to the identity matrix and the biases to zero (Le et al., 2015). All models are trained for 20 epochs. The hyperparameter values are taken from Kann and Schütze (2016) and kept unchanged for the application to canonical segmentation described here.

To train the reranking model, we first gather the sample set $\mathcal{S}_w$ on the training data. We take 500 individual samples, but (as we often sample the same form multiple times) $|\mathcal{S}_w| \approx 5$. We optimize the log-likelihood of the training data using ADADELTA. For generalization, we employ L_2 regularization and we perform grid search to determine the coefficient $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. To decode the model, we again take 500 samples to populate $\mathcal{S}_w$ and select the best segmentation.

Baselines. Our first baseline is the joint transduction and segmentation model (**JOINT**) of Cotterell et al. (2016). It is the current state of the art on the datasets we use and the task of canonical segmentation in general. This model uses a jointly trained, separate transduction and segmentation component. Importantly, the joint model of Cotterell et al. (2016) *already contains segment-level features*. Thus, reranking this baseline would not provide a similar boost.

Our second baseline is a weighted finite-state transducer (**WFST**) (Mohri et al., 2002) with a log-linear parameterization (Dreyer et al., 2008), again, taken from Cotterell et al. (2016). The WFST baseline is particularly relevant because, like our encoder-decoder, it formulates the problem directly as a string-to-string transduction.

Evaluation Metrics. We follow Cotterell et al. (2016) and use the following evaluation measures: error rate, edit distance and morpheme F_1. Error rate is defined as 1 minus the proportion of guesses that are completely correct. Edit distance is the Levenshtein distance between guess and gold standard. For this, guess and gold are each represented as one string with a distinguished character denoting the segment boundaries. Morpheme F_1 compares the morphemes in guess and gold. Precision (resp. recall) is the proportion of morphemes in guess (resp. gold) that occur in gold (resp. guess).

		RR	ED	Joint	WFST	UB
error	en	**.19** (.01)	.25 (.01)	0.27 (.02)	0.63 (.01)	.06 (.01)
	de	**.20** (.01)	.26 (.02)	0.41 (.03)	0.74 (.01)	.04 (.01)
	id	**.05** (.01)	.09 (.01)	0.10 (.01)	0.71 (.01)	.02 (.01)
edit	en	**.21** (.02)	.47 (.02)	0.98 (.34)	1.35 (.01)	.10 (.02)
	de	**.29** (.02)	.51 (.03)	1.01 (.07)	4.24 (.20)	.06 (.01)
	id	**.05** (.00)	.12 (.01)	0.15 (.02)	2.13 (.01)	.02 (.01)
F_1	en	**.82** (.01)	.78 (.01)	0.76 (.02)	0.53 (.02)	.96 (.01)
	de	**.87** (.01)	.86 (.01)	0.76 (.02)	0.59 (.02)	.98 (.00)
	id	**.96** (.01)	.93 (.01)	0.80 (.01)	0.62 (.02)	.99 (.00)

Table 1: Error rate (top), edit distance (middle), F_1 (bottom) for canonical segmentation. Each double column gives the measure and its standard deviation. Best result on each line (excluding UB) in bold. RR: encoder-decoder+reranker. ED: encoder-decoder. JOINT, WFST: baselines (see text). UB: upper bound, the maximum score our reranker could obtain, i.e., considering the best sample in the predictions of ED.

5 Results

The results of the canonical segmentation experiment in Table 1 show that both of our models improve over all baselines. The encoder-decoder alone has a .02 (English), .15 (German) and .01 (Indonesion) lower error rate than the best baseline. The encoder-decoder improves most for the language for which the baselines did worst. This suggests that, for more complex languages, a neural network model might be a good choice.

The reranker achieves an additional improvement of .04 to .06. for the error rate. This is likely due to the additional information the reranker has access to: morpheme embeddings and existing words.

Important is also the upper bound we report. It shows the maximum performance the reranker could achieve, i.e., evaluates the best solution that appears in the set of candidate answers for the reranker. The right answer is contained in $\geq 94\%$ of samples. Note that, even though the upper bound goes up with the number of samples we take, there is no guarantee for any finite number of samples that they will contain the true answer. Thus, we would need to take an infinite number of samples to get a perfect upper bound. However, as the current upper bound is quite high, the encoder-decoder proves to be an appropri-

ate model for the task. Due to the large gap between the performance of the encoder-decoder and the upper bound, a better reranker could further increase performance. We will investigate ways to improve the reranker in future work.

Error analysis. We give for representative samples the error (E for the segmentation produced by our method) and the correct analysis (G for gold).

We first analyze cases in which the right answer does not appear at all in the samples drawn from the encoder-decoder. Those include problems with umlauts in German (G: *verflüchtigen*↦ *ver+flüchten+ig*, E: *verflucht+ig*) and orthographic changes at morpheme boundaries (G:*cutter*↦*cut+er*, E: *cutter* or *cutt+er*, sampled with similar frequency). There are also errors that are due to problems with the annotation, e.g., the following two gold segmentations are arguably incorrect: *tec*↦*detective* and *syrerin*↦*syr+er+in* (*syr* is neither a word nor an affix in German).

In other cases, the encoder-decoder does find the right solution (G), but gives a higher probability to an incorrect analysis (E). Examples are a wrong split into adjectives or nouns instead of verbs (G: *fügsamkeit*↦*fügen+sam+keit*, E: *fügsam+keit*), the other way around (G: *zähler*↦*zahl+er*, E: *zählen+er*), cases where the wrong morphemes are chosen (G: *precognition*↦*pre+cognition*, E: *precognit+ion*), difficult cases where letters have to be inserted (G: *redolence*↦*redolent+ence*, E: *re+dolence*) or words the model does not split up, even though they should be (G: *additive*↦ *addition+ive*, E: *additive*).

Based on its access to lexical information and morpheme embeddings, the reranker is able to correct some of the errors made by the encoder-decoder. Samples are G: *geschwisterpärchen*↦ *geschwisterpaar+chen*, E: *geschwisterpar+chen* (*geschwisterpaar* is a word in German but *geschwisterpar* is not) or G: *zickig*↦ *zicken+ig*, E: *zick+ig* (with *zicken*, but not *zick*, being a German word).

Finally, we want to know if segments that appear in the test set without being present in the training set are a source of errors. In order to investigate that, we split the test samples into two groups: The first group contains the samples for which our system finds the right answer. The second one contains all other samples. We compare the percentage of samples that do not appear in the training data for both groups. We exemplarily use the German data and the results results are shown in Table 2. First, it can be seen that very roughly about a third of all segments does not appear in the training data. This is mainly due to unseen lemmas as their stems are naturally unknown to the system. However, the correctly solved samples contain nearly 10% more unseen segments. As the average number of segments per word for wrong and right solutions — 2.44 and 2.11, respectively — does not differ by much, it seems unlikely that many errors are caused by unknown segments.

wrong samples	right samples
27.33 (.02)	36.60 (.01)

Table 2: Percentage of segments in the solutions for the test data that do not appear in the training set - split by samples that our system does or does not get right. We use the German data and average over the 5 splits. Standard deviation in parenthesis.

6 Conclusion and Future Work

We developed a model consisting of an encoder-decoder and a neural reranker for the task of canonical morphological segmentation. Our model combines character-level information with features on the morpheme level and external information about words. It defines a new state of the art, improving over baseline models by up to .21 accuracy, 16 points F_1 and .77 Levenshtein distance.

We found that $\geq 94\%$ of correct segmentations are in the sample set drawn from the encoder-decoder model, demonstrating the upper bound on the performance of our reranker is quite high; in future work, we hope to develop models to exploit this.

Acknowledgments

We gratefully acknowledge the financial support of Siemens for this research.

References

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal Arabic speech recognition. In *Proc. of INTERSPEECH*.

R. H. Baayen, R. Piepenbrock, and H. Van Rijn. 1993. The CELEX lexical data base on CD-ROM.

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

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proc. of ACL*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *Proc. of CoNLL*.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *Proc. of NAACL*.

Thomas M Cover and Joy A Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. of the ACL-02 Workshop on Morphological and Phonological Learning*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

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

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc of. ICASSP*.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proc. of ACL*.

Martin Kay. 1977. Morphological and syntactic analysis. *Linguistic Structures Processing*, 5:131–234.

Scott Kirkpatrick. 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proc. of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. Indonesian morphology tool (morphind): Towards an indonesian corpus. In *Proc. of SFCM*. Springer.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Jackson L. Lee and John A. Goldsmith. 2016. Linguistica 5: Unsupervised learning of linguistic structure. In *Proc. of NAACL*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proc. of IJCAI*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proc. of EMNLP*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of NAACL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proc. of CoNLL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and mikko kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proc. of EACL*.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. Comparative study of minimally supervised morphological segmentation. *Computational Linguistics*, 42(1):91–120.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL*, 3:359–373.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proc. of AAAI*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Britta Zeller, Jan Šnajder, and Sebastian Padó. 2013. Derivbase: Inducing and evaluating a derivational morphology resource for german. In *Proc. of ACL*.

Exploiting Mutual Benefits between Syntax and Semantic Roles using Neural Network

Peng Shi[‡*] **Zhiyang Teng**[†] **Yue Zhang**[†]
[†]Singapore University of Technology and Design (SUTD)
[‡]Zhejiang University, China
impavidity@zju.edu.cn
zhiyang_teng@mymail.sutd.edu.sg, yue_zhang@sutd.edu.sg

Abstract

We investigate mutual benefits between syntax and semantic roles using neural network models, by studying a parsing→SRL pipeline, a SRL→parsing pipeline, and a simple joint model by embedding sharing. The integration of syntactic and semantic features gives promising results in a Chinese Semantic Treebank, demonstrating large potentials of neural models for joint parsing and semantic role labeling.

1 Introduction

The correlation between syntax and semantics has been a fundamental problem in natural language processing (Steedman, 2000). As a shallow semantic task, semantic role labeling (SRL) models have traditionally been built upon syntactic parsing results (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Punyakanok et al., 2005). It has been shown that parser output features play a crucial role for accurate SRL (Pradhan et al., 2005; Surdeanu et al., 2007).

On the reverse direction, semantic role features have been used to improve parsing (Boxwell et al., 2010). Existing methods typically use semantic features to rerank n-best lists of syntactic parsing models (Surdeanu et al., 2008; Hajič et al., 2009). There has also been attempts to learn syntactic parsing and semantic role labeling models jointly, but most such efforts have led to negative results (Sutton and McCallum, 2005; Van Den Bosch et al., 2012; Boxwell et al., 2010).

With the rise of deep learning, neural network models have been used for semantic role labeling (Collobert et al., 2011). Recently, it has been shown that a neural semantic role labeler can give state-of-the-art accuracies without using parser output features, thanks to the use of recurrent neural network structures that automatically capture syntactic information (Zhou and Xu, 2015; Wang et al., 2015). In the parsing domain, neural network models have also been shown to give state-of-the-art results recently (Dyer et al., 2015; Weiss et al., 2015; Zhou et al., 2015).

The availability of parser-independent neural SRL models allows parsing and SRL to be performed by both parsing→SRL and SRL→parsing pipelines, and gives rise to the interesting research question whether mutual benefits between syntax and semantic roles can be better exploited under the neural setting. Different from traditional models that rely on manual feature combinations for joint learning tasks (Sutton and McCallum, 2005; Zhang and Clark, 2008a; Finkel and Manning, 2009; Lewis et al., 2015), neural network models induce non-linear feature combinations automatically from input word and Part-of-Speech (POS) embeddings. This allows more complex feature sharing between multiple tasks to be achieved effectively (Collobert et al., 2011).

We take a first step[1] in such investigation by cou-

[*]Work done while the first author was visiting SUTD.

[1]Recently, Swayamdipta et al. (2016) independently proposed a similar idea to perform joint syntactic and semantic dependency parsing. Their work mainly focuses on extending actions of a greedy transition-based parser to support the joint task, achieving good performance on an English shared task, while we use a neural network for multi-task learning and we

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 968–974,
Austin, Texas, November 1-5, 2016.

pling a state-of-the-art neural semantic role labeler (Wang et al., 2015) and a state-of-the-art neural parser (Dyer et al., 2015). First, we propose a novel parsing→SRL pipeline using a tree Long Short-Term Memory (LSTM) model (Tai et al., 2015) to represent parser outputs, before feeding them to the neural SRL model as inputs. Second, we investigate a SRL→parsing pipeline, using semantic role label embeddings to enrich parser features. Third, we build a joint training model by embedding sharing, which is the most shallow level of parameter sharing between deep neural networks. This simple strategy is immune to significant differences between the network structures of the two models, which prevent direct sharing of deeper network parameters. We choose a Chinese semantic role treebank (Qiu et al., 2016) for preliminary experiments, which offers consistent dependency between syntax and semantic role representations, thereby facilitates the application of standard LSTM models. Results show that the methods give improvements to both parsing and SRL accuracies, demonstrating large potentials of neural networks for the joint task.

Our contributions can be summarized as:

• We show that the state-of-the-art LSTM semantic role labeler of Zhou and Xu (2015), which has been shown to be able to induce syntactic features automatically, can still be improved using parser output features via tree LSTM (Tai et al., 2015);

• We show that state-of-the-art neural parsing can be improved by using semantic role features;

• We show that parameter sharing between neural parsing and SRL improves both sub tasks, which is in line with the observation of Collobert et al. (2011) between POS tagging, chunking and SRL.

• Our code and all models are released at https://github.com/ShiPeng95/ShallowJoint.

2 Models

2.1 Semantic Role Labeler

We employ the SRL model of Wang et al. (2015), which uses a bidirectional Long Short-term Memory (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005; Graves et al., 2013) for sequential labeling.

work on a Chinese dataset.

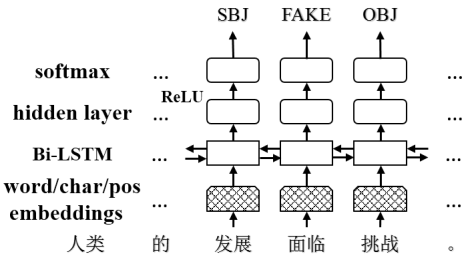


Figure 1: Bi-LSTM Semantic Role Labeler

Given the sentence "人类(human) 的(de) 发展(development) 面临(face) 挑战(challenge)", the structure of the model is shown in Figure 1. For each word w_t, the LSTM model uses a set of vectors to control information flow: an input gate i_t, a forget gate f_t, a memory cell c_t, an output gate o_t, and a hidden state h_t. The computation of each vector is as follows:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + V^{(i)}c_{t-1} + b^{(i)})$$
$$f_t = 1.0 - i_t$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)})$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + V^{(o)}c_t + b^{(o)})$$
$$h_t = o_t \odot \tanh(c_t)$$

Here σ denotes component-wise sigmoid function and $\odot$ is component-wise multiplication.

The representation of x_t is from four sources: an embedding for the word w_t, two hidden states of the last LSTM cells in a character-level bidirectional LSTM (Ballesteros et al., 2015) (denoted as $\overrightarrow{ch_t}$ and $\overleftarrow{ch_t}$, respectively), and a learned vector Part-of-Speech (POS) representation (pos_t). A linear transformation is applied to the vector representations before feeding them into a component-wise ReLU (Nair and Hinton, 2010) function.

$$x_t = \max\{0, V^{(x)}[w_t; \overrightarrow{ch_t}; \overleftarrow{ch_t}; pos_t] + b^{(x)}\}$$

The hidden state vectors at the t-th word from both directions (denote as $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, respectively) are passed through the ReLU function, before a softmax layer for semantic role detection.

2.2 Stack-LSTM Dependency Parser

We employ the Stack-LSTM model of Dyer et al. (2015) for dependency parsing. As shown in Figure 2, it uses a buffer (B) to order input words, a stack (S) to store partially constructed syntactic trees, and

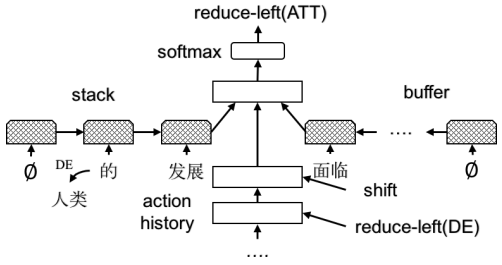


Figure 2: Stack-LSTM Parser

takes the following types of actions to build trees from input.

• **SHIFT**, which pops the top element off the buffer, pushing it into stack.

• **REDUCE-LEFT/REDUCE-RIGHT**, which pop the top two elements off the stack, pushing back the composition of the two elements with a dependent relation.

The parser is initialized by pushing input embeddings into the buffer in the reverse order. The representation of the token is same as the previous bidirectional LSTM (Bi-LSTM) model. The buffer (B), stack (S) and action history sequence (A) are all represented by LSTMs, with S being represented by a novel stack LSTM. At a time step t, the parser predicts an action according to current parser state p_t:

$$p_t = \max\{0, W^{(parser)}[s_t; b_t; a_t] + d^p\},$$
$$y_t^{(parser)} = softmax(V^{(parser)} p_t + d^y)$$

W, V and d are model parameters.

2.3 DEP→SRL Pipeline

In this pipeline model, we apply Stack-LSTM parsing first and feed the results as additional features for SRL. For each word w_t to the SRL system, the corresponding input becomes,

$$x_t^{(dep)} = \max\{0, V^{(dep)}[w_t; \overrightarrow{ch}_t; \overleftarrow{ch}_t; pos_t; \mathbf{dep_t}]\}$$

where dep_t is the t-th word's dependency information from parser output and $V^{(dep)}$ is a weight matrix. There are multiple ways to define dep_t. A simple method is to use embeddings of the dependency label at w_t. However, this input does not embody full arc information.

We propose a novel way of defining $dep_{\bar{t}}$, by using hidden vector $h_{\bar{t}}$ of a dependency tree LSTM (Tai et al., 2015) at $w_{\bar{t}}$ as $dep_{\bar{t}}$. Given a dependency tree output, we define tree LSTM inputs $x_{\bar{t}}$ in the same way as Section 2.1. The tree LSTM is a bottom-up generalization of the sequence LSTM, with a node $h_{\bar{t}}$ having multiple predecessors $h_{\bar{t}-1}^k$, which corresponding to the syntactic dependents of the word $w_{\bar{t}}$. The computation of $h_{\bar{t}}$ for each $w_{\bar{t}}$ is (unlike t, which is a left-to-right index, $\bar{t}$ is a bottom-up index, still with one $h_{\bar{t}}$ being computed for each $w_{\bar{t}}$):

$$\widetilde{h}_{\bar{t}-1} = \sum_k h_{\bar{t}-1}^k$$
$$i_{\bar{t}} = \sigma(W^{(i)} x_{\bar{t}} + U^{(i)} \widetilde{h}_{\bar{t}-1} + b^{(i)})$$
$$f_{\bar{t}^k = \sigma(W^{(f)} x_{\bar{t}} + U^{(f)} h_{\bar{t}-1}^k + b^{(f)})}$$
$$c_{\bar{t}=\sum_k f_{\bar{t}}^k \odot c_{\bar{t}-1}^k + i_{\bar{t}} \odot \tanh(W^{(u)} x_{\bar{t}} + U^{(u)} \widetilde{h}_{\bar{t}-1} + b^{(u)})}$$
$$o_{\bar{t}=\sigma(W^{(o)} x_{\bar{t}} + U^{(o)} \widetilde{h}_{\bar{t}-1} + b^{(o)})}$$
$$h_{\bar{t}=o_{\bar{t}} \odot \tanh(c_{\bar{t}})}$$

For training, we construct a corpus with all words being associated with automatic dependency labels by applying 10-fold jackknifing.

2.4 SRL→DEP Pipeline

In this pipeline model, we conduct SRL first, and feed the output semantic roles to the Stack-LSTM parser in the token level. The representation of a token becomes:

$$x_t^{(srl)} = \max\{0, V^{(srl)}[w_t; \overrightarrow{ch}_t; \overleftarrow{ch}_t; pos_t; \mathbf{srl_t}]\}$$

where srl_t is the t-th word's predicted semantic role embedding and $V^{(srl)}$ is a weight matrix.

For training, we construct a training corpus with automatically tagged semantic role labels by using 10-fold jackknifing.

2.5 Joint Model by Parameter Sharing

The structure of the joint system is shown in Figure 3. Here the parser and semantic role labeler are coupled in the embedding layer, sharing the vector lookup tables for characters, words and POS. More specifically, the Bi-LSTM model of Section 2.1 and the Stack-LSTM model of Section 2.2 are used for the SRL task and the parsing task, respectively. The Bi-LSTM labeler and Stack-LSTM parser share the embedding layer. During training, we maximize the

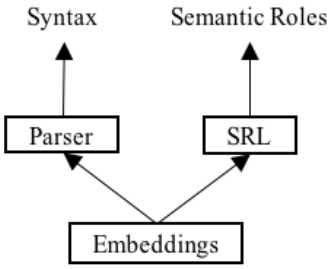


Figure 3: Joint Multi-task Model

sum of log-likelihood for the two different tasks. The loss from the semantic role labeler and the parser both propagate to the embedding layer, resulting in a better vector representation of each token, which benefits both tasks at the same time. On the other hand, due to different neural structures, there is no sharing of other parameters. The joint model offers the simplest version of *shared training* (Collobert et al., 2011), but does not employ shared decoding (Sutton and McCallum, 2005; Zhang and Clark, 2008b). Syntax and semantic roles are assigned separately, avoiding error propagation.

3 Experiments

3.1 Experimental Settings

Datasets We choose Chinese Semantic Treebank (Qiu et al., 2016) for our experiments. Similar to the CoNLL corpora (Surdeanu et al., 2008; Hajič et al., 2009) and different from PropBank (Kingsbury and Palmer, 2002; Xue and Palmer, 2005), it is a dependency-based corpus rather than a constituent-based corpus. The corpus contains syntactic dependency arc and semantic role annotations in a consistent form, hence facilitating the joint task. We follow the standard split for the training, development and test sets, as shown in Table 1.

Training Details. There is a large number of singletons in the training set and a large number of out-of-vocabulary (OOV) words in the development set. We use the mechanism of Dyer et al. (2015) to stochastically set singletons as *UNK* token in each training iteration with a probability p_{unk}. The hyper-parameter p_{unk} is set to 0.2.

For parameters used in Stack-LSTM, we follow Dyer et al. (2015). We set the number of embeddings by intuition, and decide to have the size of word embedding twice as large as that of character embedding, and the size of character embedding larger than the size of POS embedding. More specifically, we fix the size of word embeddings n_w to 64, character embeddings n_{char} to 32, POS embeddings n_{pos} to 30, action embeddings n_{dep} to 30, and semantic role embeddings n_{srl} to 30. The LSTM input size is set to 128 and the LSTM hidden size to 128.

Dataset	Words	**Types**	**Singletons**	**OOV**
Train	280,043	24,866	12,012	-
Dev	23,724	5,492	-	1,505
Test	32,326	6,989	-	1,893

Table 1: Statistics of Chinese Semantic Treebank.

We randomly initialize each parameter to a real value in $[-\sqrt{\frac{6}{r+c}}, \sqrt{\frac{6}{r+c}}]$, where r is the number of input unit and c is the number of output unit (Glorot and Bengio, 2010). To minimize the influence of external information, we did not pretrain the embedding values. In addition, we apply a Gaussian noise $N(0, 0.2)$ to word embeddings during training to prevent overfitting.

We optimize model parameters using stochastic gradient descent with momentum. The same learning rate decay mechanism of Dyer et al. (2015) is used. The best model parameters are selected according to a score metric on the development set. For different tasks, we use different score metrics to evaluate the parameters. Since there are there metrics, F1, UAS and LAS, possibly reported at the same time, we use the weighted average to consider the effect of all metrics when choosing the best model on the dev set. In particular, we use F_1 for SRL, $0.5 \times LAS + 0.5 \times UAS$ for parsing, and $0.5 \times F_1 + 0.25 \times UAS + 0.25 \times LAS$ for the joint task.

3.2 Results

The final results are shown in Table 2, where F_1 represents the F_1-score of semantic roles, and UAS and LAS represent parsing accuracies. The **Bi-LSTM** row represents the bi-directional semantic role labeler, the **S-LSTM** row represents the Stack-LSTM parser, the **DEP→SRL** row represents the dependency parsing → SRL pipeline, the **SRL→DEP** row represents the SRL → dependency parsing pipeline, and the **Joint** row represents the parameter-shared model. For the DEP→SRL pipeline, *lab* and *lstm*

Model	F_1	UAS	LAS
Bi-LSTM	72.71	-	-
S-LSTM	-	84.33	82.10
DEP→SRL(*lab*/*lstm*)	73.00/**74.18**	84.33	82.10
SRL→DEP	72.71	84.75	82.62
Joint	73.84	**85.15**	**82.91**

Table 2: Results. Bi-LSTM and S-LSTM are two baseline models for SRL and parsing, respectively. DEP→SRL and SRL→DEP are two pipeline models. 'Joint' denotes the proposed model for joint parsing and semantic role labeling. *lab* uses only the dependency label as features, while *lstm* applies features extracted from dependency trees using tree LSTMs.

represents the use of dependency label embeddings and tree LSTM hidden vectors for the additional SRL features dep_t, respectively.

Comparison between Bi-LSTM and DEP→SRL shows that slight improvement is brought by introducing dependency label features to the semantic role labeler (72.71→73.00). By introducing full tree information, the *lstm* integration leads to much higher improvements (72.71→74.18). This demonstrates that the LSTM SRL model of Zhou and Xu (2015) can still benefit from parser outputs, despite that it can learn syntactic information independently.

In the reverse direction, comparison between S-LSTM and SRL→DEP shows improvement to UAS/LAS by integrating semantic role features (82.10→82.62). This demonstrates the usefulness of semantic roles to parsing and is consistent with observations on discrete models (Boxwell et al., 2010). To our knowledge, we are the first to report results using a SRL → Parsing pipeline, which is enabled by the neural SRL model.

Using shared embeddings, the joint model gives improvements on both SRL and parsing. The most salient difference between the joint model and the two pipelines is the shared parameter space.

These results are consistent with the finds of Collobert et al. (2011) who show that POS, chunking and semantic role information can bring benefit to each other in joint neural training. In contrast to their results (SRL 74.15→74.29, POS 97.12→97.22, CHUNK 93.37→93.75), we find that parsing and SRL benefit relatively more from each other (SRL 72.72→73.84, DEP 84.33→85.15). This is intuitive because parsing offers deeper syntactic information compared to POS and shallow syntactic chunking.

4 Conclusion

We investigated the mutual benefits between dependency syntax and semantic roles using two state-of-the-art LSTM models, finding that both can be further improved. In addition, simple multitask learning is also effective. These results demonstrate potentials for deeper joint neural models between these tasks.

Acknowledgments

Yue Zhang is the corresponding author. This research is supported by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education. We appreciate anonymous reviewers for their insightful comments.

References

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Stephen A Boxwell, Dennis N Mehay, and Chris Brew. 2010. What a parser can learn from a semantic role labeler and vice versa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 736–744. Association for Computational Linguistics.

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

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. ACL*.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

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

A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

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

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. Citeseer.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* ccg parsing and semantic role labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal, September. Association for Computational Linguistics.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*, volume 5, pages 1117–1123.

Likun Qiu, Yue Zhang, and Meishan Zhang. 2016. Dependency tree representations of predicate-argument structures. In *Proc. AAAI*.

Mark Steedman. 2000. *The syntactic process*, volume 24. MIT Press.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, pages 105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 225–228. Association for Computational Linguistics.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Antal Van Den Bosch, Roser Morante, and Sander Canisius. 2012. Joint learning of dependency parsing and semantic role labeling. *Computational Linguistics in the Netherlands Journal*, 2:97–117.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1626–1631.

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

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and pos tagging using a single perceptron. In *ACL*, pages 888–896.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222.

The Effects of Data Size and Frequency Range on Distributional Semantic Models

Magnus Sahlgren
Gavagai and SICS
Slussplan 9, Box 1263
111 30 Stockholm, 164 29 Kista
Sweden
mange@[gavagai|sics].se

Alessandro Lenci
University of Pisa
via Santa Maria 36
56126 Pisa
Italy
alessandro.lenci@unipi.it

Abstract

This paper investigates the effects of data size and frequency range on distributional semantic models. We compare the performance of a number of representative models for several test settings over data of varying sizes, and over test items of various frequency. Our results show that neural network-based models underperform when the data is small, and that the most reliable model over data of varying sizes and frequency ranges is the inverted factorized model.

1 Introduction

Distributional Semantic Models (DSMs) have become a staple in natural language processing. The various parameters of DSMs — e.g. size of context windows, weighting schemes, dimensionality reduction techniques, and similarity measures — have been thoroughly studied (Weeds et al., 2004; Sahlgren, 2006; Riordan and Jones, 2011; Bullinaria and Levy, 2012; Levy et al., 2015), and are now well understood. The impact of various processing models — matrix-based models, neural networks, and hashing methods — have also enjoyed considerable attention lately, with at times conflicting conclusions (Baroni et al., 2014; Levy et al., 2015; Schnabel et al., 2015; Österlund et al., 2015; Sahlgren et al., 2016). The consensus interpretation of such experiments seems to be that the choice of processing model is less important than the parameterization of the models, since the various processing models all result in more or less equivalent DSMs (provided that the parameterization is comparable).

One of the least researched aspects of DSMs is the effect on the various models of data size and frequency range of the target items. The only previous work in this direction that we are aware of is Asr et al. (2016), who report that on small data (the CHILDES corpus), simple matrix-based models outperform neural network-based ones. Unfortunately, Asr et al. do not include any experiments using the same models applied to bigger data, making it difficult to compare their results with previous studies, since implementational details and parameterization will be different.

There is thus still a need for a consistent and fair comparison of the performance of various DSMs when applied to data of varying sizes. In this paper, we seek an answer to the question: **which DSM should we opt for if we only have access to limited amounts of data?** We are also interested in the related question: **which DSM should we opt for if our target items are infrequent?** The latter question is particularly crucial, since one of the major assets of DSMs is their applicability to create semantic representations for ever-expanding vocabularies from text feeds, in which new words may continuously appear in the low-frequency ranges.

In the next section, we introduce the contending DSMs and the general experiment setup, before turning to the experiments and our interpretation of the results. We conclude with some general advice.

2 Distributional Semantic Models

One could classify DSMs in many different ways, such as the type of context and the method to build distributional vectors. Since our main goal here is

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 975–980,
Austin, Texas, November 1-5, 2016.

to gain an understanding of the effect of data size and frequency range on the various models, we focus primarily on the differences in processing models, hence the following typology of DSMs.

Explicit matrix models

We here include what could be referred to as *explicit* models, in which each vector dimension corresponds to a specific context (Levy and Goldberg, 2014). The baseline model is a simple co-occurrence matrix F (in the following referred to as CO for Co-Occurrence). We also include the model that results from applying Positive Pointwise Mutual Information (PPMI) to the co-occurrence matrix. PPMI is defined as simply discarding any negative values of the PMI, computed as:

$$\text{PMI}(a, b) = \log \frac{f_{ab} \times T}{f_a f_b} \quad (1)$$

where f_{ab} is the co-occurrence count of word a and word b, f_a and f_b are the individual frequencies of the words, and T is the number of tokens in the data.[1]

Factorized matrix models

This type of model applies an additional factorization of the weighted co-occurrence counts. We here include two variants of applying Singular Value Decomposition (SVD) to the PPMI-weighting co-occurrence matrix; one version that discards all but the first couple of hundred latent dimensions (TSVD for *truncated* SVD), and one version that instead *removes* the first couple of hundred latent dimensions (ISVD for *inverted* SVD). SVD is defined in the standard way:

$$F = U\Sigma V^T \quad (2)$$

where U holds the eigenvectors of F, Σ holds the eigenvalues, and $V \in U(w)$ is a unitary matrix mapping the original basis of F into its eigenbasis. Since V is redundant due to invariance under unitary transformations, we can represent the factorization of $\hat{F}$ in its most compact form $\hat{F} \equiv U\Sigma$.

[1]We also experimented with *smoothed* PPMI, which raises the context counts to the power of α and normalizes them (Levy et al., 2015), thereby countering the tendency of mutual information to favor infrequent events: $f(b) = \frac{\#(b)^\alpha}{\sum_b \#(b)^\alpha}$, but it did not lead to any consistent improvements compared to PPMI.

Hashing models

A different approach to reduce the dimensionality of DSMs is to use a hashing method such as Random Indexing (RI) (Kanerva et al., 2000), which accumulates distributional vectors $\vec{d}(a)$ in an online fashion:

$$\vec{d}(a) \leftarrow \vec{d}(a_i) + \sum_{j=-c, j\neq 0}^{c} w(x^{(i+j)})\pi^j \vec{r}(x^{(i+j)}) \quad (3)$$

where c is the extension of the context window, $w(b)$ is a weight that quantifies the importance of context term b,[2] $\vec{r}_d(b)$ is a *sparse random index vector* that acts as a fingerprint of context term b, and π^j is a permutation that rotates the random index vectors one step to the left or right, depending on the position of the context items within the context windows, thus enabling the model to take word order into account (Sahlgren et al., 2008).

Neural network models

There are many variations of DSMs that use neural networks as processing model, ranging from simple recurrent networks (Elman, 1990) to more complex deep architectures (Collobert and Weston, 2008). The incomparably most popular neural network model is the one implemented in the `word2vec` library, which uses the softmax for predicting b given a (Mikolov et al., 2013):

$$p(b|a) = \frac{\exp(\vec{b} \cdot \vec{a})}{\sum_{b' \in C} \exp(\vec{b'} \cdot \vec{a})} \quad (4)$$

where C is the set of context words, and $\vec{b}$ and $\vec{a}$ are the vector representations for the context and target words, respectively. We include two versions of this general model; Continuous Bag of Words (CBOW) that predicts a word based on the context, and Skip-Gram Negative Sampling (SGNS) that predicts the context based on the current word.

3 Experiment setup

Since our main focus in this paper is the performance of the above-mentioned DSMs on data of

[2]We use $w(b) = e^{-\lambda \cdot \frac{f(b)}{V}}$ where $f(b)$ is the frequency of context item b, V is the total number of unique context items seen thus far (i.e. the current size of the growing vocabulary), and λ is a constant that we set to 60 (Sahlgren et al., 2016).

varying sizes, we use one big corpus as starting point, and split the data into bins of varying sizes. We opt for the ukWaC corpus (Ferraresi et al., 2008), which comprises some 1.6 billion words after tokenization and lemmatization. We produce subcorpora by taking the first 1 million, 10 million, 100 million, and 1 billion words.

Since the co-occurrence matrix built from the 1 billion-word ukWaC sample is very big (more than 4,000,000 $\times$ 4,000,000), we prune the co-occurrence matrix to 50,000 dimensions before the factorization step by simply removing infrequent context items.[3] As comparison, we use 200 dimensions for TSVD, 2,800 (3,000-200) dimensions for ISVD, 2,000 dimensions for RI, and 200 dimensions for CBOW and SGNS. These dimensionalities have been reported to perform well for the respective models (Landauer and Dumais, 1997; Sahlgren et al., 2008; Mikolov et al., 2013; Österlund et al., 2015). All DSMs use the same parameters as far as possible with a narrow context window of ± 2 words, which has been shown to produce good results in semantic tasks (Sahlgren, 2006; Bullinaria and Levy, 2012).

We use five standard benchmark tests in these experiments; two multiple-choice vocabulary tests (the TOEFL synonyms and the ESL synonyms), and three similarity/relatedness rating benchmarks (SimLex-999 (SL) (Hill et al., 2015), MEN (Bruni et al., 2014), and Stanford Rare Words (RW) (Luong et al., 2013)). The vocabulary tests measure the synonym relation, while the similarity rating tests measure a broader notion of semantic similarity (SL and RW) or relatedness (MEN).[4] The results for the vocabulary tests are given in accuracy (i.e., percentage of correct answers), while the results for the similarity tests are given in Spearman rank correlation.

4 Comparison by data size

Table 1 summarizes the results over the different test settings. The most notable aspect of these results is that the neural networks models do not produce competitive results for the smaller data, which corroborates the results by Asr et al. (2016). The best results for the smallest data are produced by the factorized models, with both TSVD and ISVD producing top scores in different test settings. It should be noted, however, that even the top scores for the smallest data set are substandard; only two models (PPMI and TSVD) manage to beat the random baseline of 25% for the TOEFL tests, and none of the models manage to beat the random baseline for the ESL test.

The ISVD model produces consistently good results; it yields the best overall results for the 10 mil-

DSM	TOEFL	ESL	SL	MEN	RW
1 million words					
CO	17.50	**20.00**	−1.64	10.72	−3.96
PPMI	26.25	18.00	8.28	21.49	−2.57
TSVD	**27.50**	**20.00**	4.43	**22.15**	−1.56
ISVD	22.50	14.00	**14.33**	19.74	**5.31**
RI	20.00	16.00	5.65	17.94	1.92
SGNS	15.00	8.00	3.64	12.34	1.46
CBOW	15.00	10.00	−0.16	11.59	1.39
10 million words					
CO	40.00	22.00	4.77	15.20	0.95
PPMI	52.50	38.00	26.44	39.83	4.00
TSVD	38.75	30.00	19.27	34.33	5.53
ISVD	45.00	**44.00**	**30.19**	**44.21**	**9.88**
RI	**47.50**	24.00	20.44	34.56	3.32
SGNS	43.75	42.00	28.30	26.59	2.38
CBOW	40.00	30.00	22.22	28.33	3.04
100 million words					
CO	45.00	30.00	10.00	19.36	3.12
PPMI	**66.25**	54.00	33.75	46.74	15.05
TSVD	46.25	34.00	25.11	42.49	13.00
ISVD	66.25	**66.00**	**40.98**	**54.55**	**21.27**
RI	55.00	48.00	32.31	45.71	10.15
SGNS	65.00	58.00	40.75	52.83	11.73
CBOW	61.25	46.00	36.15	48.30	15.62
1 billion words					
CO	55.00	40.00	11.85	21.83	6.82
PPMI	71.25	54.00	35.69	52.95	24.29
TSVD	56.25	46.00	31.36	52.05	13.35
ISVD	71.25	**66.00**	**44.77**	60.11	**28.46**
RI	61.25	50.00	35.35	50.51	18.58
SGNS	**76.25**	**66.00**	41.94	**67.03**	24.50
CBOW	75.00	56.00	38.31	59.84	22.80

Table 1: Results for DSMs trained on data of varying sizes.

[3] Such drastic reduction has a negative effect on the performance of the factorized methods for the 1 billion word data, but unfortunately is necessary for computational reasons.

[4] It is likely that the results on the similarity tests could be improved by using a wider context window, but such improvement would probably be consistent across all models, and is thus outside the scope of this paper.

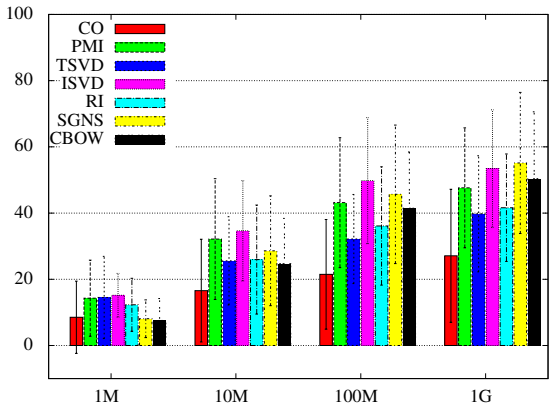


Figure 1: Average results and standard deviation over all tests.

lion and 100 million-word data, and is competitive with SGNS on the 1 billion word data. Figure 1 shows the average results and their standard deviations over all test settings.[5] It is obvious that there are no huge differences between the various models, with the exception of the baseline CO model, which consistently underperforms. The TSVD and RI models have comparable performance across the different data sizes, which is systematically lower than the PPMI model. The ISVD model is the most consistently good model, with the neural network-based models steadily improving as data becomes bigger.

Looking at the different datasets, SL and RW are the hardest ones for all the models. In the case of SL, this confirms the results in (Hill et al., 2015), and might be due to the general bias of DSMs towards semantic relatedness, rather than genuine semantic similarity, as represented in SL. The substandard performance on RW might instead be due to the low frequency of the target items. It is interesting to note that these are benchmark tests in which neural models perform the worst even when trained on the largest data.

5 Comparison by frequency range

In order to investigate how each model handles different frequency ranges, we split the test items into three different classes that contain about a third of the frequency mass of the test items each. This split was produced by collecting all test items into a common vocabulary, and then sorting this vocabulary by its frequency in the ukWaC 1 billion-word corpus. We split the vocabulary into 3 equally large parts; the HIGH range with frequencies ranging from 3,515,086 ("do") to 16,830 ("organism"), the MEDIUM range with frequencies ranging between 16,795 ("desirable") and 729 ("prickly"), and the LOW range with frequencies ranging between 728 ("boardwalk") to hapax legomenon. We then split each individual test into these three ranges, depending on the frequencies of the test items. Test pairs were included in a given frequency class if and only if both the target and its relatum occur in the frequency range for that class. For the constituent words in the test item that belong to different frequency ranges, which is the most common case, we use a separate MIXED class. The resulting four classes contain 1,387 items for the HIGH range, 656 items for the MEDIUM range, 350 items for the LOW range, and 3,458 items for the MIXED range.[6]

Table 2 (next side) shows the average results over the different frequency ranges for the various DSMs trained on the 1 billion-word ukWaC data. We also include the highest and lowest individual test scores (signified by ↑ and ↓), in order to get an idea about the consistency of the results. As can be seen in the table, the most consistent model is ISVD, which produces the best results in both the MEDIUM and MIXED frequency ranges. The neural network models SGNS and CBOW produce the best results in the HIGH and LOW range, respectively, with CBOW clearly outperforming SGNS in the latter case. The major difference between these models is that CBOW predicts a word based on a context, while SGNS predicts a context based on a word. Clearly, the former approach is more beneficial for low-frequent items.

The PPMI, TSVD and RI models perform similarly across the frequency ranges, with RI producing somewhat lower results in the MEDIUM range, and TSVD producing somewhat lower results in the LOW range. The CO model underperforms in all frequency ranges. Worth noting is the fact that all models that are based on an explicit matrix (i.e. CO,

[5] Although rank correlation is not directly comparable with accuracy, they are both bounded between zero and one, which means we can take the average to get an idea about overall performance.

[6] 233 test terms did not occur in the 1 billion-word corpus.

DSM	HIGH	MEDIUM	LOW	MIXED
CO	32.61 (↑62.5,↓04.6)	35.77 (↑66.6,↓21.2)	12.57 (↑35.7,↓00.0)	27.14 (↑56.6,↓07.9)
PPMI	55.51 (↑75.3,↓28.0)	57.83 (↑88.8,↓18.7)	25.84 (↑50.0,↓00.0)	47.73 (↑83.3,↓27.1)
TSVD	50.52 (↑70.9,↓23.2)	54.75 (↑77.9,↓24.1)	17.85 (↑50.0,↓00.0)	41.08 (↑56.6,↓19.6)
ISVD	63.31 (↑87.5,↓36.5)	**69.25** (↑88.8,↓46.3)	10.94 (↑16.0,↓00.0)	**57.24** (↑83.3,↓33.0)
RI	53.11 (↑62.5,↓30.1)	48.02 (↑72.2,↓20.4)	23.29 (↑39.0,↓00.0)	46.39 (↑66.6,↓21.0)
SGNS	**68.81** (↑87.5,↓36.4)	62.00 (↑83.3,↓27.4)	18.76 (↑42.8,↓00.0)	56.93 (↑83.3,↓30.2)
CBOW	62.73 (↑81.2,↓31.9)	59.50 (↑83.3,↓32.4)	**27.13** (↑78.5,↓00.0)	52.21 (↑76.6,↓25.9)

Table 2: Average results for DSMs over four different frequency ranges for the items in the TOEFL, ESL, SL, MEN, and RW tests. All DSMs are trained on the 1 billion words data.

PPMI, TSVD and ISVD) produce better results in the MEDIUM range than in the HIGH range.

The arguably most interesting results are in the LOW range. Unsurprisingly, there is a general and significant drop in performance for low frequency items, but with interesting differences among the various models. As already mentioned, the CBOW model produces the best results, closely followed by PPMI and RI. It is noteworthy that the low-dimensional embeddings of the CBOW model only gives a modest improvement over the high-dimensional explicit vectors of PPMI. The worst results are produced by the ISVD model, which scores even lower than the baseline CO model. This might be explained by the fact that ISVD removes the latent dimensions with largest variance, which are arguably the most important dimensions for very low-frequent items. Increasing the number of latent dimensions with high variance in the ISVD model improves the results in the LOW range (16.59 when removing only the top 100 dimensions).

6 Conclusion

Our experiments confirm the results of Asr et al. (2016), who show that neural network-based models are suboptimal to use for smaller amounts of data. On the other hand, our results also show that none of the standard DSMs work well in situations with small data. It might be an interesting novel research direction to investigate how to design DSMs that are applicable to small-data scenarios.

Our results demonstrate that the inverted factorized model (ISVD) produces the most robust results over data of varying sizes, and across several different test settings. We interpret this finding as further corroborating the results of Bullinaria and Levy (2012), and Österlund et al. (2015), with the conclusion that the inverted factorized model is a robust competitive alternative to the widely used SGNS and CBOW neural network-based models.

We have also investigated the performance of the various models on test items in different frequency ranges, and our results in these experiments demonstrate that all tested models perform optimally in the medium-to-high frequency ranges. Interestingly, all models based on explicit count matrices (CO, PPMI, TSVD and ISVD) produce somewhat better results for items of medium frequency than for items of high frequency. The neural network-based models and ISVD, on the other hand, produce the best results for high-frequent items.

None of the tested models perform optimally for low-frequent items. The best results for low-frequent test items in our experiments were produced using the CBOW model, the PPMI model and the RI model, all of which uses weighted context items without any explicit factorization. By contrast, the ISVD model underperforms significantly for the low-frequent items, which we suggest is an effect of removing latent dimensions with high variance.

This interpretation suggests that it might be interesting to investigate *hybrid models* that use different processing models — or at least different parameterizations — for different frequency ranges, and for different data sizes. We leave this as a suggestion for future research.

7 Acknowledgements

This research was supported by the Swedish Research Council under contract 2014-28199.

References

Fatemeh Asr, Jon Willits, and Michael Jones. 2016. Comparing predictive and co-occurrence based models of lexical semantics trained on child-directed speech. In *Proceedings of CogSci*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

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

John Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior Research Methods*, 44:890–907.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. *Proceedings of WAC-4*, pages 47–54.

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

Pentti Kanerva, Jan Kristofersson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of CogSci*, page 1036.

Thomas K Landauer and Susan T. Dumais. 1997. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL*, pages 171–180.

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

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113.

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

Arvid Österlund, David Ödling, and Magnus Sahlgren. 2015. Factorization of latent variables in distributional semantic models. In *Proceedings of EMNLP*, pages 227–231.

Brian Riordan and Michael N. Jones. 2011. Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. *Topics in Cognitive Science*, 3(2):303–345.

Magnus Sahlgren, Anders Holst, and Pentti Kanerva. 2008. Permutations as a means to encode order in word space. In *Proceedings of CogSci*, pages 1300–1305.

Magnus Sahlgren, Amaru Cuba Gyllensten, Fredrik Espinoza, Ola Hamfors, Anders Holst, Jussi Karlgren, Fredrik Olsson, Per Persson, and Akshay Viswanathan. 2016. The Gavagai Living Lexicon. In *Proceedings of LREC*.

Magnus Sahlgren. 2006. *The Word-Space Model*. Phd thesis, Stockholm University.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP*, pages 298–307.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of COLING*, pages 1015–1021.

Multi-Granularity Chinese Word Embedding

Rongchao Yin[†‡], Quan Wang[†‡] , Rui Li[†‡], Peng Li[†‡*], Bin Wang[†‡]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[‡]University of Chinese Academy of Sciences, Beijing 100049, China
{yinrongchao,wangquan,lirui,lipeng,wangbin}@iie.ac.cn

Abstract

This paper considers the problem of learning Chinese word embeddings. In contrast to English, a Chinese word is usually composed of characters, and most of the characters themselves can be further divided into components such as radicals. While characters and radicals contain rich information and are capable of indicating semantic meanings of words, they have not been fully exploited by existing word embedding methods. In this work, we propose *multi-granularity embedding* (MGE) for Chinese words. The key idea is to make full use of such word-character-radical composition, and enrich word embeddings by further incorporating finer-grained semantics from characters and radicals. Quantitative evaluation demonstrates the superiority of MGE in word similarity computation and analogical reasoning. Qualitative analysis further shows its capability to identify finer-grained semantic meanings of words.

1 Introduction

Word embedding, also known as distributed word representation, is to represent each word as a real-valued low-dimensional vector, through which the semantic meaning of the word can be encoded. Recent years have witnessed tremendous success of word embedding in various NLP tasks (Bengio et al., 2006; Mnih and Hinton, 2009; Collobert et al., 2011; Zou et al., 2013; Kim, 2014; Liu et al., 2015; Iyyer et al., 2015). The basic idea behind is to learn the distributed representation of a word using its context. Among existing approaches, the continuous bag-of-words model (CBOW) and Skip-Gram model are simple and effective, capable of learning word embeddings efficiently from large-scale text corpora (Mikolov et al., 2013a; Mikolov et al., 2013b).

Besides the success in English, word embedding has also been demonstrated to be extremely useful for Chinese language processing (Xu et al., 2015; Yu et al., 2015; Zhou et al., 2015; Zou et al., 2013). The work on Chinese generally follows the same idea as on English, i.e., to learn the embedding of a word on the basis of its context. However, in contrast to English where words are usually taken as basic semantic units, Chinese words may have a complicated composition structure of their semantic meanings. More specifically, a Chinese word is often composed of several characters, and most of the characters themselves can be further divided into components such as radicals (部首).[1] Both characters and radicals may suggest the semantic meaning of a word, regardless of its context. For example, the Chinese word "吃饭 (have a meal)" consists of two characters " 吃 (eat)" and "饭 (meal)", where " 吃 (eat)" has the radical of "口 (mouth)", and "饭 (meal)" the radical of " 饣 (food)". The semantic meaning of "吃饭" can be revealed by the constituent characters as well as their radicals.

Despite being the linguistic nature of Chinese and containing rich semantic information, such word-character-radical composition has not been fully exploited by existing approaches. Chen et al. (2015) introduced a character-enhanced word embedding model (CWE), which learns embeddings jointly for words and characters but ignores radicals. Sun et al. (2014) and Li et al. (2015) utilized radical information to learn better character embeddings. Similarly, Shi et al. (2015) split characters into small components based on the Wubi method,[2] and took into account those components during the learning process. In their work, however, embeddings are learned only for characters. For a word, the embedding is generated by simply combining the embeddings of the constituent characters. Since not all Chinese word-

*Corresponding author: Peng Li.

[1] https://en.wikipedia.org/wiki/Radical_(Chinese_characters)
[2] https://en.wikipedia.org/wiki/Wubi_method

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 981–986,
Austin, Texas, November 1-5, 2016.

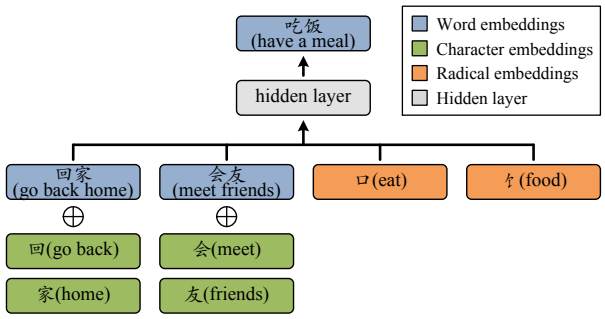


Figure 1: A simple illustration of MGE, where embeddings are learned jointly for words, characters, and radicals. Given a sequence of words {" 回家 (go back home)", " 吃饭 (have a meal)", " 会友 (meet friends)"}, MGE predicts the central word "吃饭" by using 1) the embedding composed by each context word and its constituent characters, and 2) the embedding associated with each radical detected in the target word.

s are semantically compositional (e.g., transliterated words such as "苏打 (soda)"), embeddings obtained in this way may be of low quality for these words.

In this paper, aiming at making full use of the semantic composition in Chinese, we propose *multi-granularity embedding* (MGE) which learns embeddings jointly for words, characters, and radicals. The framework of MGE is sketched in Figure 1. Given a word, we learn its embedding on the basis of 1) the context words (blue bars in the figure), 2) their constituent characters (green bars), and 3) the radicals found in the target word (orange bars). Compared to utilizing context words alone, MGE enriches the embeddings by further incorporating finer-grained semantics from characters and radicals. Similar ideas of adaptively using multiple levels of embeddings have also been investigated in English recently (Kazuma and Yoshimasa, 2016; Miyamoto and Cho, 2016).

We evaluate MGE with the benchmark tasks of word similarity computation and analogical reasoning, and demonstrate its superiority over state-of-the-art metods. A qualitative analysis further shows the capability of MGE to identify finer-grained semantic meanings of words.

2 Multi-Granularity Word Embedding

This section introduces MGE based on the continuous bag-of-words model (CBOW) (Mikolov et al., 2013b) and the character-enhanced word embedding model (CWE) (Chen et al., 2015).

MGE aims at improving word embedding by leveraging both characters and radicals. We denote the Chinese word vocabulary as $\mathcal{W}$, the character vocabulary as $\mathcal{C}$, and the radical vocabulary as $\mathcal{R}$. Each word $w_i \in \mathcal{W}$ is associated with a vector embedding $\mathbf{w}_i$, each character $c_i \in \mathcal{C}$ a vector embedding $\mathbf{c}_i$, and each radical $r_i \in \mathcal{R}$ a vector embedding $\mathbf{r}_i$. Given a sequence of words $\mathcal{D} = \{w_1, \cdots, w_N\}$, MGE predicts each word $w_i \in \mathcal{D}$ conditioned on 1) context words in a sliding window with size ℓ, denoted as $\mathcal{W}_i = \{w_{i-\ell}, ...w_{i-1}, w_{i+1}, ..., w_{i+\ell}\}$, 2) characters in each context word $w_j \in \mathcal{W}_i$, denoted as $\mathcal{C}_j$, and 3) radicals in the target word w_i, denoted as $\mathcal{R}_i$. See Figure 1 for a simple illustration.

More specifically, given the corpus $\mathcal{D}$, MGE maximizes the overall log likelihood as follows:

$$L(\mathcal{D}) = \sum_{w_i \in \mathcal{D}} \log p(\mathbf{w}_i | \mathbf{h}_i). \quad (1)$$

Here $\mathbf{h}_i$ is a hidden vector composed by the embeddings of context words, constituent characters, and radicals, defined as:

$$\mathbf{h}_i = \frac{1}{2}\Big[\frac{1}{|\mathcal{W}_i|}\sum_{w_j \in \mathcal{W}_i}\Big(\mathbf{w}_j \oplus \frac{1}{|\mathcal{C}_j|}\sum_{c_k \in \mathcal{C}_j}\mathbf{c}_k\Big) + \frac{1}{|\mathcal{R}_i|}\sum_{r_k \in \mathcal{R}_i}\mathbf{r}_k\Big]. \quad (2)$$

For each context word $w_j \in \mathcal{W}_i$, a word-character composition $(\mathbf{w}_j \oplus \frac{1}{|\mathcal{C}_j|}\sum_{c \in \mathcal{C}_j}\mathbf{c})$ is first generated by the embeddings of w_j and its constituent characters $\mathcal{C}_j$. These word-character compositions are then combined with the radical embeddings in $\mathcal{R}_i$ to predict the target word. $|\mathcal{W}_i|$/$|\mathcal{R}_i|$/$|\mathcal{C}_j|$ is the cardinality of $\mathcal{W}_i$/$\mathcal{R}_i$/$\mathcal{C}_j$, and $\oplus$ is the composition operation.[3] Given $\mathbf{h}_i$, the conditional probability $p(\mathbf{w}_i|\mathbf{h}_i)$ is defined by a softmax function:

$$p(\mathbf{w}_i|\mathbf{h}_i) = \frac{\exp(\mathbf{h}_i^\top \mathbf{w}_i)}{\sum_{w_{i'} \in \mathcal{W}} \exp(\mathbf{h}_i^\top \mathbf{w}_{i'})}. \quad (3)$$

We use negative sampling and stochastic gradient descent to solve the optimization problem.

Note that 1) Not all Chinese words are semantically compositional, e.g., transliterated words and entity names. For such words we use neither characters nor radicals. 2) A Chinese character usually plays

[3]There are a variety of options for $\oplus$, e.g., addition and concatenation. This paper follows (Chen et al., 2015) and uses the addition operation.

different roles when it appears at different positions within a word. We follow (Chen et al., 2015) and design a position-based MGE model (MGE+P). The key idea of MGE+P is to keep three embeddings for each character, corresponding to its appearance at the positions of "begin", "middle", and "end". For details, please refer to (Chen et al., 2015).

3 Experiments

We evaluate MGE with the tasks of word similarity computation and analogical reasoning.

3.1 Experimental Setups

We select the Chinese Wikipedia Dump[4] for embedding learning. In preprocessing, we use the THULAC tool[5] to segment the corpus. Pure digit words, non-Chinese words, and words whose frequencies are less than 5 in the corpus are removed. We further crawl from an online Chinese dictionary[6] and build a character-radical index with 20,847 characters and 269 radicals. We use this index to detect the radical of each character in the corpus. As such, we get a training set with 72,602,549 words, 277,200 unique words, 8,410 unique characters, and 256 unique radicals. Finally, we use THULAC to perform Chinese POS tagging on the training set and identify all entity names. For these entity names, neither characters nor radicals are considered during learning. Actually, Chen et al. (2015) categorized non-compositional Chinese words into three groups, i.e., transliterated words, single-morpheme multi-character words, and entity names. In their work, they used a human-annotated corpus, manually determining each word to be split or not. Since human annotation could be time-consuming and labor intensive, we just consider automatically identified entity names.

We compare MGE with CBOW (Mikolov et al., 2013b)[7] and CWE (Chen et al., 2015)[8]. Both CWE and MGE are extensions of CBOW, with the former taking into account characters and the latter further incorporating radical information. We further consider position-based CWE and MGE, denoted as CWE+P and MGE+P, respectively.We follow (Chen et al., 2015) and use the same hyperparameter setting. For all the methods, we set the context window size to 3, and select the embedding dimension k in $\{100, 200\}$. During optimization, we use 10-word negative sampling and fix the initial learning rate to 0.025.

Method	WordSim-239		WordSim-293	
	$k=100$	$k=200$	$k=100$	$k=200$
CBOW	0.4917	0.4971	**0.5667**	**0.5723**
CWE	0.5121	0.5197	0.5511	0.5655
CWE+P	0.4989	0.5026	0.5427	0.5545
MGE	**0.5670**	**0.5769**	0.5555	0.5659
MGE+P	0.5511	0.5572	0.5530	0.5692

Table 1: Results on word similarity computation.

3.2 Word Similarity Computation

This task is to evaluate the effectiveness of embeddings in preserving semantic relatedness between two words. We use the WordSim-240 and WordSim-296 datasets[9] provided by Chen et al. (2015) for evaluation, both containing Chinese word pairs with human-labeled similarity scores. On WordSim-240 there is a pair containing new words (i.e., words that have not appeared in the training set), and on WordSim-296 there are 3 such pairs. We remove these pairs from both datasets, and accordingly get WordSim-239 and WordSim-293.

We compute the Spearman correlation coefficient (Myers et al., 2010) between the similarity scores given by the embedding models and those given by human annotators. For the embedding models, the similarity score between two words is calculated as the cosine similarity between their embeddings. The Spearman correlation coefficient is a nonparametric measure of rank correlation, assessing how well the relationship between two variables can be described. The results are shown in Table 1.

From the results, we can see that 1) On WordSim-239, MGE(+P) performs significantly better than CWE(+P), which in turn outperforms CBOW. This observation demonstrates the superiority of incorporating finer-grained semantics, particularly from radicals. For example, MGE performs much better on word pairs such as "银行 (bank)" and " 钱 (money)", in which the two words share the same radical of "钅 (gold)". 2) On WordSim-293, MGE(+P)

[4] http://download.wikipedia.com/zhwiki
[5] http://thulac.thunlp.org/
[6] http://zd.diyifanwen.com/zidian/bs/
[7] https://code.google.com/p/word2vec/
[8] https://github.com/Leonard-Xu/CWE
[9] https://github.com/Leonard-Xu/CWE/tree/master/data

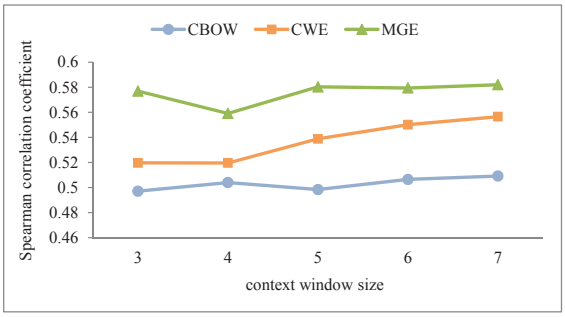


Figure 2: Word similarity computation results with different context window sizes on WordSim-239 ($k = 200$).

performs equally well as CWE(+P), but both are slightly worse than CBOW. The reason may be that WordSim-293 contains a great many of word pairs in which the two words belonging to different domains, e.g., "公鸡 (rooster)" and "航程 (flying range)". These pairs usually get low human-labeled similarity scores. However, splitting the words in such pairs into characters, and further the characters into radicals will not help to effectively identify the *dissimilarity* between them.[10]

We further investigate the influence of the context window size in word similarity computation. Figure 2 gives the results of CBOW, CWE, and MGE on WordSim-239, with the context window size set in $\{3, 4, 5, 6, 7\}$. The results indicate that MGE performs consistently better than CBOW and CWE on this dataset, unaffected by varying the context window size.

3.3 Word Analogical Reasoning

This task evaluates the effectiveness of embeddings in capturing linguistic regularities between pairs of words, in the form of " 伦敦 (London) : 英国 (England) ≈ 巴黎 (Paris) : 法国 (France)". We use the dataset provided by Chen et al. (2015) for evaluation. It contains 1,124 analogies categorized into 3 types: 1) capitals of countries (677 groups); 2) states/provinces of cities (175 groups); and 3) family relations (272 groups). All the words in this dataset are covered by the training set.

Method	Total	Capital	State	Family
CBOW	0.7498	0.8109	0.8400	**0.5294**
CWE	0.7248	0.8375	0.8541	0.3566
CWE+P	0.7391	0.8065	0.8114	0.5147
MGE	0.7524	**0.8804**	0.8686	0.3529
MGE+P	**0.7720**	0.8685	**0.8857**	0.4485

Table 2: Results on word analogical reasoning ($k = 200$).

For each analogy "$a : b \approx c : d$", we create a question "$a : b \approx c :$? ", and predict the answer as: $d^* = \arg\max_{w \in \mathcal{W}} \cos(\mathbf{b} - \mathbf{a} + \mathbf{c}, \mathbf{w})$. Here $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{w}$ are the word embeddings, and $\cos(\cdot, \cdot)$ the cosine similarity. The question is considered to be correctly answered if $d^* = d$. We use accuracy as the evaluation metric, and report the results in Table 2.

The results indicate that 1) MGE(+P) substantially outperforms the baseline methods on almost all types of analogies (except for the Family type). This again demonstrates the superiority of incorporating radical information. 2) For the Capital and State types, all the words are entity names for which neither characters nor radicals are used. MGE(+P) still outperforms the baselines on these two types, showing its capability to learn better embeddings even for non-compositional words. 3) On the Family type, both MGE(+P) and CWE(+P) perform worse than CBOW. This may be caused by the inappropriate decomposition of family words into characters. Consider, for example, the question " 叔叔 (uncle) : 阿姨 (aunt) ≈ 王子 (prince) : ? ". If we split " 王子" into "王 (king)" and "子 (son)", we will more likely to predict " 女王 (queen)" rather than the correct answer "公主 (princess)", since "女王" contains the character " 女 (daughter)" which is usually the antonym of " 子 (son)".

3.4 Case Study

Besides quantitative evaluation, this section further provides qualitative analysis to show in what manner the semantic meaning of a radical, character and word can be captured by their embeddings.

Take the word "游泳 (swimming)" as an example. Table 3 presents the words that are most similar to it (with the highest cosine similarity between their embeddings), discovered by MGE, CWE, and CBOW. The results show that 1) By incorporating the character information, MGE and CWE are capable of

[10] This observation is inconsistent with that reported in (Chen et al., 2015), which shows that CWE outperforms CBOW on WordSim-296. The reason may be that Chen et al. (2015) used a human-annotated corpus for embedding learning, and manually determined each word to be split or not. In contrast, we use the publicly available Chinese Wikipedia data, and automatically segment the corpus and identify entity names (words that are not to be split), without human annotation.

MGE	潜泳(underwater swimming), 畅泳(swimming happily) 爬泳(front crawl swimming), 泳手(swimmer) 泳术(swimming skill), 冬泳(winter swimming) 裸泳(swimming skill), 田径(track and field)
CWE	潜泳(underwater swimming), 畅泳(swimming happily) 爬泳(front crawl swimming), 田径(track and field) 泳手(swimmer), 习泳(learn to swim) 冬泳(winter swimming), 泳术(swimming skill)
CBOW	田径(track and field), 跳高(high jump) 跳水(diving), 跳绳(rope skipping) 划船(boating), 撑竿跳(pole vaulting) 皮划艇(canoeing), 体操(gymnastics)

Table 3: The most similar words to "游泳 (swimming)".

Radical	疒(illness)
Closest characters	佝(rickets) 痫(chronic disease) 偻(bending one's back) 疬(epidemic disease) 痨(tuberculosis) 淬(quenching) 疥(scabies) 痔(hemorrhoids)
Closest words	佝偻(rickets) 癣疥(ringworm scabies) 痘疤(pock) 瘴疬(communicable subtropical disease) 疮痍(traumata) 疮疤(scar) 痲疹(measles) 疱疮(pemphigus)

Table 4: The most similar characters/words to "疒 (illness)".

capturing finer-grained semantics that are more specific to the word. The top words discovered by them are semantically related to "游泳 (swimming)" itself, e.g., "潜泳 (underwater swimming)" and "爬泳 (front crawl swimming)". But the top words discovered by CBOW are just other types of sports in parallel with " 游泳 (swimming)", e.g., " 跳高 (high jump)" and " 跳水 (diving)". 2) MGE performs even better than CWE by further incorporating the radical information. The less relevant word " 田径 (track and field)" is ranked 4th by CWE. But after introducing the radical "氵 (water)", MGE can successfully rank "泳手 (swimmer)", "泳术 (swimming skill)", and " 冬泳 (winter swimming)" before it. All these words contain the radical "氵 (water)" and are more relevant to "游泳 (swimming)".

We further take the radical "疒 (illness)" as an example, and list the most similar characters and words discovered by MGE in Table 4. The similarity between a radical and a character/word is also defined as the cosine similarity between their embeddings. From the results, we can see that almost all the characters and words are disease-related, e.g., "佝 (rickets)", "痨 (tuberculosis)", and "癣疥 (ringworm scabies)", and most of them share the same radical "疒 (illness)". This observation demonstrates the rationality of embedding Chinese words, characters, and radicals into the same vector space, and measuring their similarities directly in that space. Note that this operation might be problematic for English. For example, it could be hard to figure out what kind of similarity there is between the character "i" and the word "ill". But for Chinese, this problem might be alleviated since characters and radicals themselves contain rich semantic information.

4 Conclusion and Future Work

In this paper we propose a new approach to Chinese word embedding, referred to as *multi-granularity embedding* (MGE). MGE improves word embedding by further leveraging both characters and radicals, and hence makes full use of the word-character-radical semantic composition. Experimental results on word similarity computation and analogical reasoning demonstrate the superiority of MGE over state-of-the-art methods. A qualitative analysis further shows that by incorporating radical information MGE can identify finer-grained semantic meanings of words.

As future work, we would like to 1) Investigate more complicate composition manners among radicals, characters, and words, e.g., a hierarchical structure of them. 2) Explore the semantic composition of higher level language units such as phrases, sentences, and even documents.

5 Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by the National Natural Science Foundation of China (grant No. 61402465 and No. 61402466) and the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200).

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and

word embeddings. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1236–1242.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1681–1691.

Hashimoto Kazuma and Tsuruoka Yoshimasa. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015. Component-enhanced chinese character embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 829–834.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700*.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21*, pages 1081–1088.

Jerome L Myers, Arnold Well, and Robert Frederick Lorch. 2010. *Research design and statistical analysis*. Routledge.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 594–598.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang, 2014. *Radical-Enhanced Chinese Character Embedding*, chapter Proceedings of the 21st International Conference on Neural Information Processing, pages 279–286.

Ruifeng Xu, Tao Chen, Yunqing Xia, Qin Lu, Bin Liu, and Xuan Wang. 2015. Word embedding composition for data imbalances in sentiment and emotion classification. *Cognitive Computation*, 7(2):226–240.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1374–1379.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 250–259.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Numerically Grounded Language Models for Semantic Error Correction

Georgios P. Spithourakis and **Isabelle Augenstein** and **Sebastian Riedel**
Department of Computer Science
University College London
{g.spithourakis, i.augenstein, s.riedel}@cs.ucl.ac.uk

Abstract

Semantic error detection and correction is an important task for applications such as fact checking, speech-to-text or grammatical error correction. Current approaches generally focus on relatively shallow semantics and do not account for numeric quantities. Our approach uses language models grounded in numbers within the text. Such groundings are easily achieved for recurrent neural language model architectures, which can be further conditioned on incomplete background knowledge bases. Our evaluation on clinical reports shows that numerical grounding improves perplexity by 33% and F1 for semantic error correction by 5 points when compared to ungrounded approaches. Conditioning on a knowledge base yields further improvements.

1 Introduction

In many real world scenarios it is important to detect and potentially correct semantic errors and inconsistencies in text. For example, when clinicians compose reports, some statements in the text may be inconsistent with measurements taken from the patient (Bowman, 2013). Error rates in clinical data range from 2.3% to 26.9% (Goldberg et al., 2008) and many of them are number-based errors (Arts et al., 2002). Likewise, a blog writer may make statistical claims that contradict facts recorded in databases (Munger, 2008). Numerical concepts constitute 29% of contradictions in Wikipedia and GoogleNews (De Marneffe et al., 2008) and 8.8% of contradictory pairs in entailment datasets (Dagan et al., 2006).

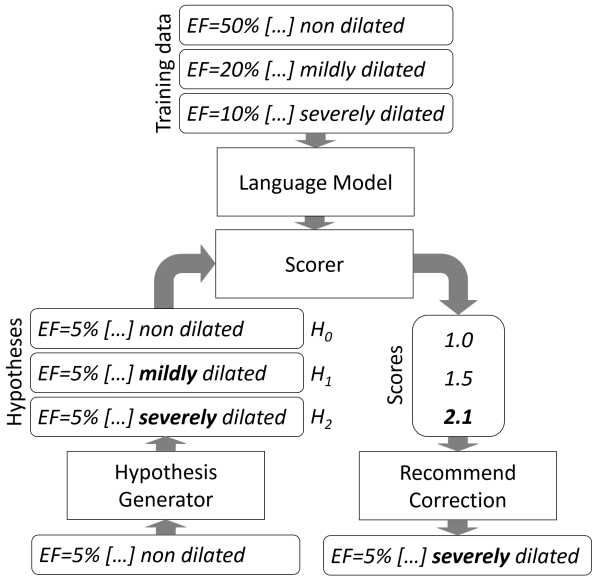


Figure 1: Semantic error correction using language models. "EF" is a clinical term and stands for "ejection fraction".

These inconsistencies may stem from oversight, lack of reporting guidelines or negligence. In fact they may not even be errors at all, but point to interesting outliers or to errors in a reference database. In all cases, it is important to spot and possibly correct such inconsistencies. This task is known as semantic error correction (*SEC*) (Dahlmeier and Ng, 2011).

In this paper, we propose a SEC approach to support clinicians with writing patient reports. A SEC system reads a patient's structured background information from a knowledge base (*KB*) and their clinical report. Then it recommends improvements to the text of the report for semantic consistency. An example of an inconsistency is shown in Figure 1.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 987–992,
Austin, Texas, November 1-5, 2016.

The SEC system has been trained on a dataset of records and learnt that the phrases "non dilated" and "severely dilated" correspond to high and low values for "EF" (abbreviation for "ejection fraction", a clinical measurement), respectively. If the system is then presented with the phrase "non dilated" in the context of a low value, it will detect a semantic inconsistency and correct the text to "severely dilated".

Our contributions are: 1) a straightforward extension to recurrent neural network (RNN) language models for *grounding* them in numbers available in the text; 2) a simple method for modelling text *conditioned on* an incomplete KB by lexicalising it; 3) our evaluation on a semantic error correction task for clinical records shows that our method achieves F1 improvements of 5 and 6 percentage points with grounding and KB conditioning, respectively, over an ungrounded approach (F1 of 49%).

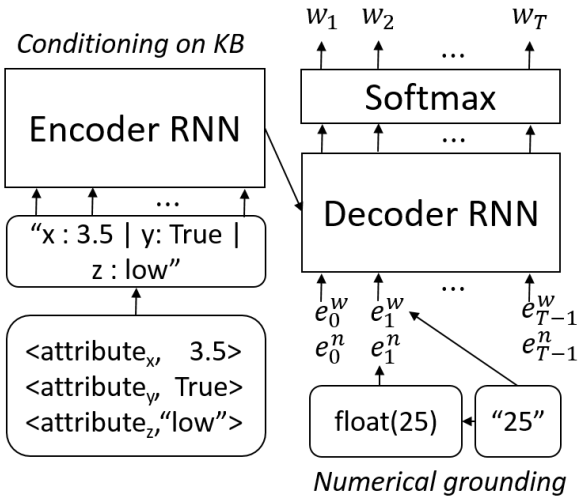


Figure 2: A language model that is numerically grounded and conditioned on a lexicalised KB. Examples of data in rounded rectangles.

2 Methodology

Our approach to semantic error correction (Figure 1) starts with training a language model (LM), which can be grounded in numeric quantities mentioned inline with text (Subsection 2.1) and/or conditioned on a potentially incomplete KB (Subsection 2.2). Given a document for semantic checking, a hypothesis generator proposes corrections, which are then scored using the trained language model (Subsection 2.3). A final decision step involves accepting the best scoring hypothesis.

2.1 Numerically grounded language modelling

Let $\{w_1, ..., w_T\}$ denote a document, where w_t is the one-hot representation of the t-th token and V is the vocabulary size. A neural LM uses a matrix, $E_{in} \in \mathbb{R}^{D \times V}$, to derive word embeddings, $e^w_t = E_{in} w_t$. A hidden state from the previous time step, h_{t-1}, and the current word embedding, e^w_t, are sequentially fed to an RNN's recurrence function to produce the current hidden state, $h_t \in \mathbb{R}^D$. The conditional probability of the next word is estimated as $\mathrm{softmax}(E_{out} h_t)$, where $E_{out} \in \mathbb{R}^{V \times D}$ is an output embeddings matrix.

We propose concatenating a representation, e^n_t, of the numeric value of w_t to the inputs of the RNN's recurrence function at each time step. Through this numeric representation, the model can generalise to out-of-vocabulary numbers. A straightforward representation is defining $e^n_t = \mathrm{float}(w_t)$, where $\mathrm{float}(.)$ is a numeric conversion function that returns a floating point number constructed from the string of its input. If conversion fails, it returns zero.

The proposed mechanism for *numerical grounding* is shown in Figure 2. Now the probability of each next word depends on numbers that have appeared earlier in the text. We treat numbers as a separate modality that happens to share the same medium as natural language (text), but can convey exact measurements of properties of the real world. At training time, the numeric representations mediate to ground the language model in the real world.

2.2 Conditioning on incomplete KBs

The proposed extension can also be used in *conditional language modelling* of documents given a knowledge base. Consider a set of KB tuples accompanying each document and describing its attributes in the form $< attribute, value >$, where attributes are defined by a KB schema. We can *lexicalise* the KB by converting its tuples into textual statements of the form "$attribute$: $value$". An example of how we lexicalise the KB is shown in Figure 2. The generated tokens can then be interpreted for their word embeddings and numeric representations. This

			train	dev	test
	#documents		11,158	1,625	3,220
#tokens/	doc	all	204.9	204.4	202.2
		words	95.7%	95.7%	95.7%
		numeric	4.3%	4.3%	4.3%
#unique	tokens	all	18,916	6,572	9,515
		words	47.8%	58.25%	54.1%
		numeric	52.24%	41.9%	45.81%
OOV	rate	all	5.0%	5.1%	5.2%
		words	3.4%	3.5%	3.5%
		numeric	40.4%	40.8%	41.8%

Table 1: Statistics for clinical dataset. Counts for non-numeric (*words*) and *numeric* tokens reported as percentage of counts for *all* tokens. Out-of-vocabulary (OOV) rates are for vocabulary of 1000 most frequent words in the train data.

approach can incorporate KB tuples flexibly, even when values of some attributes are missing.

2.3 Semantic error correction

A statistical model chooses the most likely correction from a set of possible correction choices. If the model scores a corrected hypothesis higher than the original document, the correction is accepted.

A *hypothesis generator function*, G, takes the original document, H_0, as input and generates a set of candidate corrected documents $G(H_0) = \{H_1, ..., H_M\}$. A simple hypothesis generator uses confusion sets of semantically related words to produce all possible substitutions.

A *scorer model*, s, assigns a score $s(H_i) \in \mathbb{R}$ to a hypothesis H_i. The scorer is based on a likelihood ratio test between the original document (null hypothesis, H_0) and each candidate correction (alternative hypotheses, H_i), i.e. $s(H_i) = \frac{p(H_i)}{p(H_0)}$. The assigned score represents how much more probable a correction is than the original document.

The probability of observing a document, $p(H_i)$, can be estimated using language models, or grounded and conditional variants thereof.

3 Data

Our dataset comprises 16,003 clinical records from the London Chest Hospital (Table 1). Each patient record consists of a text report and accompanying structured KB tuples. The latter describe 20 possible numeric attributes (age, gender, etc.), which are also partly contained in the report. On average, 7.7 tuples are completed per record. Numeric tokens constitute only a small proportion of each sentence (4.3%), but account for a large part of the unique tokens vocabulary (>40%) and suffer from high OOV rates.

description	confusion set
intensifiers (adv):	*non*, *mildly*, *severely*
intensifiers (adj):	*mild*, *moderate*, *severe*
units:	*cm*, *mm*, *ml*, *kg*, *bpm*
viability:	*viable*, *non-viable*
quartiles:	*25*, *50*, *75*, *100*
inequalities:	$<$, $>$

Table 2: Confusion sets.

To evaluate SEC, we generate a "corrupted" dataset of semantic errors from the test part of the "trusted" dataset (Table 1, last column). We manually build confusion sets (Table 2) by searching the development set for words related to numeric quantities and grouping them if they appear in similar contexts. Then, for each document in the trusted test set we generate an erroneous document by sampling a substitution from the confusion sets. Documents with no possible substitution are excluded. The resulting "corrupted" dataset is balanced, containing 2,926 correct and 2,926 incorrect documents.

4 Results and discussion

Our *base LM* is a single-layer long short-term memory network (LSTM, Hochreiter and Schmidhuber (1997) with all latent dimensions (internal matrices, input and output embeddings) set to $D = 50$. We extend this baseline to a *conditional* variant by conditioning on the lexicalised KB (see Section 2.2). We also derive a numerically *grounded* model by concatenating the numerical representation of each token to the inputs of the base LM model (see Section 2.1). Finally, we consider a model that is both grounded and conditional (*g-conditional*).

The vocabulary contains the $V = 1000$ most frequent tokens in the training set. Out-of-vocabulary tokens are substituted with *<num_unk>*, if numeric, and *<unk>*, otherwise. We extract the numerical representations before masking, so that the grounded models can generalise to out-of-vocabulary numbers. Models are trained to minimise token cross-entropy, with 20 epochs of back-

model	tokens	PP	APP
base LM	all	14.96	22.11
	words	13.93	17.94
	numeric	72.38	2289.47
conditional	all	14.52	21.47
	words	13.49	17.38
	numeric	74.48	2355.77
grounded	all	9.91	14.66
	words	9.28	11.96
	numeric	42.67	1349.59
g-conditional	all	**9.39**	**13.88**
	words	**8.80**	**11.33**
	numeric	**39.84**	**1260.28**

Table 3: Language modelling evaluation results on the test set. We report perplexity (PP) and adjusted perplexity (APP). Best results in **bold**.

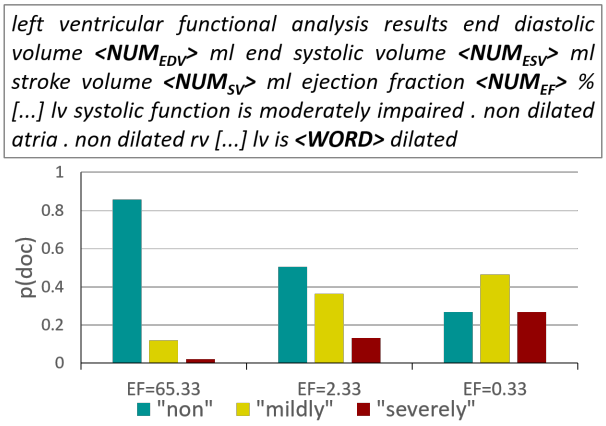


Figure 3: Qualitative example. Template document and document probabilities for <WORD>={'non', 'mildly', 'severely'} and varying numbers. Probabilities are renormalised over the set of possible choices.

propagation and adaptive mini-batch gradient descent (AdaDelta) (Zeiler, 2012).

For SEC, we use an oracle hypothesis generator that has access to the groundtruth confusion sets (Table 2). We estimate the scorer (Section 2.3) using the trained *base*, *conditional*, *grounded* or *g-conditional* LMs. As additional baselines we consider a scorer that assigns *random* scores from a uniform distribution and *always* (*never*) scorers that assign the lowest (highest) score to the original document and uniformly random scores to the corrections.

4.1 Experiment 1: Numerically grounded LM

We report perplexity and adjusted perplexity (Ueberla, 1994) of our LMs on the test set for all tokens and token classes (Table 3). Adjusted perplexity is not sensitive to OOV-rates and thus allows for meaningful comparisons across token classes. Perplexities are high for numeric tokens because they form a large proportion of the vocabulary. The *grounded* and *g-conditional* models achieved a 33.3% and 36.9% improvement in perplexity, respectively, over the *base LM* model. Conditioning without grounding yields only slight improvements, because most of the numerical values from the lexicalised KB are out-of-vocabulary.

The qualitative example in Figure 3 demonstrates how numeric values influence the probability of tokens given their history. We select a document from the development set and substitute its numeric values as we vary EF (the rest are set by solving a known system of equations). The selected exact values were unseen in the training data. We calculate the probabilities for observing the document with different word choices {"non", "mildly", "severely"} under the *grounded* LM and find that "non dilated" is associated with higher EF values. This shows that it has captured semantic dependencies on numbers.

4.2 Experiment 2: Semantic error correction

We evaluate SEC systems on the corrupted dataset (Section 3) for detection and correction.

For detection, we report precision, recall and F1 scores in Table 4. Our *g-conditional* model achieves the best results, a total F1 improvement of 2 points over the *base LM* model and 7 points over the best baseline. The conditional model without grounding performs slightly worse in the F1 metric than the *base LM*. Note that with more hypotheses the *random* baseline behaves more similarly to *always*. Our hypothesis generator generated on average 12 hypotheses per document. The results of *never* are zero as it fails to detect any error.

For correction, we report mean average precision (MAP) in addition to the same metrics as for detection (Table 5). The former measures the position of the ranking of the correct hypothesis. The *always* (*never*) baseline ranks the correct hypothesis at the top (bottom). Again, the *g-conditional* model

model	P	R	F1
random	50.27	90.29	64.58
always	50.00	100.0	66.67
never	0.0	0.0	0.0
base LM	57.51	94.05	71.38
conditional	56.86	94.43	70.98
grounded	58.87	94.70	72.61
g-conditional	**60.48**	**95.25**	**73.98**

Table 4: Error detection results on the test set. We report precision (P), recall (R) and F1. Best results in **bold**.

model	MAP	P	R	F1
random	27.75	5.73	10.29	7.36
always	20.39	6.13	12.26	8.18
never	60.06	0.0	0.0	0.0
base LM	64.37	39.54	64.66	49.07
conditional	62.76	37.46	62.20	46.76
grounded	68.21	44.25	71.19	54.58
g-conditional	**69.14**	**45.36**	**71.43**	**55.48**

Table 5: Error correction results on the test set. We report mean average precision (MAP), precision (P), recall (R) and F1. Best results in **bold**.

yields the best results, achieving an improvement of 6 points in F1 and 5 points in MAP over the *base LM* model and an improvement of 47 points in F1 and 9 points in MAP over the best baseline. The *conditional* model without grounding has the worst performance among the LM-based models.

5 Related Work

Grounded language models represent the relationship between words and the non-linguistic context they refer to. Previous work grounds language on vision (Bruni et al., 2014; Socher et al., 2014; Silberer and Lapata, 2014), audio (Kiela and Clark, 2015), video (Fleischman and Roy, 2008), colour (McMahan and Stone, 2015), and olfactory perception (Kiela et al., 2015). However, no previous approach has explored in-line numbers as a source of grounding.

Our language modelling approach to SEC is inspired by LM approaches to grammatical error detection (GEC) (Ng et al., 2013; Felice et al., 2014). They similarly derive confusion sets of semantically related words, substitute the target words with alternatives and score them with an LM. Existing semantic error correction approaches aim at correcting word error choices (Dahlmeier and Ng, 2011), collocation errors (Kochmar, 2016), and semantic anomalies in adjective-noun combinations (Vecchi et al., 2011). So far, SEC approaches focus on short distance semantic agreement, whereas our approach can detect errors which require to resolve long-range dependencies. Work on GEC and SEC shows that language models are useful for error correction, however they neither ground in numeric quantities nor incorporate background KBs.

6 Conclusion

In this paper, we proposed a simple technique to model language in relation to numbers it refers to, as well as conditionally on incomplete knowledge bases. We found that the proposed techniques lead to performance improvements in the tasks of language modelling, and semantic error detection and correction. Numerically grounded models make it possible to capture semantic dependencies of content words on numbers.

In future work, we will plan to apply numerically grounded models to other tasks, such as numeric error correction. We will explore alternative ways for deriving the numeric representations, such as accounting for verbal descriptions of numbers. For SEC, a trainable hypothesis generator can potentially improve the coverage of the system.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. We also thank Steffen Petersen for providing the dataset and advising us on the clinical aspects of this work. This research was supported by the Farr Institute of Health Informatics Research, an Allen Distinguished Investigator award and Elsevier.

References

Danielle GT Arts, Nicolette F De Keizer, and Gert-Jan Scheffer. 2002. Defining and improving data quality in medical registries: a literature review, case study, and generic framework. *Journal of the American Medical Informatics Association*, 9(6):600–611.

Sue Bowman. 2013. Impact of Electronic Health Record Systems on Information Integrity: Quality and Safety Implications. *Perspectives in Health Information Management*, page 1.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47).

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting Semantic Collocation Errors with L1-induced Paraphrases. In *Proceedings of EMNLP*, pages 107–117.

Marie-Catherine De Marneffe, Anna N Rafferty, and Christopher D Manning. 2008. Finding Contradictions in Text. In *ACL*, volume 8, pages 1039–1047.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *CoNLL Shared Task*, pages 15–24.

Michael Fleischman and Deb Roy. 2008. Grounded Language Modeling for Automatic Speech Recognition of Sports Video. In *Proceedings of ACL*, pages 121–129.

Saveli Goldberg, Andrzej Niemierko, and Alexander Turchin. 2008. Analysis of data errors in clinical research databases. In *AMIA*. Citeseer.

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

Douwe Kiela and Stephen Clark. 2015. Multi- and Cross-Modal Semantics Beyond Vision: Grounding in Auditory Perception. In *Proceedings of EMNLP*, pages 2461–2470.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015. Grounding Semantics in Olfactory Perception. In *Proceedings of ACL*, pages 231–236.

Ekaterina Kochmar. 2016. *Error Detection in Content Word Combinations*. Ph.D. thesis, University of Cambridge, Computer Laboratory.

Brian McMahan and Matthew Stone. 2015. A bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Michael C Munger. 2008. Blogging and political information: truth or truthiness? *Public Choice*, 134(1-2):125–138.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In Hwee Tou Ng, Joel Tetreault, Siew Mei Wu, Yuanbin Wu, and Christian Hadiwinoto, editors, *Proceedings of the CoNLL: Shared Task*, pages 1–12.

Carina Silberer and Mirella Lapata. 2014. Learning Grounded Meaning Representations with Autoencoders. In *Proceedings of ACL*, pages 721–732.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *TACL*, 2:207–218.

Joerg Ueberla. 1994. Analysing a simple language model· some general conclusions for language models for speech recognition. *Computer Speech & Language*, 8(2):153–176.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (Linear) Maps of the Impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.

Towards Semi-Automatic Generation of Proposition Banks for Low-Resource Languages

Alan Akbik
IBM Research
Almaden Research Center
San Jose, CA 95120

Vishwajeet Kumar
IIT Bombay
CS and Engineeering
Mumbai, India

Yunyao Li
IBM Research
Almaden Research Center
San Jose, CA 95120

{akbika,yunyaoli}@us.ibm.com vishwajeetkumar86@gmail.com

Abstract

Annotation projection based on parallel corpora has shown great promise in inexpensively creating Proposition Banks for languages for which high-quality parallel corpora and syntactic parsers are available. In this paper, we present an experimental study where we apply this approach to three languages that lack such resources: *Tamil*, *Bengali* and *Malayalam*. We find an average quality difference of 6 to 20 absolute F-measure points vis-a-vis high-resource languages, which indicates that annotation projection alone is insufficient in low-resource scenarios. Based on these results, we explore the possibility of using annotation projection as a starting point for inexpensive data curation involving both experts and non-experts. We give an outline of what such a process may look like and present an initial study to discuss its potential and challenges.

1 Introduction

Creating syntactically and semantically annotated NLP resources for low-resource languages is known to be immensely costly. For instance, the Proposition Bank (Palmer et al., 2005) was created by annotating predicate-argument structures in the Penn Treebank (Marcus et al., 1993) with shallow semantic labels: *frame* labels for verbal predicates and *role* labels for arguments. Similarly, the SALSA (Burchardt et al., 2006) resource added FrameNet-style annotations to the TIGER Treebank (Brants et al., 2002), the Chinese Propbank (Xue, 2008) is built on the Chinese Treebank (Xue et al., 2005), and so forth. Since each such layer of annotation typically requires years of manual work, the accumulated costs can be prohibitive for low-resource languages.

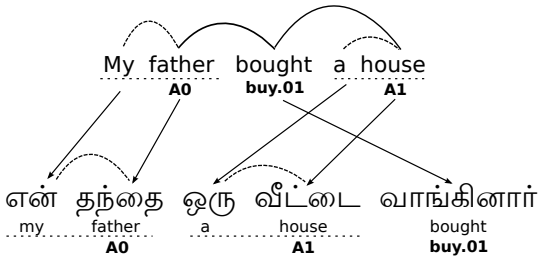


Figure 1: Annotation projection on a pair of very simple sentences. English Propbank frame (**buy.01**) and role (**A0**, **A1**) labels are projected onto aligned Tamil words. Furthermore, the typed dependencies between the words "my father" and "a house" (dotted lines) are projected onto their Tamil equivalents.

Recent work on **annotation projection** offers a way to inexpensively label a target language corpus with linguistic annotation (Padó and Lapata, 2009). This only requires a word-aligned parallel corpus of labeled English sentences and their translations in the target language. English labels are then automatically projected onto the aligned target language words. Refer to Figure 1 for an example.

Low-resource languages. However, previous work that investigated Propbank annotation projection has focused only on languages for which treebanks - and therefore syntactic parsers - already exist. Since syntactic information is typically used to increase projection accuracy (Padó and Lapata, 2009; Akbik et al., 2015), we must expect this approach to work less well for low-resource languages. In addition, low-resource languages have fewer sources of high-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 993–998,
Austin, Texas, November 1-5, 2016.

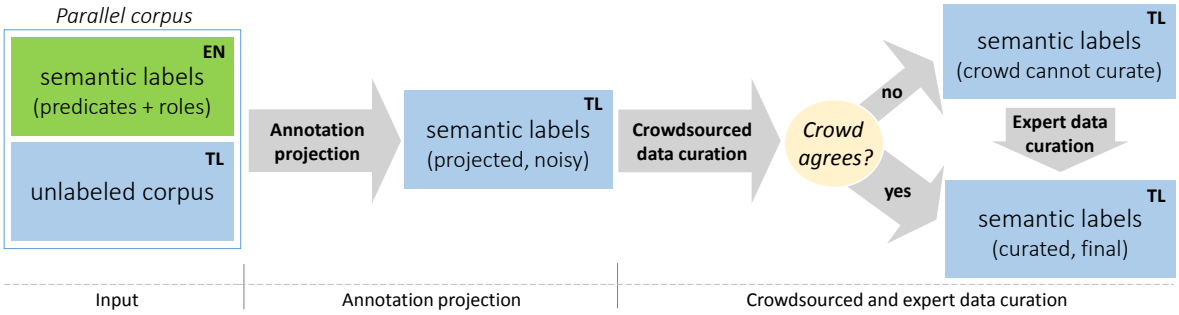


Figure 2: Proposed process of using annotation projection in a parallel corpus from English (EN) to a target language (TL) as basis for crowdsourced data curation. Experts are only involved in cases where the crowd cannot agree on a label.

quality parallel data available, further complicating annotation projection.

Contributions. In this paper, we present a study in which we apply annotation projection to three low-resource languages in order to quantify the difference in precision and recall vis-a-vis high-resource languages. Our study finds overall F1-measure of generated Proposition Banks to be significantly below state-of-the-art results, leading us to conclude that annotation projection may at best be a *starting point* for the generation of semantic resources for low-resource languages. To explore this idea, we outline a potential semi-automatic process in which we use crowdsourced data curation and limited expert involvement to confirm and correct automatically projected labels. Based on this initial study, we discuss the potential and challenges of the proposed approach.

2 Annotation Projection

Annotation projection takes as input a word-aligned parallel corpus of sentences in a source language (usually English) and their target language translations. A syntactic parser and a semantic role labeler produce labels for the English sentences, which are then projected onto aligned target language words. The underlying theory is that parallel sentences share a degree of syntactic and, in particular, semantic similarity, making such projection possible (Padó and Lapata, 2009).

State-of-the-art. Previous work analyzed errors in annotation projection and found that they are often caused by non-literal translations (Akbik et al., 2015). For this reason, previous work defined lexical and syntactic constraints to increase projection quality. These include verb filters to allow only verbs to be labeled as frames (Van der Plas et al., 2011), heuristics to ensure that only heads of syntactic constituents are labeled as arguments (Padó and Lapata, 2009) and the use of verb translation dictionaries (Akbik et al., 2015) to constrain frame mappings.

Adaptation to low-resource languages. Low-resource languages, however, lack syntactic parsers to identify target language predicate-argument structures. This requires us to make the following modifications to the approach:

Target language predicates We define lexical constraints using verb translation dictionaries. This ensures that only target language verbs that are aligned to literal source language translations are labeled as frames.

Target language arguments To identify arguments, we project not only the role label of source language arguments heads, but the entire argument dependency structure. This is illustrated in Figure 1: Two dependency arcs are projected from English onto Tamil, giving evidence that arguments **A0** and **A1** in the Tamil sentence each consist of two words.

This step produces a target language corpus with semantically annotated predicate-argument structure.

3 Outline of a Data Curation Process

As confirmed in the experiments section of this paper, the quality of the Proposition Banks generated using annotation projection is significantly lower for low-resource languages. We therefore propose to use this approach only as a starting point for an inexpensive curation process as illustrated in Figure 2:

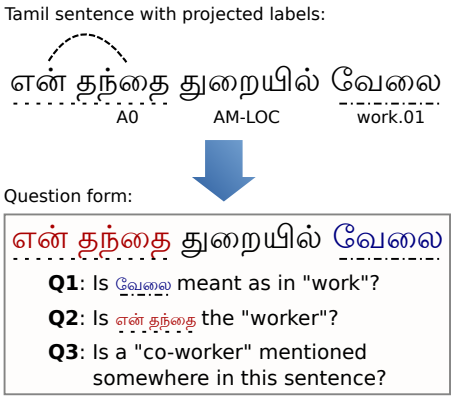


Figure 3: Example of how data curation questions may be formulated for the labels projected onto Tamil in Figure 1.

Step 1: Crowdsourced data curation. Previous work has experimented with different approaches in crowdsourcing to generate frame-semantic annotations over text (Hong and Baker, 2011), including selection tasks (selecting one answer from a list of options) (Fossati et al., 2013) and marking tasks (marking text passages that evoke a certain semantic role) (Feizabadi and Padó, 2014). While these studies only report moderate results on annotator correctness and agreement, our goal is different from these works in that we only wish to curate projected labels, not generate SRL annotations from scratch. A related project in extending FrameNet with paraphrases (Pavlick et al., 2015) has shown that the crowd can effectively curate wrong paraphrases by answering a series of confirm-or-reject questions.

For our initial study, we generate human readable question-answer pairs (He et al., 2015) using the label descriptions of the English Propbank (see Figure 3). We generate two types of questions:

Label confirmation questions are confirm-or-reject questions on whether projected labels are correct (e.g. **Q1** and **Q2** in Figure 3). Workers further qualify their answers to indicate whether a sequence of words marked as an argument is incomplete.

Missing label questions are marking tasks which ask whether any core role labels of a frame are missing. For example, the BUY.01 frame has 5 core roles (labeled **A0** to **A4**), one of which is the ”price” (**A3**). Since no ”price” is labeled in the Tamil sentence in Figure 3, question **Q3** asks users to add this label if a ”price” is mentioned.

DATA SET	Bengali	Malayalam	Tamil
OPENSUBTITLES2016	75K	224K	21K
SPOKENTUTORIALS	31K	17K	32K
Total # sentences	106K	241K	53K

Table 1: Parallel data sets and number of parallel sentences used for each language.

Our goal is to effectively distribute a large part of the curation workload. In cases where the crowd unanimously agrees, we remove labels judged to be incorrect and add labels judged to be missing.

Step 2: Expert data curation. We also expect a percentage of questions for which non-experts will give conflicting answers[1]. As Figure 2 shows, such cases will be passed to experts for further curation. However, for the purpose of scalability, we aim to keep expert involvement to a minimum.

4 Experimental Study

We report our initial investigations over the following questions: (1) What are the differences in annotation projection quality between low- and high-resource languages?; and (2) Can non-experts be leveraged to at least partially curate projected labels?

4.1 Experimental Setup

Languages. We evaluate three low-resource languages, namely *Bengali*, an Indo-Aryan language, as well as *Tamil* and *Malayalam*, two South Dravidian languages. Between them, they are estimated to have more than 300 million first language speakers, yet there are few NLP resources available.

Data sets. We use two parallel corpora (see Table 1): OPENSUBTITLES2016 (Tiedemann, 2012), a corpus automatically generated from movie subtitles, and SPOKENTUTORIALS, a corpus of technical-domain tutorial translations.

Evaluation. For the purpose of comparison to previous work on high-resource languages, we replicate

[1]Common problems for non-experts that we observe in our initial experiments involve ambiguities caused by implicit or causal role-predicate relationships, as well as figurative usage and hypotheticals.

		PRED.	ARGUMENT			
LANG.	Match	P	P	R	F1	%Agree
Bengali	partial	1.0	0.84	0.68	0.75	0.67
PROJECTED	exact	1.0	0.83	0.68	**0.75**	
Bengali	partial	1.0	0.88	0.69	0.78	
CURATED	exact	1.0	0.87	0.69	**0.77**	
Malayalam	partial	0.99	0.87	0.65	0.75	**0.65**
PROJECTED	exact	0.99	0.79	0.63	**0.7**	
Malayalam	partial	0.99	0.92	0.69	0.78	
CURATED	exact	0.99	0.84	0.67	**0.74**	
Tamil	partial	0.77	0.49	0.59	0.53	0.75
PROJECTED	exact	0.77	0.45	0.58	**0.5**	
Tamil	partial	0.77	0.62	0.67	0.64	
CURATED	exact	0.77	0.58	0.65	**0.61**	
Chinese	partial	0.97	0.93	0.83	0.88	**0.92**
(Akbik et al., 2015)	exact	0.97	0.83	0.81	**0.82**	
German	partial	0.96	0.95	0.73	0.83	**0.92**
(Akbik et al., 2015)	exact	0.96	0.91	0.73	**0.81**	
Hindi	partial	0.91	0.93	0.66	0.77	0.81
(Akbik et al., 2015)	exact	0.91	0.58	0.54	**0.56**	

Table 2: Estimated precision and recall for Tamil, Bengali and Malayalam before and after non-expert curation. We list state-of-the-art results for German and Hindi for comparison.

earlier evaluation practice and English preprocessing steps (Akbik et al., 2015). After projection, we randomly select 100 sentences for each target language and pass them to a curation step by 2 non-experts. We then measure the inter-annotator agreement and the quality of the generated Proposition Banks in terms of predicate precision[2] and argument F1-score before and after crowdsourced curation[3].

4.2 Results

The evaluation results are listed in Table 2. For comparison, we include evaluation results reported for three high-resource languages: German and Chinese, representing average high-resource results, as well as Hindi, a below-average outlier. We make the following observations:

Lower annotation projection quality. We find that the F1-scores of Bengali, Malayalam and Tamil are 6, 11 and 31 pp below that of an average high-resource language (as exemplified by German in Table 2). Bengali and Malayalam, however, do surpass Hindi, for which only a relatively poor dependency parser was used. This suggests that syntactic annotation projection may be a better method for identifying predicate-argument structures in languages that lack fully developed dependency parsers.

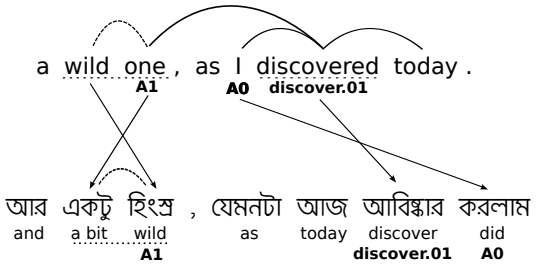


Figure 4: Example of a projection error. The verb *discover* in Bengali is a light verb construction. In addition, the pronoun *I* is not explicitly mentioned in the Bengali target sentence. This causes the pronoun *I* to be mistakenly aligned to the auxiliary of the light verb, causing it to be falsely labeled as **A0**.

Impact of parallel data. We note a significant impact of the size and quality of available parallel data on overall quality. For instance, the lowest-scoring language in our experiments, Tamil, use the smallest amount parallel data (see Table 1), most of which was from the SPOKENTUTORIALS corpus. This data is specific to the technical domain and seems less suited for annotation projection than the more general OPENSUBTITLES2016 corpus.

A qualitative inspection of projection errors points to a large portion of errors stemming from translation shifts. For instance, refer to Figure 4 for an English-Bengali example of the impact of even slight differences in translation: The English verb *discover* is expressed in Bengali as a light verb, while the pronoun *I* is dropped in the Bengali sentence (it is still implicitly evoked through the verb being in first person form). This causes the word alignment to align the English *I* to the Bengali auxiliary, onto which the role label **A0** is then incorrectly projected.

5 Discussion

In all three languages, we note improvements through curation. Argument F1-score improves to 77% (↑2 pp) for Bengali, to 74% (↑4 pp) for Malay-

[2]Since we do not ask missing label questions for predicates, we cannot estimate predicate recall.

[3]Following (Akbik et al., 2015), in the *exact* evaluation scheme, labels marked as correct and complete count as true positives. In *partial*, incomplete correct labels also count as true positives.

alam, and to 61% (↑11 pp) for Tamil on exact matches. Especially Tamil improves drastically, albeit from a much lower initial score than the other languages. This supports our general observation that crowd workers are good at spotting obvious errors, while they often disagree about more subtle differences in semantics. These results indicate that a curation process can at least be partially crowdsourced. An interesting question for further investigation is to what degree this is possible. As Table 2 shows, non-expert agreement in our initial study was far below reported expert agreement, with 25% to 35% of all questions problematic for non-experts.

A particular focus of our future work is therefore to quantify to which extent crowd-feedback can be valuable and how far the involvement of experts can be minimized for cost-effective resource generation. However, a Proposition Bank generated through this process would be peculiar in several ways:

Crowd semantics. First, generated Proposition Banks would be created in a drastically different way than current approaches that rely on experts to create and annotate frames. Effectively, the non-expert crowd would, to a large degree, shape the selection and annotation of English frame and role annotation for new target languages. An important question therefore is to what degree an auto-generated Propbank would differ from an expertly created one. In a related line of work (Akbik et al., 2016), we have conducted a preliminary comparison of an auto-generated Proposition Bank for Chinese and the manually created Chinese Proposition Bank (Xue and Palmer, 2005). Encouragingly, we find a significant overlap between both versions. Future work will further explore the usefulness of auto-generated Propbanks to train a semantic role labeler (Akbik and Li, 2016) and their usefulness for downstream applications in low-resource languages.

Partial syntactic annotation. Second, while curation of semantically labeled predicate-argument structure can be formulated as human intelligence tasks, this will not in all likelihood be possible for full parse trees. These Propbanks would therefore lack a treebank-style syntactic layer of annotation. Would an existing Propbank facilitate the future task of creating treebanks for low-resource languages? In other words, could the traditional order of first creating treebanks and then Propbanks be reversed?

PROPBANK	#SENTENCES	#LABELS	#FRAMES
Bengali	5,757	17,899	88
Malayalam	10,579	26,831	95
Tamil	3,486	11,765	68

Table 3: Number of labeled sentences, semantic labels and distinct frames of each auto-generated Propbank (*before* non-expert curation).

6 Conclusion and Outlook

We applied annotation projection to low-resource languages and found a significant drop in quality vis-a-vis high-resource languages. We then proposed and outlined a curation process for semi-automatically generating Proposition Banks and noted encouraging results in an initial study. To encourage discussion within the research community, we make our generated Proposition Banks for Bengali, Malayalam and Tamil (see Table 3 for an overview) publicly available[4].

References

[Akbik and Li2016] Alan Akbik and Yunyao Li. 2016. Polyglot: Multilingual semantic role labeling with unified labels. In *ACL 2016, 54th Annual Meeting of the Association for Computational Linguistics: Demostration Session*, page to appear.

[Akbik et al.2015] Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *ACL 2015, 53rd Annual Meeting of the Association for Computational Linguistics Beijing, China*, pages 397–407.

[Akbik et al.2016] Alan Akbik, Xinyu Guan, and Yunyao Li. 2016. Multilingual aliasing for auto-generating proposition banks. In *COLING 2016, the 26th International Conference on Computational Linguistics (to appear)*.

[Brants et al.2002] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, volume 168.

[Burchardt et al.2006] Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The salsa corpus: a german

[4]Datasets will be made available at this page: `http://researcher.watson.ibm.com/researcher/view_group_subpage.php?id=7454`

corpus resource for lexical semantics. In *Proceedings of LREC 2006, Fifth International Conference on Language Resources and Evaluation*, volume 6.

[Feizabadi and Padó2014] Parvin Sadat Feizabadi and Sebastian Padó. 2014. Crowdsourcing annotation of non-local semantic roles. In *EACL*, pages 226–230.

[Fossati et al.2013] Marco Fossati, Claudio Giuliano, and Sara Tonelli. 2013. Outsourcing framenet to the crowd. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 742–747, Sofia, Bulgaria, August. Association for Computational Linguistics.

[He et al.2015] Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal, September. Association for Computational Linguistics.

[Hong and Baker2011] Jisup Hong and Collin F Baker. 2011. How good is the crowd at real wsd? In *Proceedings of the 5th linguistic annotation workshop*, pages 30–37. Association for Computational Linguistics.

[Marcus et al.1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

[Padó and Lapata2009] Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

[Palmer et al.2005] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

[Pavlick et al.2015] Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *ACL (2)*, pages 408–413. The Association for Computer Linguistics.

[Tiedemann2012] Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of LREC 2012, Eighth International Conference on Language Resources and Evaluation*, pages 2214–2218.

[Van der Plas et al.2011] Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 299–304. Association for Computational Linguistics.

[Xue and Palmer2005] Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

[Xue et al.2005] Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.

[Xue2008] Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics*, 34(2):225–255.

A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis

Sebastian Ruder[1,2], **Parsa Ghaffari**[2], **and John G. Breslin**[1]

[1]Insight Centre for Data Analytics
National University of Ireland, Galway
{sebastian.ruder,john.breslin}@insight-centre.org
[2]Aylien Ltd.
Dublin, Ireland
{sebastian,parsa}@aylien.com

Abstract

Opinion mining from customer reviews has become pervasive in recent years. Sentences in reviews, however, are usually classified independently, even though they form part of a review's argumentative structure. Intuitively, sentences in a review build and elaborate upon each other; knowledge of the review structure and sentential context should thus inform the classification of each sentence. We demonstrate this hypothesis for the task of aspect-based sentiment analysis by modeling the interdependencies of sentences in a review with a hierarchical bidirectional LSTM. We show that the hierarchical model outperforms two non-hierarchical baselines, obtains results competitive with the state-of-the-art, and outperforms the state-of-the-art on five multilingual, multi-domain datasets without any hand-engineered features or external resources.

1 Introduction

Sentiment analysis (Pang and Lee, 2008) is used to gauge public opinion towards products, to analyze customer satisfaction, and to detect trends. With the proliferation of customer reviews, more fine-grained aspect-based sentiment analysis (ABSA) has gained in popularity, as it allows aspects of a product or service to be examined in more detail.

Reviews – just with any coherent text – have an underlying structure. A visualization of the discourse structure according to Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) for the example review in Figure 1 reveals that sentences and clauses are connected via different rhetorical relations, such as *Elaboration* and *Background*.

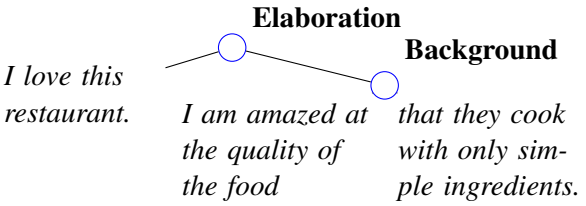


Figure 1: RST structure of an example review.

Intuitively, knowledge about the relations and the sentiment of surrounding sentences should inform the sentiment of the current sentence. If a reviewer of a restaurant has shown a positive sentiment towards the quality of the food, it is likely that his opinion will not change drastically over the course of the review. Additionally, overwhelmingly positive or negative sentences in the review help to disambiguate sentences whose sentiment is equivocal.

Neural network-based architectures that have recently become popular for sentiment analysis and ABSA, such as convolutional neural networks (Severyn and Moschitti, 2015), LSTMs (Vo and Zhang, 2015), and recursive neural networks (Nguyen and Shirai, 2015), however, are only able to consider intra-sentence relations such as *Background* in Figure 1 and fail to capture inter-sentence relations, e.g. *Elaboration* that rely on discourse structure and provide valuable clues for sentiment prediction.

We introduce a hierarchical bidirectional long short-term memory (H-LSTM) that is able to leverage both intra- and inter-sentence relations. The sole dependence on sentences and their structure

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 999–1005,
Austin, Texas, November 1-5, 2016.

within a review renders our model fully language-independent. We show that the hierarchical model outperforms strong sentence-level baselines for aspect-based sentiment analysis, while achieving results competitive with the state-of-the-art and outperforming it on several datasets without relying on any hand-engineered features or sentiment lexica.

2 Related Work

Aspect-based sentiment analysis. Past approaches use classifiers with expensive hand-crafted features based on n-grams, parts-of-speech, negation words, and sentiment lexica (Pontiki et al., 2014; Pontiki et al., 2015). The model by Zhang and Lan (2015) is the only approach we are aware of that considers more than one sentence. However, it is less expressive than ours, as it only extracts features from the preceding and subsequent sentence without any notion of structure. Neural network-based approaches include an LSTM that determines sentiment towards a target word based on its position (Tang et al., 2015) as well as a recursive neural network that requires parse trees (Nguyen and Shirai, 2015). In contrast, our model requires no feature engineering, no positional information, and no parser outputs, which are often unavailable for low-resource languages. We are also the first – to our knowledge – to frame sentiment analysis as a sequence tagging task.

Hierarchical models. Hierarchical models have been used predominantly for representation learning and generation of paragraphs and documents: Li et al. (2015) use a hierarchical LSTM-based autoencoder to reconstruct reviews and paragraphs of Wikipedia articles. Serban et al. (2016) use a hierarchical recurrent encoder-decoder with latent variables for dialogue generation. Denil et al. (2014) use a hierarchical ConvNet to extract salient sentences from reviews, while Kotzias et al. (2015) use the same architecture to learn sentence-level labels from review-level labels using a novel cost function. The model of Lee and Dernoncourt (2016) is perhaps the most similar to ours. While they also use a sentence-level LSTM, their class-level feed-forward neural network is only able to consider a limited number of preceding texts, while our review-level bidirectional LSTM is (theoretically) able to consider an unlimited number of preceding *and* successive sentences.

3 Model

In the following, we will introduce the different components of our hierarchical bidirectional LSTM architecture displayed in Figure 2.

3.1 Sentence and Aspect Representation

Each review consists of sentences, which are padded to length l by inserting padding tokens. Each review in turn is padded to length h by inserting sentences containing only padding tokens. We represent each sentence as a concatentation of its word embeddings $x_{1:l}$ where $x_t \in \mathbb{R}^k$ is the k-dimensional vector of the t-th word in the sentence.

Every sentence is associated with an aspect. Aspects consist of an entity and an attribute, e.g. `FOOD#QUALITY`. Similarly to the entity representation of Socher et al. (2013), we represent every aspect a as the average of its entity and attribute embeddings $\frac{1}{2}(x_e + x_a)$ where $x_e, x_a \in \mathbb{R}^m$ are the m-dimensional entity and attribute embeddings respectively[1].

3.2 LSTM

We use a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), which adds input, output, and forget gates to a recurrent cell, which allow it to model long-range dependencies that are essential for capturing sentiment.

For the t-th word in a sentence, the LSTM takes as input the word embedding x_t, the previous output h_{t-1} and cell state c_{t-1} and computes the next output h_t and cell state c_t. Both h and c are initialized with zeros.

3.3 Bidirectional LSTM

Both on the review and on the sentence level, sentiment is dependent not only on preceding but also successive words and sentences. A Bidirectional LSTM (Bi-LSTM) (Graves et al., 2013) allows us to look ahead by employing a forward LSTM, which processes the sequence in chronological order, and a backward LSTM, which processes the sequence in reverse order. The output h_t at a given time step is then the concatenation of the corresponding states of the forward and backward LSTM.

[1] Averaging embeddings produced slightly better results than using a separate embedding for every aspect.

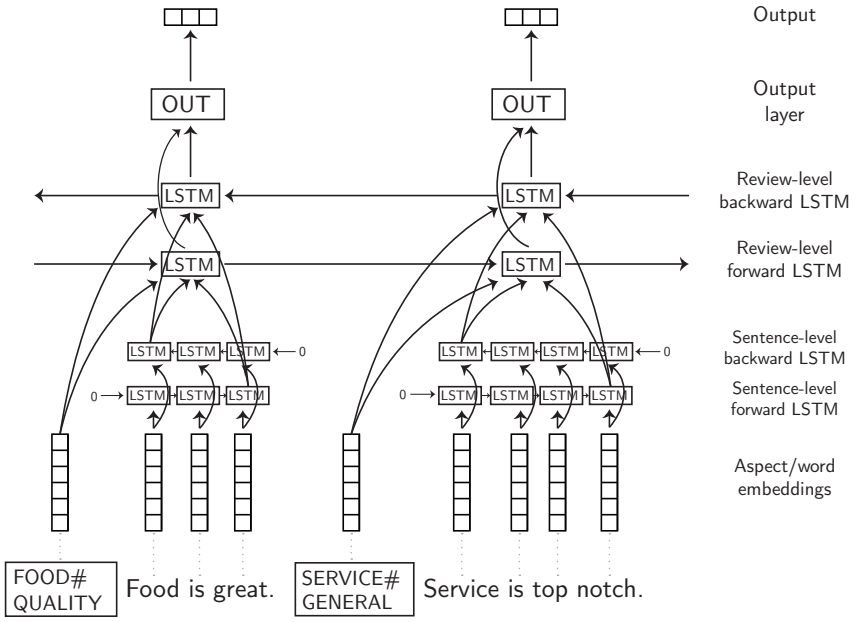


Figure 2: The hierarchical bidirectional LSTM (H-LSTM) for aspect-based sentiment analysis. Word embeddings are fed into a sentence-level bidirectional LSTM. Final states of forward and backward LSTM are concatenated together with the aspect embedding and fed into a bidirectional review-level LSTM. At every time step, the output of the forward and backward LSTM is concatenated and fed into a final layer, which outputs a probability distribution over sentiments.

3.4 Hierarchical Bidirectional LSTM

Stacking a Bi-LSTM on the review level on top of sentence-level Bi-LSTMs yields the hierarchical bidirectional LSTM (H-LSTM) in Figure 2.

The sentence-level forward and backward LSTMs receive the sentence starting with the first and last word embedding x_1 and x_l respectively. The final output h_l of both LSTMs is then concatenated with the aspect vector a[2] and fed as input into the review-level forward and backward LSTMs. The outputs of both LSTMs are concatenated and fed into a final softmax layer, which outputs a probability distribution over sentiments[3] for each sentence.

4 Experiments

4.1 Datasets

For our experiments, we consider datasets in five domains (restaurants, hotels, laptops, phones, cameras) and eight languages (English, Spanish, French, Russian, Dutch, Turkish, Arabic, Chinese) from the recent SemEval-2016 Aspect-based Sentiment Analysis task (Pontiki et al., 2016), using the provided train/test splits. In total, there are 11 domain-language datasets containing 300-400 reviews with 1250-6000 sentences[4]. Each sentence is annotated with none, one, or multiple domain-specific aspects and a sentiment value for each aspect.

4.2 Training Details

Our LSTMs have one layer and an output size of 200 dimensions. We use 300-dimensional word embeddings. We use pre-trained GloVe (Pennington et al., 2014) embeddings for English, while we train embeddings on `frWaC`[5] for French and on the Leipzig Corpora Collection[6] for all other languages.[7] Entity

[2]We experimented with other interactions, e.g. rescaling the word embeddings by their aspect similarity, an attention-like mechanism, as well as summing and multiplication, but found that simple concatenation produced the best results.

[3]The sentiment classes are *positive*, *negative*, and *neutral*.

[4]Exact dataset statistics can be seen in (Pontiki et al., 2016).

[5]`http://wacky.sslmit.unibo.it/doku.php?id=corpora`

[6]`http://corpora2.informatik.uni-leipzig.de/download.html`

[7]Using 64-dimensional Polyglot embeddings (Al-Rfou et al., 2013) yielded generally worse performance.

Language	Domain	Best	XRCE	IIT-TUDA	CNN	LSTM	H-LSTM	HP-LSTM
English	Restaurants	**88.1**	**88.1**	86.7	82.1	81.4	83.0	85.3
Spanish	Restaurants	**83.6**	-	**83.6**	79.6	75.7	79.5	81.8
French	Restaurants	**78.8**	**78.8**	72.2	73.2	69.8	73.6	75.4
Russian	Restaurants	77.9	-	73.6	75.1	73.9	**78.1**	77.4
Dutch	Restaurants	77.8	-	77.0	75.0	73.6	82.2	**84.8**
Turkish	Restaurants	**84.3**	-	**84.3**	74.2	73.6	76.7	79.2
Arabic	Hotels	82.7	-	81.7	82.7	80.5	82.8	**82.9**
English	Laptops	**82.8**	-	**82.8**	78.4	76.0	77.4	80.1
Dutch	Phones	83.3	-	82.6	83.3	81.8	81.3	**83.6**
Chinese	Cameras	**80.5**	-	-	78.2	77.6	78.6	78.8
Chinese	Phones	73.3	-	-	72.4	70.3	**74.1**	73.3

Table 1: Results of our system with randomly initialized word embeddings (H-LSTM) and with pre-trained embeddings (HP-LSTM) for ABSA for each language and domain in comparison to the best system for each pair (Best), the best two single systems (XRCE, IIT-TUDA), a sentence-level CNN (CNN), and our sentence-level LSTM (LSTM).

and attribute embeddings of aspects have 15 dimensions and are initialized randomly. We use dropout of 0.5 after the embedding layer and after LSTM cells, a gradient clipping norm of 5, and no l_2 regularization.

We unroll the aspects of every sentence in the review, e.g. a sentence with two aspects occurs twice in succession, once with each aspect. We remove sentences with no aspect[8] and ignore predictions for all sentences that have been added as padding to a review so as not to force our model to learn meaningless predictions, as is commonly done in sequence-to-sequence learning (Sutskever et al., 2014). We segment Chinese data before tokenization.

We train our model to minimize the cross-entropy loss, using stochastic gradient descent, the Adam update rule (Kingma and Ba, 2015), mini-batches of size 10, and early stopping with a patience of 10.

4.3 Comparison models

We compare our model using random (H-LSTM) and pre-trained word embeddings (HP-LSTM) against the best model of the SemEval-2016 Aspect-based Sentiment Analysis task (Pontiki et al., 2016) for each domain-language pair (Best) as well as against the two best single models of the competition: IIT-TUDA (Kumar et al., 2016), which uses large sentiment lexicons for every language, and XRCE (Brun et al., 2016), which uses a parser augmented with hand-crafted, domain-specific rules. In order to ascertain that the hierarchical nature of our model is the deciding factor, we additionally compare against the sentence-level convolutional neural network of Ruder et al. (2016) (CNN) and against a sentence-level Bi-LSTM (LSTM), which is identical to the first layer of our model.[9]

5 Results and Discussion

We present our results in Table 1. Our hierarchical model achieves results superior to the sentence-level CNN and the sentence-level Bi-LSTM baselines for almost all domain-language pairs by taking the structure of the review into account. We highlight examples where this improves predictions in Table 2.

In addition, our model shows results competitive with the best single models of the competition, while requiring no expensive hand-crafted features or external resources, thereby demonstrating its language and domain independence. Overall, our model compares favorably to the state-of-the-art, particularly for low-resource languages, where few hand-engineered features are available. It outperforms the state-of-the-art on four and five datasets using randomly initialized and pre-trained embeddings respectively.

[8] Labeling them with a NONE aspect and predicting *neutral* slightly decreased performance.

[9] To ensure that the additional parameters do not account for the difference, we increase the number of layers and dimensions of LSTM, which does not impact the results.

Id	Sentence	LSTM	H-LSTM
1.1	No Comparison	*negative*	*positive*
1.2	It has great sushi and even better service.	*positive*	*positive*
2.1	Green Tea creme brulee is a must!	*positive*	*positive*
2.2	Don't leave the restaurant without it.	*negative*	*positive*

Table 2: Example sentences where knowledge of other sentences in the review (not necessarily neighbors) helps to disambiguate the sentiment of the sentence in question. For the aspect in 1.1, the sentence-level LSTM predicts *negative*, while the context of the service and food quality in 1.2 allows the H-LSTM to predict *positive*. Similarly, for the aspect in 2.2, knowledge of the quality of the green tea crème brulée helps the H-LSTM to predict the correct sentiment.

5.1 Pre-trained embeddings

In line with past research (Collobert et al., 2011), we observe significant gains when initializing our word vectors with pre-trained embeddings across almost all languages. Pre-trained embeddings improve our model's performance for all languages except Russian, Arabic, and Chinese and help it achieve state-of-the-art in the Dutch phones domain. We release our pre-trained multilingual embeddings so that they may facilitate future research in multilingual sentiment analysis and text classification[10].

5.2 Leveraging additional information

As annotation is expensive in many real-world applications, learning from only few examples is important. Our model was designed with this goal in mind and is able to extract additional information inherent in the training data. By leveraging the structure of the review, our model is able to inform and improve its sentiment predictions as evidenced in Table 2.

The large performance differential to the state-of-the-art for the Turkish dataset where only 1104 sentences are available for training and the performance gaps for high-resource languages such as English, Spanish, and French, however, indicate the limits of an approach such as ours that only uses data available at training time.

While using pre-trained word embeddings is an effective way to mitigate this deficit, for high-resource languages, solely leveraging unsupervised language information is not enough to perform on-par with approaches that make use of large external resources (Kumar et al., 2016) and meticulously hand-crafted features (Brun et al., 2016).

Sentiment lexicons are a popular way to inject additional information into models for sentiment analysis. We experimented with using sentiment lexicons by Kumar et al. (2016) but were not able to significantly improve upon our results with pre-trained embeddings[11]. In light of the diversity of domains in the context of aspect-based sentiment analysis and many other applications, domain-specific lexicons (Hamilton et al., 2016) are often preferred. Finding better ways to incorporate such domain-specific resources into models as well as methods to inject other forms of domain information, e.g. by constraining them with rules (Hu et al., 2016) is thus an important research avenue, which we leave for future work.

6 Conclusion

In this paper, we have presented a hierarchical model of reviews for aspect-based sentiment analysis. We demonstrate that by allowing the model to take into account the structure of the review and the sentential context for its predictions, it is able to outperform models that only rely on sentence information and achieves performance competitive with models that leverage large external resources and hand-engineered features. Our model achieves state-of-the-art results on 5 out of 11 datasets for aspect-based sentiment analysis.

Acknowledgments

We thank the anonymous reviewers, Nicolas Pécheux, and Hugo Larochelle for their constructive feedback. This publication has emanated from research conducted with the financial support of the Irish Research Council (IRC) under Grant Number EBPPG/2014/30 and with Aylien Ltd. as Enterprise Partner as well as from research supported by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

[10] https://s3.amazonaws.com/aylien-main/data/multilingual-embeddings/index.html

[11] We tried bucketing and embedding of sentiment scores as well as filtering and pooling as in (Vo and Zhang, 2015)

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed Word Representations for Multilingual NLP. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192.

Caroline Brun, Julien Perez, and Claude Roux. 2016. XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modelling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 282–286.

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

Misha Denil, Alban Demiraj, and Nando de Freitas. 2014. Extraction of Salient Sentences from Labelled Documents. *arXiv preprint arXiv:1412.6815*, pages 1–9.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (3):6645–6649.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1–18.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–13.

Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. 2015. From Group to Individual Labels using Deep Features. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597—-606.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. *Proceedings of NAACL-HLT 2016*.

Jiwei Li, Minh-Thang Luong, and Daniel Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1106–1115.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (September):2509–2514.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

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

Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeny Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 Task 5: Aspect-Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, San Diego, California. Association for Computational Linguistics.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. INSIGHT-1 at SemEval-2016 Task 5: Deep Learning for Multilingual Aspect-based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. *Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 1–14.

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 464–469.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Proceedings of the Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 1–10.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, page 9.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-Dependent Sentiment Classification with Long Short Term Memory. *arXiv preprint arXiv:1512.01100*.

Duy-tin Vo and Yue Zhang. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1347–1353.

Zhihua Zhang and Man Lan. 2015. ECNU: Extracting Effective Features from Multiple Sequential Sentences for Target-dependent Sentiment Analysis in Reviews. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 736–741.

Are Word Embedding-based Features Useful for Sarcasm Detection?

Aditya Joshi[1,2,3] **Vaibhav Tripathi**[1] **Kevin Patel**[1]
Pushpak Bhattacharyya[1] **Mark Carman**[2]
[1]Indian Institute of Technology Bombay, India
[2]Monash University, Australia
[3]IITB-Monash Research Academy, India
{adityaj,kevin.patel,pb}@cse.iitb.ac.in, mark.carman@monash.edu

Abstract

This paper makes a simple increment to state-of-the-art in sarcasm detection research. Existing approaches are unable to capture subtle forms of context incongruity which lies at the heart of sarcasm. We explore if prior work can be enhanced using semantic similarity/discordance between word embeddings. We augment word embedding-based features to four feature sets reported in the past. We also experiment with four types of word embeddings. We observe an improvement in sarcasm detection, irrespective of the word embedding used or the original feature set to which our features are augmented. For example, this augmentation results in an improvement in F-score of around 4% for three out of these four feature sets, and a minor degradation in case of the fourth, when Word2Vec embeddings are used. Finally, a comparison of the four embeddings shows that Word2Vec and dependency weight-based features outperform LSA and GloVe, in terms of their benefit to sarcasm detection.

1 Introduction

Sarcasm is a form of verbal irony that is intended to express contempt or ridicule. Linguistic studies show that the notion of context incongruity is at the heart of sarcasm (Ivanko and Pexman, 2003). A popular trend in automatic sarcasm detection is semi-supervised extraction of patterns that capture the underlying context incongruity (Davidov et al., 2010; Joshi et al., 2015; Riloff et al., 2013). However, techniques to extract these patterns rely on sentiment-bearing words and may not capture nuanced forms of sarcasm. Consider the sentence '*With a sense of humor like that, you could make a living as a garbage man anywhere in the country.*[1]' The speaker makes a subtle, contemptuous remark about the sense of humor of the listener. However, absence of sentiment words makes the sarcasm in this sentence difficult to capture as features for a classifier.

In this paper, we explore use of word embeddings to capture context incongruity in the absence of sentiment words. The intuition is that **word vector-based similarity/discordance is indicative of semantic similarity which in turn is a handle for context incongruity**. In the case of the 'sense of humor' example above, the words 'sense of humor' and 'garbage man' are semantically dissimilar and their presence together in the sentence provides a clue to sarcasm. Hence, our set of features based on word embeddings aim to capture such semantic similarity/discordance. Since such semantic similarity is but one of the components of context incongruity **and** since existing feature sets rely on sentiment-based features to capture context incongruity, it is imperative that the two be combined for sarcasm detection. Thus, our paper deals with the question:

Can word embedding-based features when augmented to features reported in prior work improve the performance of sarcasm detection?

To the best of our knowledge, this is the first attempt that uses word embedding-based features to detect sarcasm. In this respect, the paper makes a simple increment to state-of-the-art but opens up a new direction in sarcasm detection research. We establish our hypothesis in case of four past works and four types of word embeddings, to show that the benefit of using word embedding-based features holds across multiple feature sets and word embeddings.

2 Motivation

In our literature survey of sarcasm detection (Joshi et al., 2016), we observe that a popular trend is semi-supervised extraction of patterns with implicit sentiment. One such work is by Riloff et al. (2013) who give a bootstrapping algorithm that discovers a set of positive verbs and

[1]All examples in this paper are actual instances from our dataset.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1006–1011,
Austin, Texas, November 1-5, 2016.

negative/undesirable situations. However, this simplification (of representing sarcasm merely as positive verbs followed by negative situation) may not capture difficult forms of context incongruity. Consider the sarcastic sentence 'A woman needs a man like a fish needs bicycle'[2]. The sarcasm in this sentence is understood from the fact that a fish does not need bicycle - and hence, the sentence ridicules the target 'a man'. However, this sentence does not contain any sentiment-bearing word. Existing sarcasm detection systems relying on sentiment incongruity (as in the case of our past work reported as Joshi et al. (2015)) may not work well in such cases of sarcasm.

To address this, we use semantic similarity as a handle to context incongruity. To do so, we use word vector similarity scores. Consider similarity scores (as given by Word2Vec) between two pairs of words in the sentence above:

similarity(man,woman) = 0.766
similarity(fish,bicycle) = 0.131

Words in one part of this sentence ('man' and 'woman') are lot more similar than words in another part of the sentence ('fish' and 'bicycle'). This semantic discordance can be a clue to presence of context incongruity. Hence, we propose features based on similarity scores between word embeddings of words in a sentence. In general, we wish to capture the most similar and most dissimilar word pairs in the sentence, and use their scores as features for sarcasm detection.

3 Background: Features from prior work

We augment our word embedding-based features to the following four feature sets that have been reported:

1. **Liebrecht et al. (2013)**: They consider unigrams, bigrams and trigrams as features.
2. **González-Ibánez et al. (2011a)**: They propose two sets of features: unigrams and dictionary-based. The latter are words from a lexical resource called LIWC. We use words from LIWC that have been annotated as emotion and psychological process words, as described in the original paper.
3. **Buschmeier et al. (2014)**: In addition to unigrams, they propose features such as: (a) Hyperbole (captured by three positive or negative words in a row), (b) Quotation marks and ellipsis, (c) Positive/Negative Sentiment words followed by an exclamation mark or question mark, (d) Positive/Negative Sentiment Scores followed by ellipsis (represented by a '...'), (e) Punctuation, (f) Interjections, and (g) Laughter expressions.
4. **Joshi et al. (2015)**: In addition to unigrams, they use features based on implicit and explicit incongruity. Implicit incongruity features are patterns with implicit sentiment as extracted in a pre-processing step. Explicit incongruity features consist of number of sentiment flips, length of positive and negative subsequences and lexical polarity.

4 Word Embedding-based Features

In this section, we now describe our word embedding-based features. We reiterate that these features will be augmented to features from prior works (described in Section 3).

As stated in Section 2, our word embedding-based features are based on similarity scores between word embeddings. The similarity score is the cosine similarity between vectors of two words. To illustrate our features, we use our example '*A woman needs a man like a fish needs a bicycle*'. The scores for all pairs of words in this sentence are given in Table 1.

	man	**woman**	**fish**	**needs**	**bicycle**
man	-	**0.766**	**0.151**	**0.078**	**0.229**
woman	0.766	-	0.084	0.060	0.229
fish	0.151	0.084	-	0.022	0.130
needs	0.078	0.060	0.022	-	0.060
bicycle	0.229	0.229	0.130	0.060	-

Table 1: Similarity scores for all pairs of content words in 'A woman needs a man like a fish needs bicycle'

Using these similarity scores, we compute two sets of features:

1. **Unweighted similarity features (S)**: We first compute similarity scores for all pairs of words (except stop words). We then return four feature values per sentence.[3]:
   - Maximum score of most *similar* word pair
   - Minimum score of most *similar* word pair
   - Maximum score of most *dissimilar* word pair
   - Minimum score of most *dissimilar* word pair

   For example, in case of the first feature, we consider the most similar word to every word in the sentence, and the corresponding similarity scores. These most similar word scores for each word are indicated in bold in Table 1. Thus, the first feature in case of our example would have the value 0.766 derived from the man-woman pair and the second feature would take the value 0.078 due to the needs-man pair. The other features are computed in a similar manner.

[2]This quote is attributed to Irina Dunn, an Australian writer (https://en.wikipedia.org/wiki/Irina_Dunn

[3]These feature values consider all words in the sentence, *i.e.*, the 'maximum' is computed over all words

2. **Distance-weighted similarity features (WS)**: Like in the previous case, we first compute similarity scores for all pairs of words (excluding stop-words). For all similarity scores, we divide them by square of distance between the two words. Thus, the similarity between terms that are close in the sentence is weighted higher than terms which are distant from one another. Thus, for all possible word pairs, we compute four features:

   - Maximum distance-weighted score of most *similar* word pair
   - Minimum distance-weighted score of most *similar* word pair
   - Maximum distance-weighted score of most *dissimilar* word pair
   - Minimum distance-weighted score of most *dissimilar* word pair

   These are computed similar to unweighted similarity features.

5 Experiment Setup

We create a dataset consisting of quotes on GoodReads [4]. GoodReads describes itself as '*the world's largest site for readers and book recommendations.*' The website also allows users to post quotes from books. These quotes are snippets from books labeled by the user with tags of their choice. We download quotes with the tag 'sarcastic' as sarcastic quotes, and the ones with 'philosophy' as non-sarcastic quotes. Our labels are based on these tags given by users. We ensure that no quote has both these tags. This results in a dataset of 3629 quotes out of which 759 are labeled as sarcastic. This skew is similar to skews observed in datasets on which sarcasm detection experiments have been reported in the past (Riloff et al., 2013).

We report five-fold cross-validation results on the above dataset. We use SVM_{perf} by Joachims (2006) with c as 20, w as 3, and loss function as F-score optimization. This allows SVM to be learned while optimizing the F-score.

As described above, we compare features given in prior work alongside the augmented versions. This means that for each of the four papers, we experiment with four configurations:

1. Features given in paper X
2. Features given in paper X + unweighted similarity features (S)
3. Features given in paper X + weighted similarity features (WS)
4. Features given in paper X + S+WS (*i.e.*, weighted and unweighted similarity features)

Features	P	R	F
Baseline			
Unigrams	67.2	78.8	72.53
S	64.6	75.2	69.49
WS	67.6	51.2	58.26
Both	67	52.8	59.05

Table 2: Performance of unigrams versus our similarity-based features using embeddings from Word2Vec

We experiment with four types of word embeddings:

1. **LSA**: This approach was reported in Landauer and Dumais (1997). We use pre-trained word embeddings based on LSA[5]. The vocabulary size is 100,000.
2. **GloVe**: We use pre-trained vectors avaiable from the GloVe project[6]. The vocabulary size in this case is 2,195,904.
3. **Dependency Weights**: We use pre-trained vectors[7] weighted using dependency distance, as given in Levy and Goldberg (2014). The vocabulary size is 174,015.
4. **Word2Vec**: use pre-trained Google word vectors. These were trained using Word2Vec tool [8] on the Google News corpus. The vocabulary size for Word2Vec is 3,000,000. To interact with these pre-trained vectors, as well as compute various features, we use gensim library (Řehůřek and Sojka, 2010).

To interact with the first three pre-trained vectors, we use scikit library (Pedregosa et al., 2011).

6 Results

Table 2 shows performance of sarcasm detection when our word embedding-based features are used on their own *i.e*, not as augmented features. The embedding in this case is Word2Vec. The four rows show baseline sets of features: unigrams, unweighted similarity using word embeddings (S), weighted similarity using word embeddings (WS) and both (*i.e.*, unweighted plus weighted similarities using word embeddings). Using only unigrams as features gives a F-score of 72.53%, while only unweighted and weighted features gives F-score of 69.49% and 58.26% respectively. This **validates our intuition**

[4] www.goodreads.com

[5] http://www.lingexp.uni-tuebingen.de/z2/LSAspaces/

[6] http://nlp.stanford.edu/projects/glove/

[7] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

[8] https://code.google.com/archive/p/Word2Vec/

	LSA			GloVe			Dependency Weights			Word2Vec		
	P	R	F	P	R	F	P	R	F	P	R	F
L	73	79	75.8	73	79	75.8	73	79	75.8	73	79	75.8
+S	81.8	78.2	**79.95**	81.8	79.2	**80.47**	81.8	78.8	80.27	80.4	80	**80.2**
+WS	76.2	79.8	77.9	76.2	79.6	77.86	81.4	80.8	81.09	80.8	78.6	79.68
+S+WS	77.6	79.8	78.68	74	79.4	76.60	82	80.4	**81.19**	81.6	78.2	79.86
G	84.8	73.8	78.91	84.8	73.8	78.91	84.8	73.8	**78.91**	84.8	73.8	**78.91**
+S	84.2	74.4	**79**	84	72.6	77.8	84.4	72	77.7	84	72.8	78
+WS	84.4	73.6	78.63	84	75.2	**79.35**	84.4	72.6	78.05	83.8	70.2	76.4
+S+WS	84.2	73.6	78.54	84	74	78.68	84.2	72.2	77.73	84	72.8	78
B	81.6	72.2	76.61	81.6	72.2	76.61	81.6	72.2	76.61	81.6	72.2	76.61
+S	78.2	75.6	**76.87**	80.4	76.2	**78.24**	81.2	74.6	**77.76**	81.4	72.6	76.74
+WS	75.8	77.2	76.49	76.6	77	76.79	76.2	76.4	76.29	81.6	73.4	77.28
+S+WS	74.8	77.4	76.07	76.2	78.2	77.18	75.6	78.8	77.16	81	75.4	**78.09**
J	85.2	74.4	79.43	85.2	74.4	79.43	85.2	74.4	79.43	85.2	74.4	79.43
+S	84.8	73.8	78.91	85.6	74.8	79.83	85.4	74.4	79.52	85.4	74.6	**79.63**
+WS	85.6	75.2	**80.06**	85.4	72.6	78.48	85.4	73.4	78.94	85.6	73.4	79.03
+S+WS	84.8	73.6	78.8	85.8	75.4	**80.26**	85.6	74.4	**79.6**	85.2	73.2	78.74

Table 3: Performance obtained on augmenting word embedding features to features from four prior works, for four word embeddings; L: Liebrecht et al. (2013), G: González-Ibánez et al. (2011a), B: Buschmeier et al. (2014) , J: Joshi et al. (2015)

that word embedding-based features alone are not sufficient, and should be augmented with other features.

Following this, we show performance using features presented in four prior works: Buschmeier et al. (2014), Liebrecht et al. (2013), Joshi et al. (2015) and González-Ibánez et al. (2011a), and compare them with augmented versions in Table 3.

Table 3 shows results for four kinds of word embeddings. All entries in the tables are **higher than the simple unigrams baseline**, *i.e.*, F-score for each of the four is higher than unigrams - highlighting that these are better features for sarcasm detection than simple unigrams. Values in bold indicate the best F-score for a given prior work-embedding type combination. In case of Liebrecht et al. (2013) for Word2Vec, the overall improvement in F-score is 4%. Precision increases by 8% while recall remains nearly unchanged. For features given in González-Ibánez et al. (2011a), there is a negligible degradation of 0.91% when word embedding-based features based on Word2Vec are used. For Buschmeier et al. (2014) for Word2Vec, we observe an improvement in F-score from 76.61% to 78.09%. Precision remains nearly unchanged while recall increases. In case of Joshi et al. (2015) and Word2Vec, we observe a slight improvement of 0.20% when unweighted (S) features are used. This shows that word embedding-based features are useful, across four past works for Word2Vec.

Table 3 also shows that the improvement holds across the four word embedding types as well. The maximum improvement is observed in case of Liebrecht et al. (2013). It is around 4% in case of LSA, 5% in case of GloVe, 6% in case of Dependency weight-based and 4% in case of Word2Vec. These improvements are not directly comparable because the four embeddings have different vocabularies (since they are trained on different datasets) and vocabulary sizes, their results cannot be directly compared.

Therefore, we take an intersection of the vocabulary (*i.e.*, the subset of words present in all four embeddings) and repeat all our experiments using these intersection files. The vocabulary size of these intersection files (for all four embeddings) is 60,252. Table 4 shows the average increase in F-score when a given word embedding and a word embedding-based feature is used, with the intersection file as described above. These gain values are lower than in the previous case. This is because these are the values in case of the intersection versions - which are subsets of the complete embeddings. Each gain value is averaged over the four prior works. Thus, when unweighted similarity (+S) based features computed using LSA are augmented to features from prior work, an average increment of 0.835% is obtained over the four prior works. The values allow us to compare the benefit of using these four kinds of embeddings. In case of unweighted similarity-based features, dependency-based weights give the maximum gain (0.978%). In case of weighted similarity-based features and '+S+WS', Word2Vec gives the maximum gain (1.411%). Table 5 averages these values over the

	Word2Vec	LSA	GloVe	Dep. Wt.
+S	0.835	0.86	0.918	**0.978**
+WS	**1.411**	0.255	0.192	1.372
+S+WS	**1.182**	0.24	0.845	0.795

Table 4: Average gain in F-Scores obtained by using intersection of the four word embeddings, for three word embedding feature-types, augmented to four prior works; Dep. Wt. indicates vectors learned from dependency-based weights

Word Embedding	Average F-score Gain
LSA	0.452
Glove	0.651
Dependency	1.048
Word2Vec	1.143

Table 5: Average gain in F-scores for the four types of word embeddings; These values are computed for a subset of these embeddings consisting of words common to all four

three types of word embedding-based features. Using Dependency-based and Word2Vec embeddings results in a higher improvement in F-score (1.048% and 1.143% respectively) as compared to others.

7 Error Analysis

Some categories of errors made by our system are:

1. **Embedding issues due to incorrect senses**: Because words may have multiple senses, some embeddings lead to error, as in '*Great. Relationship advice from one of America's most wanted.*'.
2. **Contextual sarcasm**: Consider the sarcastic quote '*Oh, and I suppose the apple ate the cheese*'. The similarity score between 'apple' and 'cheese' is 0.4119. This comes up as the maximum similar pair. The most dissimilar pair is 'suppose' and 'apple' with similarity score of 0.1414. The sarcasm in this sentence can be understood only in context of the complete conversation that it is a part of.
3. **Metaphors in non-sarcastic text**: Figurative language may compare concepts that are not directly related but still have low similarity. Consider the non-sarcastic quote '*Oh my love, I like to vanish in you like a ripple vanishes in an ocean - slowly, silently and endlessly*'. Our system incorrectly predicts this as sarcastic.

8 Related Work

Early sarcasm detection research focused on speech (Tepperman et al., 2006) and lexical features (Kreuz and Caucci, 2007). Several other features have been proposed (Kreuz and Caucci, 2007; Joshi et al., 2015; Khattri et al., 2015; Liebrecht et al., 2013; González-Ibánez et al., 2011a; Rakov and Rosenberg, 2013; Wallace, 2015; Wallace et al., 2014; Veale and Hao, 2010; González-Ibánez et al., 2011b; Reyes et al., 2012). Of particular relevance to our work are papers that aim to first extract patterns relevant to sarcasm detection. Davidov et al. (2010) use a semi-supervised approach that extracts sentiment-bearing patterns for sarcasm detection. Joshi et al. (2015) extract phrases corresponding to implicit incongruity i.e. the situation where sentiment is expressed without use of sentiment words. Riloff et al. (2013) describe a bootstrapping algorithm that iteratively discovers a set of positive verbs and negative situation phrases, which are later used in a sarcasm detection algorithm. Tsur et al. (2010) also perform semi-supervised extraction of patterns for sarcasm detection. The only prior work which uses word embeddings for a related task of sarcasm detection is by Ghosh et al. (2015). They model sarcasm detection as a word sense disambiguation task, and use embeddings to identify whether a word is used in the sarcastic or non-sarcastic sense. Two sense vectors for every word are created: one for literal sense and one for sarcastic sense. The final sense is determined based on the similarity of these sense vectors with the sentence vector.

9 Conclusion

This paper shows the benefit of features based on word embedding for sarcasm detection. We experiment with four past works in sarcasm detection, where we augment our word embedding-based features to their sets of features. Our features use the similarity score values returned by word embeddings, and are of two categories: similarity-based (where we consider maximum/minimum similarity score of most similar/dissimilar word pair respectively), and weighted similarity-based (where we weight the maximum/minimum similarity scores of most similar/dissimilar word pairs with the linear distance between the two words in the sentence). We experiment with four kinds of word embeddings: LSA, GloVe, Dependency-based and Word2Vec. In case of Word2Vec, for three of these past feature sets to which our features were augmented, we observe an improvement in F-score of at most 5%. Similar improvements are observed in case of other word embeddings. A comparison of the four embeddings shows that Word2Vec and dependency weight-based features outperform LSA and GloVe.

This work opens up avenues for use of word embeddings for sarcasm classification. Our word embedding-based features may work better if the similarity scores are computed for a subset of words in the sentence, or using weighting based on syntactic distance instead of linear distance as in the case of our weighted similarity-based features.

References

Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan. 2015. Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *EMNLP*.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011a. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011b. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Stacey L Ivanko and Penny M Pexman. 2003. Context incongruity and irony processing. *Discourse Processes*, 35(3):241–279.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 757–762.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2015. Your sentiment precedes you: Using an authors historical tweets to predict sarcasm. In *6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, page 25.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4. Association for Computational Linguistics.

Thomas K Landauer and Susan T. Dumais. 1997. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *PSYCHOLOGICAL REVIEW*, 104(2):211–240.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308.

CC Liebrecht, FA Kunneman, and APJ van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Rachel Rakov and Andrew Rosenberg. 2013. " sure, i did the right thing": a system for sarcasm detection in speech. In *INTERSPEECH*, pages 842–846.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, pages 704–714.

Joseph Tepperman, David R Traum, and Shrikanth Narayanan. 2006. " yeah right": sarcasm recognition for spoken dialogue systems. In *INTERSPEECH*. Citeseer.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.

Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons. In *ECAI*, volume 215, pages 765–770.

Byron C Wallace, Laura Kertz Do Kook Choe, and Eugene Charniak. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 512–516.

Byron C Wallace. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities,entities and sentiment. In *ACL*.

Weakly Supervised Tweet Stance Classification by Relational Bootstrapping

Javid Ebrahimi and **Dejing Dou** and **Daniel Lowd**
Department of Computer and Information Science, University of Oregon
Eugene, Oregon 97403, USA
{javid,dou,lowd}@cs.uoregon.edu

Abstract

Supervised stance classification, in such domains as Congressional debates and online forums, has been a topic of interest in the past decade. Approaches have evolved from text classification to structured output prediction, including collective classification and sequence labeling. In this work, we investigate collective classification of stances on Twitter, using hinge-loss Markov random fields (HL-MRFs). Given the graph of all posts, users, and their relationships, we constrain the predicted post labels and latent user labels to correspond with the network structure. We focus on a weakly supervised setting, in which only a small set of hashtags or phrases is labeled. Using our relational approach, we are able to go beyond the stance-indicative patterns and harvest more stance-indicative tweets, which can also be used to train any linear text classifier when the network structure is not available or is costly.

1 Introduction

Stance classification is the task of determining from text whether the author of the text is in favor of, against, or neutral towards a target of interest. This is an interesting task to study on social networks due to the abundance of personalized and opinionated language. Studying stance classification can be beneficial in identifying electoral issues and understanding how public stance is shaped (Mohammad et al., 2015).

Twitter provides a wealth of information: public tweets by individuals, their profile information, whom they follow, and more. Exploiting all these pieces of information, in addition to the text, could help build better NLP systems. Examples of this approach include user preference modeling (Li et al., 2014), stance classification (Rajadesingan and Liu, 2014), and geolocation identification (Jurgens, 2013; Rahimi et al., 2015). For stance classification, knowing the author's past posting behavior, or her friends' stances on issues, could improve the stance classifier. These are inherently structured problems, and they demand structured solutions, such as Statistical Relational Learning (SRL) (Getoor, 2007). In this paper, we use hinge-loss Markov random fields (HL-MRFs) (Bach et al., 2015), a recent development in the SRL community.

SemEval 2016 Task 6 organizers (Mohammad et al., 2016) released a dataset with Donald Trump as the target, without stance annotation. The goal of the task was to evaluate stance classification systems, which used minimal labeling on phrases. This scenario is becoming more and more relevant due to the vast amount of data and ever-changing nature of the language on social media. This is critical in applications in which a timely detection is highly desired, such as violence detection (Cano Basave et al., 2013) and disaster situations.

Our work is the first to use SRL for stance classification on Twitter. We formulate the weakly supervised stance classification problem as a bi-type collective classification problem: We start from a small set of stance-indicative patterns and label the tweets as positive and negative, accordingly. Then, our relational learner uses these noisy-labeled tweets, as well as the network structure, to classify the stance

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1012–1017,
Austin, Texas, November 1-5, 2016.

of other tweets and authors. Our goal will be to constrain pairs of similar tweets, pairs of tweets and their authors, and pairs of neighboring users to have similar labels. We do this through hinge-loss feature functions that encode our background knowledge about the domain: (1) A person is pro/against Trump if she writes a tweet with such stance; (2) Friends in a social network often agree on their stance toward Trump; (3) similar tweets express similar stances.

2 Related Work

Stance classification is related to sentiment classification with a major difference that the target of interest may not be explicitly mentioned in the text and it may not be the target of opinion in the text (Mohammad et al., 2016). Previous work has focused on Congressional debates (Thomas et al., 2006; Yessenalina et al., 2010), company-internal discussions (Agrawal et al., 2003), and debates in online forums (Anand et al., 2011; Somasundaran and Wiebe, 2010). Stance classification has newly been posed as structured output prediction. For example, citation structure (Burfoot et al., 2011) or rebuttal links (Walker et al., 2012) are used as extra information to model agreements or disagreements in debate posts and to infer their labels. Arguments and counter-arguments occur in sequences; Hasan and Ng (2014) used this observation and posed stance classification in debate forums as a sequence labeling task, and used a global inference method to classify the posts.

Sridhar et al. (2015) use HL-MRFs to collectively classify stance in online debate forums. We address a weakly supervised problem, which makes our approach different as we do not rely on local text classifiers. Rajadesingan et al. (2014) propose a retweet-based label propagation method which starts from a set of known opinionated users and labels the tweets posted by the people who were in the retweet network.

3 Stance Classification on Twitter

3.1 Markov Random Fields

Markov random fields (MRFs) are widely used in machine learning and statistics. Discriminative Markov random fields such as conditional random fields (Lafferty et al., 2001) are defined by a joint distribution over random variables $Y_1, ..., Y_m$ conditioned on $X_1, ..., X_n$ that is specified by a vector of d real-valued potential functions $\phi_l(\boldsymbol{y}, \boldsymbol{x})$ for $l = 1, ..., d$, and a parameter (weight) vector $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$\mathrm{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta}, \boldsymbol{x})} \exp\Big(\langle \boldsymbol{\theta}, \phi(\boldsymbol{y}, \boldsymbol{x}) \rangle\Big)$$

where $\langle \boldsymbol{\theta}, \phi(\boldsymbol{y}, \boldsymbol{x}) \rangle$ denotes the dot product of the parameters and the potential functions, and $Z(\boldsymbol{\theta}, \boldsymbol{x})$ is the partition function.

3.2 HL-MRFs for Tweet Stance Classification

Finding the *maximum a posteriori* (MAP) state is a difficult discrete optimization problem and, in general, is NP-hard. One particular class of MRFs that allows for convex inference is hinge-loss Markov random fields (HL-MRFs) (Bach et al., 2015). In this graphical model, each potential function is a hinge-loss function, and instead of discrete variables, MAP inference is performed over relaxed continuous variables with domain $[0, 1]^n$. These hinge-loss functions, multiplied by the corresponding model parameters (weights), act as penalizers for soft linear constraints in the graphical model.

Consider $\boldsymbol{t}_i$, $\boldsymbol{u}_j$ as the random variables denoting the ith tweet and the jth user. The potential function, $\phi(\boldsymbol{t}_i, \boldsymbol{u}_j)$, relating a user and her tweet is as follows,

$$\max(0, t_{ik} - u_{jk}) \tag{1}$$

where t_{ik} and u_{jk} denote the respective assertions that $\boldsymbol{t}_i$ has label k, and $\boldsymbol{u}_j$ has label k . The function captures the distance between the label for a user and her tweet. In other words, this function measures the penalty for dissimilar labels for a user and her tweet.

For users who are "friends" (i.e., who "follow" each other on Twitter), we add this potential function,

$$\max(0, u_{ik} - u_{jk}) \tag{2}$$

and for the tweet-tweet relations,

$$s_{ij} \max(0, t_{ik} - t_{jk}) \tag{3}$$

where s_{ij} measures the similarity between two tweets. This scalar helps penalize violations in proportion to the similarity between the tweets. For the similarity measure, we simply used the cosine similarity between the n-gram (1-4-gram) representation of the tweets and set 0.7 as the cutoff threshold.

Finally, two hard linear constraints are added, to ensure that $\boldsymbol{t}_i$, and $\boldsymbol{u}_j$ are each assigned a single label, or in other words, are fractionally assigned labels with weights that sum to one.

$$\sum_k t_{ik} = 1 \ , \ \sum_k u_{ik} = 1 \tag{4}$$

Weight learning is performed by an improved structured voted perceptron (Lowd and Domingos, 2007), at every iteration of which we estimate the labels of the users by hard EM. This formulation can work in weakly supervised settings, because the constraints simply dictate similar/neighboring nodes to have similar labels.

In the language of Probabilistic Soft Logic (PSL) (Bach et al., 2015), the constraints can be defined by the following rules:

PSL Rules:

tweet-label(T, L) $\wedge$ tweet-user(T, U) $\Rightarrow$ user-label(U, L)
user-label(U_1, L) $\wedge$ friend(U_1, U_2) $\Rightarrow$ user-label(U_2, L)
tweet-label(T_1, L) $\wedge$ similar(T_1, T_2) $\Rightarrow$ tweet-label(T_2, L)
PredicateConstraint.Functional , on : user-label
PredicateConstraint.Functional , on : tweet-label

Our post-similarity constraint implementation is different from the original PSL implementation due to the *multiplicative* similarity scalar[1].

This work is a first step toward relational stance classification on Twitter. Incorporating other relational features, such as mention networks and retweet networks can potentially improve our results. Similarly, employing textual entailment techniques for tweet similarity will most probably improve our results.

4 Experiments and Results

4.1 Data

SemEval-2016 Task 6.b (Mohammad et al., 2016) provided 78,000+ tweets associated with "Donald Trump". The protocol of the task only allowed minimal manual labeling, i.e. "tweets or sentences that are manually labeled for stance" were not allowed, but "manually labeling a handful of hashtags" was permitted. Additionally, using Twitter's API, we collected each user's follower list and their profile information. This often requires a few queries per user. We only considered the tweets which contain no URL, are not retweets, and have at most three hashtags and three mentions.

[1]The original implementation would result in the function, max(0, $t_{ik} + s_{ij} - t_{jk} - 1$), which is less intuitive than ours.

Algorithm Relational Bootstrapping

Input:
Unlabeled pairs of tweets and authors ($\mathbf{t}_i$, $\mathbf{u}_i$).
Friendship pairs ($\mathbf{u}_i$, $\mathbf{u}_j$) between users.
Similarity triplets ($\mathbf{t}_i$, $\mathbf{t}_j$, s_{ij}) between tweets.
Stance-indicative regexes **R**.
// Create an initial dataset.
Training set **X** = $\{\}$.
Harvest positive and negative tweets based on **R**.
Add the harvested tweets to **X**.
// Augment the dataset by the relational classifier.
Learning & inference over $P(\mathbf{U}, \mathbf{T}|\mathbf{X})$ by our HL-MRF.
Add some classified tweets to training set: **X** = **X** + **T**.
Output: X.

Favor. make(?)america(?)great(?)again, #trumpforpresident, I{'m, am} voting trump, #illegal(.*), patriot, #boycottmacy
Against. racist, bigot, idiot, hair, narcissis(.+)

Table 1: Patterns to collect pro-Trump and anti-Trump tweets.

This task's goal was to test stance towards the target in 707 tweets. The authors in the test set are not identified, which prevents us from pursuing a fully relational approach. Thus, we adopt a two-phase approach: First, we predict the stance of the training tweets using our HL-MRF. Second, we use the labeled instances as training for a linear text classifier. This dataset-augmenting procedure is summarized in the Algorithm *Relational Bootstrapping*.

4.2 Experimental Setup

We pick the pro-Trump and anti-Trump indicative regular expressions and hashtags, which are shown in Table 1. Tweets that have at least one positive or one negative pattern, and do not have both positive and negative patterns, are considered as our initial positive and negative instances. This gives us a dataset with noisy labels; for example, the tweet "*his #MakeAmericaGreatAgain #Tag is a bummer.*" is against Donald Trump, incorrectly labeled favorable. A quantitative analysis of the impact of noise, and the goodness of initial patterns, can be pursued in the future through a supervised approach.

Tweets in the "neither" class range from news about the target of interest, to tweets totally irrele-

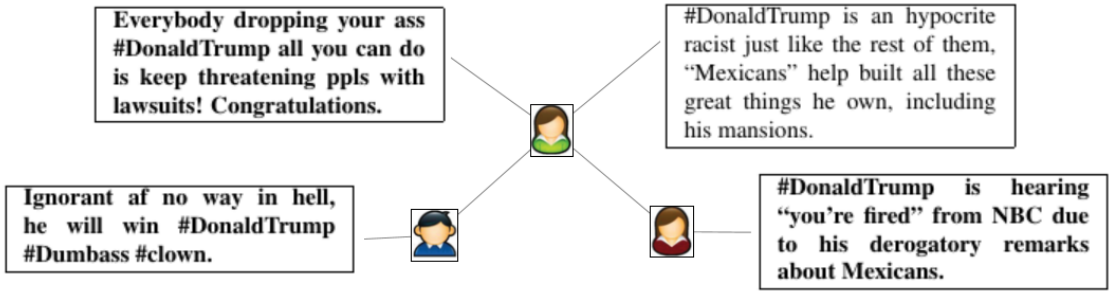


Figure 1: An example of the output of our relational bootstrapper. A small excerpt of the network, consisting of three users, four tweets and two friendship links. The tweet in regular type face is labeled as anti-Trump in the first phase, because of the word "racist" in the tweet. The other tweets, which are in boldface, are found through SRL harvesting, and are automatically labeled as anti-Trump tweets correctly.

vant to him. This makes it difficult to collect neutral tweets, and we will classify tweets to be in that class based on a heuristic described in the next subsection.

Given the limited number of seeds, we need to collect more training instances to build a stance classifier. Because of the original noise in the labels and the imposed fragmentary view of data, *self-learning* would perform poorly. Instead, we augment the dataset with tweets that our relational model classifies as positive or negative with a minimum confidence (class value 0.52 for pro-Trump and 0.56 for anti-Trump). The hyper-parameters were found through experimenting on a development set, which was the stance-annotated dataset of SemEval Task 6.a. The targets of that dataset include Hillary Clinton, Abortion, Climate Change, and Athesim. Since there are more anti-Trump tweets than pro-Trump (Mohammad et al., 2016), for our grid search we prefer a higher confidence threshold for the anti-Trump class, making it harder for the class bias to adversely impact the quality of harvested tweets. We also exclude the tweets that were sent by a user with no friends in the network. An example which showcases relational harvesting of tweets can be seen in Figure 1, wherein given the evidence, some of which is shown, three new tweets are found.

4.3 Classification

We convert the tweets to lowercase, and we remove stopwords and punctuation marks. For tweet classification, we use a linear-kernel SVM, which has proven to be effective for text classification and robust in high-dimensional spaces. We use the implementation of Pedregosa et al. (2011), and we employ the features below, which are normalized to unit length after conjoinment.

No. total tweets	21,000
No. initial pro tweets	1,100
No. initial anti tweets	1,490
No. relational-harvested pro tweets	960
No. relational-harvested anti tweets	780
No. edges in tweet similarity network	7,400
No. edges in friend network	131,000

Table 2: Statistics of the data

N-grams: `tf-idf` of binary representation of word n-grams (1–4 gram) and character n-grams (2–6 gram). After normalization, we only pick the top 5% most frequent grams.

Lexicon: Binary indicators of positive-emotion and negative-emotion words in LIWC2007 categories (Tausczik and Pennebaker, 2010).

Sentiment: Sentiment distribution, based on a sentiment analyzer for tweets, VADER (Hutto and Gilbert, 2014).

Table 3 demonstrates the results of stance classification. The metrics used are the macro-average of the F1-score for favor, against, and average of these two. The best competing system for the task used a deep convolutional neural network to train on pro and against instances, which were collected through linguistic patterns. At test time, they randomly assigned the instances, about which the classifier was less confident, to the "neither" class. Another base-

Method	F_{favor}	$F_{against}$	F_{avg}
SVM-ngrams-comb	18.42	38.45	28.43
best-system	**57.39**	55.17	56.28
SVM-IN	30.43	59.52	44.97
SVM-NB	47.67	57.53	52.60
SVM-RB	52.14	59.26	55.70
SVM-RB-N	54.27	**60.77**	**57.52**

Table 3: Evaluation on SemEval-2016 Task 6.b.

line is an SVM, trained on another stance classification dataset (Task 6.a), using a combination of n-gram features (SVM-ngrams-comb).

SVM-IN is trained on the initial dataset created by linguistic patterns, SVM-RB is trained on the relational-augmented dataset, and SVM-NB is a naive bootstrapping method that simply adds more instances, from the users in the initial dataset, with the same label as their tweets in the initial dataset, and for those who have both positive and negative tweets, does not add more of their tweets.

At test time, we could predict an instance to be of the "neither" class if it contains none of our stance-indicative patterns, nor any of the top 100 word grams that have the highest `tf-idf` weight in the training set. SVM-RB-N follows this heuristic for the "neither" class, while SVM-RB ignores this class altogether.

4.4 Demographics of the Users

As an application of stance classification, we analyze the demographics of the users based on their profile information. Due to the demographics of Twitter users, one has to be cautious about drawing generalizing conclusions from the analysis of Twitter data. We pick a balanced set of 1000 users with the highest degree of membership to any of the two groups. In Figure 2, we plot states represented by at least 50 users in the dataset. We can see that the figure correlates with US presidential electoral politics; supporters of Trump dominate Texas, and they are in the clear minority in California.

5 Conclusions and Future Work

In this paper, we propose a weakly supervised stance classifier that leverages the power of relational learning to incorporate extra features that are generally present on Twitter and other social media, i.e., authorship and friendship information. HL-MRFs enables us to use a set of hard and soft linear constraints to employ both the noisy-labeled instances and background knowledge in the form of soft constraints for stance classification on Twitter.

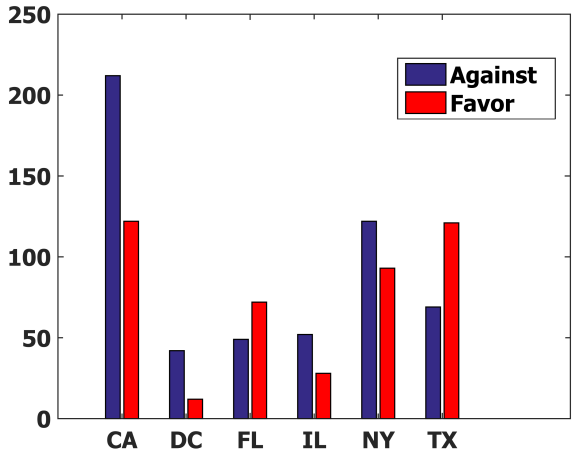


Figure 2: Distribution of Twitter users in a number of states.

While the relational learner tends to smooth out the incorrectly labeled instances, this model still suffers from noise in the labels. Labeling features and enforcing model expectation can be used to alleviate the impact of noise; currently, the initial linguistic patterns act as hard constraints for the label of the tweets, which can be relaxed by techniques such as generalized expectation (Druck et al., 2008).

The SemEval dataset has only one target of interest, Donald Trump. But the target of the opinion in the tweet may not necessarily be him, but related targets, such as Hillary Clinton and Ted Cruz. Thus, automatic detection of targets and inferring the stance towards all of the targets is the next step toward creating a practical weakly-supervised stance classifier.

6 Acknowledgments

This work was supported by NIH grant R01GM103309 and ARO grant W911NF-15-1-0265. We would like to thank anonymous reviewers for their helpful comments, Saed Rezayi for helping with Twitter API, and Ellen Klowden for discussions.

References

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of WWW*, pages 529–535.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 1–9.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss Markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG].

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of ACL*, pages 1506–1515.

Amparo Elizabeth Cano Basave, Yulan He, Kang Liu, and Jun Zhao. 2013. A weakly supervised Bayesian model for violence detection in social media. In *Proceedings of IJCNLP*, pages 109–117.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of SIGIR*, pages 595–602.

Lise Getoor. 2007. *Introduction to statistical relational learning*. MIT press.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? Identifying and classifying reasons in ideological debates. In *Proceedings of EMNLP*, pages 751–762.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of ICWSM*, pages 216–225.

David Jurgens. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *Proceedings of ICWSM*, pages 273–282.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2014. Inferring user preferences by probabilistic logical reasoning over social networks. arXiv:1411.2679 [cs.SI].

Daniel Lowd and Pedro Domingos. 2007. Efficient weight learning for Markov logic networks. In *Proceedings of PKDD*, pages 200–211.

Saif M Mohammad, Xiaodan Zhu, Svetlana Kiritchenko, and Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management*, 51(4):480–499.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*, pages 31–41.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015. Twitter user geolocation using a unified text and network prediction model. In *Proceedings of ACL*, pages 630–636.

Ashwin Rajadesingan and Huan Liu. 2014. Identifying users with opposing opinions in Twitter debates. In *Proceedings of SBP*, pages 153–160.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of ACL*, pages 116–125.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of NAACL-HLT*, pages 592–596.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of EMNLP*, pages 1046–1056.

The Gun Violence Database: A new task and data set for NLP

Ellie Pavlick[1] **Heng Ji**[2] **Xiaoman Pan**[2] **Chris Callison-Burch**[1]
[1]Computer and Information Science Department, University of Pennsylvania
[2]Computer Science Department, Rensselaer Polytechnic Institute

Abstract

We argue that NLP researchers are especially well-positioned to contribute to the national discussion about gun violence. Reasoning about the causes and outcomes of gun violence is typically dominated by politics and emotion, and data-driven research on the topic is stymied by a shortage of data and a lack of federal funding. However, data abounds in the form of unstructured text from news articles across the country. This is an ideal application of NLP technologies, such as relation extraction, coreference resolution, and event detection. We introduce a new and growing dataset, the Gun Violence Database, in order to facilitate the adaptation of current NLP technologies to the domain of gun violence, thus enabling better social science research on this important and under-resourced problem.

1 Introduction

The field of natural language processing often touts its mission as harnessing the information contained in human language: taking unstructured data in the form of speech and text, and transforming it into information that can be searched, categorized, and reasoned about. This is an ambitious goal, and the current state-of-the-art of language technology has made impressive strides towards understanding "who did what to whom, when, where, how, and why" (Kao and Poteet, 2007). Advances in NLP have enabled us to read news in real time (Petrović et al., 2010), identify the key players (Ruppenhofer et al., 2009), recognize the relationships between them (Riedel et al., 2013), summarize the new information (Wang et al., 2016), update central databases (Singhal, 2012), and use those databases to answer questions about the world (Berant et al., 2013).

Although these technological achievements are profound, often times we as researchers apply them to somewhat trivial settings like learning about the latest Hollywood divorces (Wijaya et al., 2015) or learning silly facts about the world, like that ⟨*white suites*, *will never go out of*, *style*⟩ (Fader et al., 2011). In this paper, we call the attention of the NLP community to one particularly good use case of our current technology, which could have profound policy implications: gun violence research.

Gun violence is an undeniable problem in the United States, but its causes are poorly understood, and attempts to reason about solutions are often marred by emotions and political bias. Research into the factors that cause and prevent gun violence is limited by the fact that data collection is expensive, and political agendas have all but eliminated funding on the topic. However, in the form of unstructured natural language published daily by newspapers across the country, data abounds. We argue that this is the exact type of information that NLP is designed to organize, and the positive social impact of doing so would be substantial. We introduce the Gun Violence Database (GVDB), a new dataset of gun violence articles paired with NLP annotations. Our hope is that the GVDB will facilitate the adaptation of core NLP technologies to the domain of gun violence. In turn, we believe these NLP technologies can help overcome the data vacuum that is currently preventing productive discussion about gun violence and its possible solutions.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1018–1024,
Austin, Texas, November 1-5, 2016.

What we have:	Daily reports of gun violence, published as free text by local newspapers and TV stations.
What we need:	Structured, queryable database with one record per incident.

Information Retrieval: Find articles about gun violence.
Event Detection: Identify precise incident being reported.
Temporal Annotation: Pinpoint precise time of the event.
NER: Extract key locations and participants from the event.
Semantic Role Labeling: Relate participants to their role in the incident (e.g. shooter, victim).
With-document Coref: Resolve mentions to consistently model each participant throughout the event.
Semantic Parsing: Extract precise, detailed information about participants, e.g. race, age, and gender.
Cross-document Coref: Recognize mentions of the same shooter or victim appearing in different articles.
Event Coref: Identify articles reporting the same event, and resolve to a single database entry.

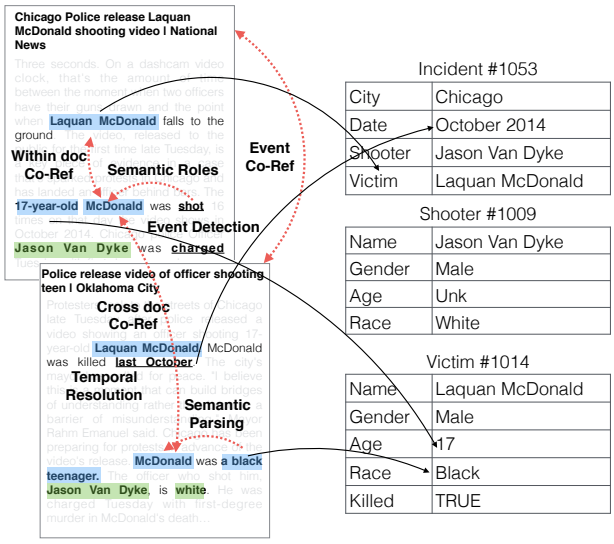


Figure 1: Turning daily news reports into usable data for public health and social science researchers is a textbook application of NLP technologies, and one that can have meaningful social impact.

2 Gun Violence's Data Problem

It is not difficult to motivate why gun violence is an important problem for research. Gun violence causes approximately 34,000 deaths in the US every year and more than twice as many injuries (FICAP, 2006), with violence especially high among young people and racial minorities (CDC, 2013).

The magnitude of the gun violence problem, the inherent gravity of the topic, and that fact that it inevitably leads to discussion of race, personal safety, and constitutional rights, makes the topic highly emotional and politically charged. Research into such hot-blooded topics stands to benefit immensely from data. In the past decade, machine learning researchers have championed data-driven decision making in place of oft-fallible human intuition. This approach has revolutionized the way we design and evaluate the effectiveness of business practices (Brynjolfsson et al., 2011; Kohavi et al., 2009), advertisements (Breese et al., 1998), and political campaigns (Issenberg, 2013). Gun violence policy should be no different. The problem is that researchers lack the data they need to answer the questions they want to ask. There is no single database[1] of gun violence incidents in the US, and the data that is available is mostly aggregated at the state level. Without locally-aggregated data, it is impossible to conduct meaningful studies of how firearm injury varies by community, a key step toward designing good policies for prevention (FICAP, 2006). However, for the past 25 years, research in this area has been, in the best case, massively underfunded (Roth et al., 1993) and in the worst case, actively blocked by federal legislation (Kassirer, 1995; Frankel, 2015; Bertrand, 2015). As a result, federal resources for gun violence research are orders of magnitude lower than is warranted (Branas et al., 2005), and there is no near-term likelihood of a federally-funded effort to collect detailed datasets to facilitate gun violence research.

Why NLP? Local newspapers and television stations report daily on gun injuries and fatalities. Many of these stories never make national news, but they represent precisely the kind of high-resolution data that epidemiologists need. The details of these reports could transform gun violence research if they were in a structured database, rather than spread

[1]There are 13 national data systems in the U.S., managed by separate federal agencies. The National Violent Death Registry System, arguably the most organized effort, receives data from only 16 states. Most large-scale epidemiological studies sample information from only 100 Emergency Departments.

across the text of thousands of web pages.

Replacing expensive, manual data entry with automated processing is exactly the type of problem that NLP is made to solve. In fact, the recent application of NLP tools to social science problems has generated a flurry of exciting and encouraging results. NLP has made novel contributions to the way scientists measure everything from income (Preoctiuc-Pietro et al., 2015b) to mental health (Preoctiuc-Pietro et al., 2015a; Schwartz et al., 2016; Choudhury et al., 2016), disease (Santillana et al., 2015; Ireland et al., 2015; Eichstaedt et al., 2015), and the quality of patient care (Nakhasi et al., 2016; Ranard et al., 2016).

Text mining has promise for the study of gun violence, too (Bushman et al., 2016). However, most questions about gun violence are not easily answered using shallow analyses like topic models or word clusters. Epidemiologists want to know, for example, does gun ownership lead to increases in gun violence? Or, is there evidence of contagion in suicides, and if so, does the style of reporting on suicides affect the likelihood that others will commit suicide after the initial event? Answering these questions requires extracting precise information from text: identifying entities, their actions, and their attributes specifically and reliably.

We believe this level of depth is well within the reach of current NLP technology. The state-of-the-art tools that NLP researchers have been building and fine-tuning for decades are an ideal fit for the problem described. Nearly every step of this process, from retrieving articles about gun violence to correctly determining whether the phrase *14 year old girl* describes the victim or the shooter, has been studied as a core NLP problem in its own right (Figure 1). These NLP tools have the potential to make a marked difference for gun violence researchers.

3 The Gun Violence Database

In order to facilitate the adaptation of NLP tools for use in gun violence research, we introduce the Gun Violence Database[2] (GVDB), a dataset for training and evaluating the performance of NLP systems in the domain of gun violence. The GVDB is the result of a large crowdsourced annotation effort. This annotation is ongoing, and the GVDB will be regularly updated with new data and new layers of annotation, making it an interesting and challenging data set on which to evaluate state-of-the-art NLP tools.

Crowdsourced Annotation The GVDB is built and updated through a continuously running crowdsourced annotation pipeline. The pipeline consists of daily crawls of local newspapers and television websites from across the US. The crawled articles are automatically classified using a high-recall text classifier, and then manually vetted by humans to filter out false positives. So far, the GVDB contains 60K articles (∼49M words) describing incidents of gun violence, and is (sadly) growing at a rate of nearly 1,000 per day.

Crowdsourced annotators then mark up the text of the articles with the key information we expect automated NLP systems to extract. In addition to classifying articles according to multiple binary dimensions (e.g. whether or not the shooting was intentional), annotators mark specific spans of the text which populate the database schema. For example, workers highlight the shooters, the victims, and the location.[3] These precise spans are stored in the database so that automated systems can be trained to reproduce the extracted information. Our annotation interface is shown in Figure 2.

At the time of writing, the GVDB contains 7,366 fully annotated articles (Table 1) coming from 1,512 US cities, and the database is continuing to grow. The latest version of the database will be maintained and available for download at `http://gun-violence.org/`.

60,443	Articles reporting incidents of gun violence
7,366	Articles fully-annotated for IE
6,804	*w/ location information*
5,394	*w/ shooter/victim information*
4,143	*w/ temporal information*
1,666	*w/ weapon information*

Table 1: Current contents of the GVDB. Size and level of annotation is continually growing. See Forthcoming Extensions.

Current Baselines To establish a baseline level of performance, we run an off-the-shelf information

[2] `http://gun-violence.org/`

[3] See supplementary material for all extracted information and screenshots.

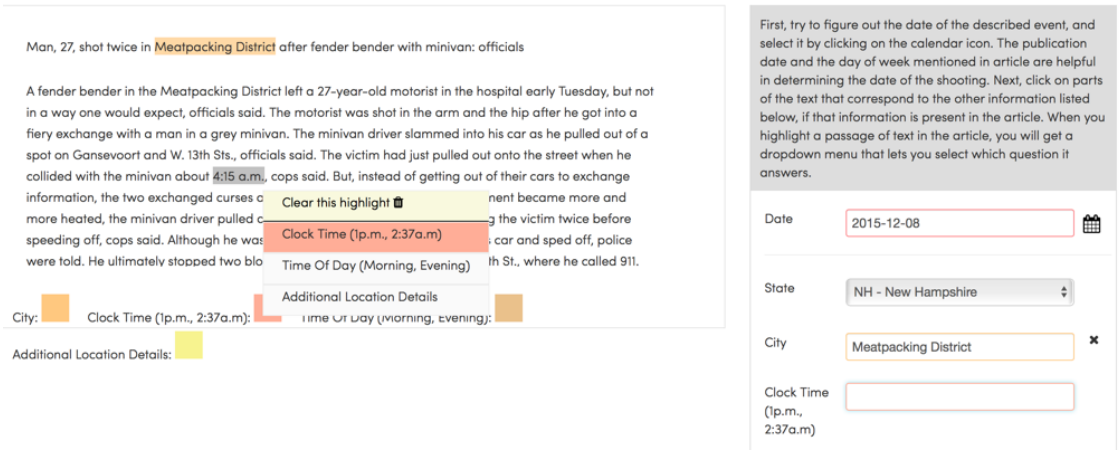


Figure 2: Annotation interface associates structured information (e.g. the time of day when the shooting occurred) with a specific span of text in the article.

extraction system on the 7,366 articles and measure precision and recall for identifying key information about the incidents. We use the Li et al. (2013) systems, which identifies a range of entities and events. We focus on the those events identified by the system which are relevant to the main fields in the GVDB schema.[4] We map the arguments of these events onto the corresponding database fields, e.g. the *agent* of the event corresponds to the GVDB's *shooter name*. Since the system identifies multiple such events per article, we count it as correct as long as one argument correctly matches the corresponding value in the GVDB (e.g. the system is correct as long as one extracted event has an *agent* which matches the GVDB's *shooter name* for that article). In addition, we run the Stanford CoreNLP TimeEx system (Chang and Manning, 2012) over the articles in order to identify the time of the reported incident.

We report the system's performance using both exact match against the gold annotation ("strict") as well as an approximate match, in which the system is correct if it is either a substring or a superstring of the gold annotation. E.g. if the victim name is *Sean Bolton*, the approximate metric will count both *Bolton* and *Officer Sean Bolton* as correct.

While performance is high for certain structured types of information, like dates and times, fields like victim and shooter name are much less reliably identified. Furthermore, many key pieces of information in the GVDB, such as age and race, are not supported by the off-the-shelf system. These baselines are evidence that NLP systems have potential, but require some effort to make their output usable for downstream research. Our hope is that the GVDB will serve as the impetus for undertaking this effort.

	Strict		Approx.	
	Prec.	Rec.	Prec.	Rec.
Date/Time	69.3%	66.9%	70.5%	68.1%
Location	19.9%	8.8%	30.8%	13.6%
Victim	10.2%	8.5%	59.5%	49.6%
Shooter	5.8%	3.9%	30.2%	20.1%
Weapon	2.1%	0.7%	36.8%	11.8%

Table 2: Performance of an off-the-shelf IE system on identifying key information about gun violence incidents from news articles. For "strict" vs. "approximate", see text.

Forthcoming Extensions The building of the GVDB is an ongoing effort, with new articles and deeper annotation being continuously added. We are currently adding approximately 300 new fully-annotated articles per day, while simultaneously enriching the annotation pipeline. The GVDB is soon to include annotation for event coreference, which will link articles describing the same incident, and cross-document coreference, which will link mentions of the same shooter/victim appearing in separate documents. In the future, the database will also include full within-document coreference annotation, with all mentions of a shooter/victim being flagged as such, and will incorporate visual data, so that within-article images are tagged with relevant

[4] Specifically, we focus on *Attack*, *Injure*, and *Die* events

information which may not be communicated by the text alone (e.g. race/approximate age).

4 Related Efforts

Several projects collect data about gun violence via newspaper teams (Boyle, 2013; Swaine et al., 2015) or volunteer crowds (Burghart, 2014; Wagner, 2014; Kirk and Kois, 2013). Perhaps the largest such effort is the Gun Violence Archive[5]. However, none are aimed at the eventual automation of the process. We believe that automating this data collection is key to keeping it scalable, consistent, and unbiased. Our focus is therefore on collecting data that is well-suited for training and evaluating NLP systems.

5 Conclusion

We believe that NLP researchers have the potential to significantly advance gun violence research. The shortage of data and funding for studying gun violence in America has severely limited the ability of scientists to have productive conversations about practical solutions. Applying core NLP technologies to local news reports of gun violence could transform raw text into structured, queryable data that public health researchers can use. We have introduced the Gun Violence Database, a new dataset of gun violence articles with rich NLP annotations which will support efforts on this new NLP task.

Acknowledgments

We would like to thank Douglas Wiebe for his advice and insight on building a useful resource for public health researchers. We also thank the students of the University of Pennsylvania's crowdsourcing class (NETS 213) for their involvement in building and testing a useful crowdsourcing pipeline for information extraction. Finally, we thank the developers at 10clouds for the excellent engineering and design of `http://gun-violence.org/`.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.

Natasha Bertrand. 2015. Congress quietly renewed a ban on gun-violence research. *Business Insider (July 7)*.

Andy Boyle. 2013. Mapping Chicago's shooting victims (Chicago Tribune), July.

Charles C Branas, Douglas J Wiebe, CW Schwab, and TS Richmond. 2005. Getting past the f word in federally funded public health research. *Injury prevention*, 11(3):191–191.

John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52.

Erik Brynjolfsson, Lorin M Hitt, and Heekyung Hellen Kim. 2011. Strength in numbers: How does data-driven decisionmaking affect firm performance? *Available at SSRN 1819486*.

Brian D. Burghart. 2014. What I've learned from two years collecting data on police killings, August.

Brad J. Bushman, Katherine Newman, Sandra L. Calvert, Geraldine Downey, Mark Dredze, Michael Gottfredson, Nina G. Jablonski, Ann S. Masten, Calvin Morrill, Daniel B. Neill, Daniel Romer, and Daniel W. Webster. 2016. Youth violence: What we know and what we need to know. *American Psychologist*, 71(1):17–39, Jan.

CDC. 2013. Deaths: Final data for 2013. *National vital statistics reports: from the Centers for Disease Control and Prevention, National Center for Health Statistics, National Vital Statistics System*, 64(2).

Angel X Chang and Christopher D Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740.

Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. 2016. Discovering shifts to suicidal ideation from mental health content in social media. In *Conference on Human Factors in Computing Systems (CHI)*.

Johannes C Eichstaedt, Hansen Andrew Schwartz, Margaret L Kern, Gregory Park, Darwin R Labarthe, Raina M Merchant, Sneha Jha, Megha Agrawal, Lukasz A Dziurzynski, Maarten Sap, et al. 2015. Psychological language on Twitter predicts county-level heart disease mortality. *Psychological science*, 26(2):159–169.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*,

[5] `http://www.gunviolencearchive.org`

pages 1535–1545, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

FICAP. 2006. *Firearm injury in the US.* Online Resource Book from The Firearm and Injury Center at Penn.

Todd C Frankel. 2015. Why the CDC still isn't researching gun violence, despite the ban being lifted two years ago. *The Washington Post (January 14)*.

Molly E Ireland, Qijia Chen, H Andrew Schwartz, Lyle H Ungar, and Dolores Albarracin. 2015. Action tweets linked to reduced county-level HIV prevalence in the United States: Online messages and structural determinants. *AIDS and Behavior*, pages 1–9.

Sasha Issenberg. 2013. How president Obama's campaign used big data to rally individual voters. *Technology Review*, 116(1):38–49.

Anne Kao and Steve R Poteet. 2007. *Natural language processing and text mining*. Springer Science & Business Media.

Jerome P Kassirer. 1995. A partisan assault on science–the threat to the CDC. *New England journal of medicine*, 333(12):793–794.

Chris Kirk and Dan Kois. 2013. How many people have been killed by guns since Newtown?, December.

Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18(1):140–181.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria, August. Association for Computational Linguistics.

Atul Nakhasi, Sarah G Bell, Ralph J Passarella, Michael J Paul, Mark Dredze, and Peter J Pronovost. 2016. The potential of Twitter as a data source for patient safety. *Journal of Patient Safety*, Jan.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189, Los Angeles, California, June. Association for Computational Linguistics.

Daniel Preoctiuc-Pietro, Maarten Sap, H Andrew Schwartz, and Lyle H Ungar. 2015a. Mental illness detection at the World Well-Being Project for the CLPsych 2015 Shared Task. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, NAACL.

Daniel Preoctiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying User Income through Language, Behaviour and Affect in Social Media. *PLoS ONE*, 10(9), 09.

Benjamin L Ranard, Rachel M Werner, Tadas Antanavicius, H Andrew Schwartz, Robert J Smith, Zachary F Meisel, David A Asch, Lyle H Ungar, and Raina M Merchant. 2016. Yelp reviews of hospital care can supplement and inform traditional surveys of the patient experience of care. *Health Affairs*, 35(4):697–705.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.

Jeffrey A Roth, Albert J Reiss Jr, et al. 1993. *Understanding and preventing violence*, volume 1. National Academies Press.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado, June. Association for Computational Linguistics.

Mauricio Santillana, Andre T. Nguyen, Mark Dredze, Michael J. Paul, Elaine Nsoesie, and John S. Brownstein. 2015. Combining search, social media, and traditional data sources to improve influenza surveillance. *PLOS Computational Biology*.

H Andrew Schwartz, Maarten Sap, Margaret L Kern, Johannes C Eichstaedt, Adam Kapelner, Megha Agrawal, Eduardo Blanco, Lukasz Dziurzynski, Gregory Park, David Stillwell, Michal Kosinski, Martin EP Seligman, and Lyle H. Ungar. 2016. Predicting Individual Well-Being Through the Language of Social Media. *Pacific Symposium on Biocomputing*, 21:516–527.

Amit Singhal. 2012. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*.

Jon Swaine, Oliver Laughland, Jamiles Lartey, and Ciara McCarthy. 2015. The counted: People killed by police in the US (The Guardian), June.

Kyle Wagner. 2014. We're compiling every police-involved shooting in America. Help us., August.

William Yang Wang, Yashar Medhad, Dragomir Radev, and Amanda Stent. 2016. A low-rank approximation

approach to learning joint embeddings of news stories and images for timeline summarization. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, CA, USA. ACL.

Derry Tanti Wijaya, Ndapandula Nakashole, and Tom Mitchell. 2015. A spousal relation begins with a deletion of engage and ends with an addition of divorce: Learning state changing verbs from Wikipedia revision history. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 518–523, Lisbon, Portugal, September. Association for Computational Linguistics.

Fluency detection on communication networks

Tom Lippincott and **Benjamin Van Durme**
Human Language Technology Center of Excellence
Johns Hopkins University
tom@cs.jhu.edu, vandurme@cs.jhu.edu

Abstract

When considering a social media corpus, we often have access to structural information about how messages are flowing between people or organizations. This information is particularly useful when the linguistic evidence is sparse, incomplete, or of dubious quality. In this paper we construct a simple model to leverage the structure of Twitter data to help determine the set of languages each user is fluent in. Our results demonstrate that imposing several intuitive constraints leads to improvements in performance and stability. We release the first annotated data set for exploring this task, and discuss how our approach may be extended to other applications.

1 Introduction

Language identification (LID) is an important first step in many NLP pipelines since most downstream tasks need to employ language-specific resources. In many situations, LID is a trivial task that can be addressed e.g. by a simple Naive Bayes classifier trained on word and character n-gram data (Lui and Baldwin, 2012): a document of significant length will be quickly disambiguated based on its vocabulary (King et al., 2014). However, social media platforms like Twitter produce data sets in which individual documents are extremely short, and language use is idiosyncratic: LID performance on such data is dramatically lower than on traditional corpora (Bergsma et al., 2012; Carter et al., 2013). The widespread adoption of social media throughout the world amplifies the problem as less-studied languages lack the annotated resources needed to train the most effective NLP models (e.g. treebanks for statistical parsing, tagged corpora for part-of-speech tagging, etc). All of this motivates the research community's continued interest in LID (Zampieri et al., 2014).

Tweet #1	Коз эверисинг ю ду ис мэджик
Tweet #2	omg favourite day of the week!

Table 1: Multilingual social media users often communicate in different languages depending on their intended audience, such as with these Russian and English tweets posted by the same Twitter account

In this paper, we consider the closely-related task of determining an actor's *fluencies*, the set of languages they are capable of speaking and understanding. The observed language data will be the same as for LID, but is now considered to indicate a latent property of the actor. This information has a number of downstream uses, such as providing a strong prior on the language of the actor's future communications, constructing monolingual data sets, and recommending appropriate content for display or further processing.

This paper also focuses on the situation where a very small amount of content has been observed from the particular user. While this may seem strange considering the volume of data generated by social media, this is dominated by particularly active users: for example, 30% of Twitter users post only once per month (Leetaru et al., 2013). This content-starved situation is exacerbated by certain use-cases, such as responding to emergency events where sudden focus is directed at a particular location, or focusing on new users with shallow histories.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1025–1029,
Austin, Texas, November 1-5, 2016.

2 Previous Work

Twitter and other social media platforms are a major area of ongoing NLP research, including dedicated workshops(NAA, 2015; ACL, 2014). Previous work has considered macroscopic properties of the entire Twitter network (Gabielkov et al., 2014), and pondered whether it is an "information" or "social" network (Myers et al., 2014). Studies have focused on determining user attributes such as gender (Li et al., 2015), political allegiance (Volkova et al., 2014), brand affinity (Pennacchiotti and Popescu, 2011a), sentiment analysis (West et al., 2014), and more abstract roles (Beller et al., 2014). Such demographic information is known to help downstream tasks (Hovy, 2015). Research involving social media communication networks has typically focused on *homophily*, the tendency of users to connect to others with similar properties (Barberá, 2014). A number of papers have employed features drawn from both the content and structure of network entities in pursuit of latent user attributes (Pennacchiotti and Popescu, 2011b; Campbell et al., 2014; Suwan et al., 2015).

3 Definitions

We refer to the entities that produce and consume communications as *Actors*, and the communications (packets of language data) as *Messages*. Each message occurs in a particular *Language*, and each actor has a set of *Fluencies*, representing the ability to produce and consume a message in a given language. We refer to a connected graph of such entities as a *Communication Network*. For Twitter data, messages are simply associated with a single actor, who is in turn associated with other actors via the "following" relationship, the actor's "friends" in Twitter's terminology.[1] We assume each message (tweet) is written in a single language, and actors are either fluent or not in each possible language.

[1]Note that, confusingly, Twitter's "friend" relationship is not symmetric: Mary's friends are users she has decided to follow, and not necessarily vise-versa.

4 Twitter Data Set

To build a suitable data set[2] for fluency detection, we first identified 1000 Twitter users who, according to the Twitter LID system, have tweeted in Russian and at least one additional language. For each of these "seed" users, we gather a local context (a "snowflake") as follows: we choose 20 of their friends at random. For each of these friends, we choose 15 of *their* friends (again, at random). Finally, we randomly pull 200 tweets for each identified user. The data set consists of 989 seed users, 165,042 friends, and 55,019,811 tweets. We preserve all Twitter meta-data for the users and tweets, such as location, follower count, hashtags, etc, though for the purposes of this paper we are only interested in the friendship structure and message text. We then had an annotator determine the set of languages each of the 1000 seed users is fluent in. For each seed user, the annotator was presented with their 200 tweets, grouped by Twitter language ID, and was asked to 1) flag users that appear to be bots and 2) list the languages they believe the user is fluent in. These steps are reflected in Figure 4. Over 50% (507) of the users were flagged as possible bots and not used in this study. The remaining 482 were observed employing 7 different languages: Russian, Ukrainian, German, Polish, Bulgarian, Latvian, and English. At most, a single user was found to be fluent in three languages.

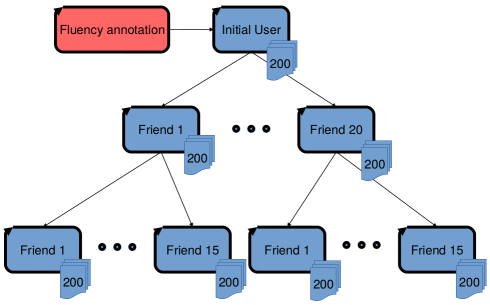


Figure 1: Structure of one snowflake in the Twitter Fluency data set.

5 Structure-Aware Fluency Model

Our goal was to explicitly model each actor's fluency in different languages, using a model with sim-

[2]The full data set is available at `www.university.edu/link`

ple, interpretable parameters that can be used to encode well-motivated assumptions about the data. In particular, we want to bias the model towards the belief that actors typically speak a small number of languages, and encode the belief that all actors participating in a message are highly likely to be fluent in its language. Our basic hypothesis is that, in addition to scores from traditional LID modules, such a model will benefit from considering the behavior of an actor's interlocutors. To test this, we designed a model that employs scores from an existing LID system, and compare performance with and without awareness of the communication network structure. To demonstrate the effectiveness of the model in situations with sparse or unreliable linguistic content, we perform experiments where the number of messages associated with each actor has been randomly down-sampled.

Linear Programming Linear Programming (LP) is a method for specifying constraints and cost functions in terms of linear relationships between variables, and then finding the optimal solution that respects the constraints. The restriction to linear equations ensures that the objective function is itself linear, and can be efficiently solved. If some or all variables are restricted to take discrete values, referred to as (Mixed) Integer Linear Programming (ILP), finding a solution becomes NP-hard, though common special cases remain efficiently solvable. We specify our model as an ILP with the hope that it provides sufficient expressiveness for the task, while remaining intuitive and tractable. Inference is performed using the Gurobi modeling toolkit (Gurobi Optimization, 2015).

Model definition Given a communication network with no LID information, ideally we would like to determine the language of each message, and the set of languages each actor is fluent in. Initially, we assume access to a probablistic LID system that maps unicode text to a distribution over possible languages. We use the following notation: $A_{1:T}$ and $M_{1:U}$ are the actors and messages, respectively. $F(a_i)$ is a binary vector indicating which languages we believe actor a_i is fluent in. $L(m_i)$ is a one-on binary vector indicating which language we believe message m_i is written in. $P(m_i)$ is the set of actors participating in message m_i: for Twitter data, where messages are (usually) not directed at specific users, we treat a user and the users' friends as participants. $LID(m_i)$ is a real vector representing the probability of message m_i being in each language, according to the LID system.

To build our ILP model, we iterate over actors and messages, defining constraints and the objective function as we go. There are two types of structural constraints: first, we restrict each message to have a single language assignment:

$$\sum L(m_i) = 1 \tag{1}$$

Second, we ensure that all actors participating in a given message are fluent in its language:

$$\forall a \in P(m_i), L(m_i) \times F(a) = 1 \tag{2}$$

The objective function also has two components: first, the *language fit* encourages the model to assign each message a language that has high probability according the the LID system:

$$\text{LF} = \sum_{m \in M} L(m) \times LID(m) \tag{3}$$

Second, the *structure fit* minimizes the cardinality of the actors' fluency sets (subject to the structural constraints), and thus avoids the trivial solution where each actor is fluent in all languages:

$$\text{SF} = -\sum_{a \in A} \sum F(a) \tag{4}$$

Finally, the two components of the objective function are combined with an empirically-determined *language weight* to get the complete objective function:

$$\text{LW} \times \text{LF} + (1.0 - \text{LW}) \times \text{SF} \tag{5}$$

Note that these are not all linear relationships: in particular, the multiplication operator cannot be used in ILP when the operands are both variables, as in equation 2. There are however techniques that can represent these situations in a linear program by introducing helper variables and constraints (Bisschop, 2015).

Language Identification Scores and Fluency Baseline To get LID scores, we ran the VaLID system (Bergsma et al., 2012) on each message, and normalize the output into distributions over 261 possible languages. VaLID is trained on Wikipedia data (i.e. out-of-domain relative to Twitter), although it does employ hand-specified rules for sanitizing tweet text, such as normalizing whitespace and removing URLs and user tags. VaLID uses a data-compression approach that is competitive with Twitter's in-house LID, despite no consideration of geographic or user priors. These language scores are used in the structure-aware model to compute the language fit.

Because VaLID makes no use of the communication network structure, we also use its scores to create a baseline structure-unaware fluency model. To get structure-unaware baseline scores for the fluency identification task, we average the LID distributions for each actor's messages and consider them fluent in a language if its probability is above an empirically-determined threshold.

Tuning parameters We empirically determine the thresholds for the baseline model and the language weights for the structure-aware model via a simple grid search, repeated 100 times. We randomly split the data into 20%/80% tune/test sets, and evaluate filter thresholds and language weights from 0 to 1 in .01 increments, with messages per actor ranging between 1 and 10. We expected the baseline model to have a consistent optimal threshold (though with higher performance variance with fewer messages), and this was borne out with optimal performance at a threshold of 0.06, independent of the number of messages per actor. For the structure-aware model, the optimal language weight was 0.9, although the entire range from 0.1–0.9 showed similar performance. This result was surprising, as we expect the structure-aware model to rely heavily on the structural fit when the number of messages is small, and on the language fit when the number is large. This trend doesn't emerge because the structural fit actually relies on the language fit to make assignments for the seed actor's friends and their messages.

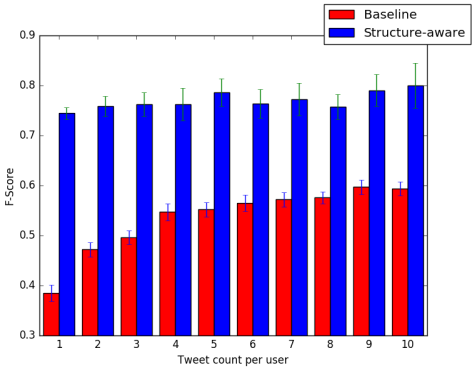


Figure 2: Performance of baseline and structure-aware models as a function of the number of messages per actor used as evidence. Each bar represents the average over 100 random tuning/testing splits, with whiskers showing the standard deviation.

6 Results and discussion

Figure 2 compares the performance[3] of the structure-aware ILP model with the baseline model as a function of the number of messages per actor, using the empirically-determined threshold and language weight. At the left extreme, the models only have a single, randomly-selected message from each actor. As this number increases, the baseline model improves as it becomes more likely to have seen enough messages to reflect the actor's full spectrum of language use. The structure-aware model is able to make immediate use of the actor's friends, immediately reaching high performance even when the language data is very sparse. Its most frequent type of error is over-hypothesizing fluency in both Ukrainian and Russian, when the user is in fact monolingual, followed by incorrectly hypothesizing fluency in English. This is understandable given the similarity of the languages in the former case, and the popularity of English expressions, titles, and the like in the latter.

7 Conclusion

We have presented promising results from leveraging structural information from a communication network to improve performance on fluency detection in situations where direct linguistic data is sparse. In addition to defining the task itself,

[3]F-score calculated based on correct and hypothesized fluency-assignments for each actor.

we release an annotated data set for training and evaluating future models. Planned future work includes a more flexible decoupling of the language and structure fits (in light of Section 5), and moving from pre-existing LID systems to joint models where LID scores are directly informed by structural information.

References

2014. ACL Joint Workshop on Social Dynamics and Personal Attributes in Social Media.

Pablo Barberá. 2014. Birds of the same feather tweet together: Bayesian ideal point estimation using twitter data. *Political Analysis*, 23:76–91.

Charley Beller, Rebecca Knowles, Craig Harman, Shane Bergsma, Margaret Mitchell, and Benjamin Van Durme. 2014. I'm a belieber: Social roles via self-identification and conceptual attributes. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*, pages 181–186, Baltimore, Maryland, USA.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific Twitter collections. In *Proc. Second Workshop on Language in Social Media*, pages 65–74.

Johannes Bisschop. 2015. Aimms optimization modeling.

W.M. Campbell, E. Baseman, and K. Greenfield. 2014. langid.py: An off-the-shelf language identification tool. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media*, pages 59–65, Dublin, Ireland.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Lang. Resour. Eval.*, 47(1):195–215, March.

Maksym Gabielkov, Ashwin Rao, and Arnaud Legout. 2014. Studying social networks at scale: Macroscopic anatomy of the twitter social graph. In *SIGMETRICS '14*, Austin, Texas, USA.

Inc. Gurobi Optimization. 2015. Gurobi optimizer reference manual.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 752–762, Beijing, China.

Ben King, Dragomir Radev, and Steven Abney. 2014. Experiments in sentence language identification with groups of similar languages. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 146–154, Dublin, Ireland.

Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Eric Shook. 2013. Mapping the global twitter heartbeat: The geography of twitter. *First Monday*, 18(5).

Shoushan Li, Jingjing Wang, Guodong Zhou, and Hanxiao Shi. 2015. Interactive gender inference with integer linear programming. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2341–2347. AAAI Press.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 25–30, Jeju, Republic of Korea.

Seth A. Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. 2014. Information network or social network? the structure of the twitter follow graph. In *WWW '14 Companion*, Seoul, Korea.

2015. NAACL International Workshop on Natural Language Processing for Social Media.

Marco Pennacchiotti and Ana-Maria Popescu. 2011a. Democrats, republicans and starbucks afficionados: User classification in twitter. In *KDD '11*, San Diego, California, USA.

Marco Pennacchiotti and Ana-Maria Popescu. 2011b. A machine learning approach to twitter user classification. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 281–288.

Shakira Suwan, Dominic Lee, and Carey Priebe. 2015. Bayesian vertex nomination using content and context. *WIREs Comput Stat*, 7:400–416.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*, pages 186–196, Baltimore, Maryland, USA.

Robert West, Hristo Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics*, 2:297–310.

Marcos Zampieri, Liling Tang, Nikola Ljubešić, and Jörg Tiedemann. 2014. Discriminating similar languages shared task at coling 2014.

Characterizing the Language of Online Communities and its Relation to Community Reception

Trang Tran and **Mari Ostendorf**
Electrical Engineering, University of Washington, Seattle, WA
{ttmt001,ostendor}@uw.edu

Abstract

This work investigates style and topic aspects of language in online communities: looking at both utility as an identifier of the community and correlation with community reception of content. Style is characterized using a hybrid word and part-of-speech tag n-gram language model, while topic is represented using Latent Dirichlet Allocation. Experiments with several Reddit forums show that style is a better indicator of community identity than topic, even for communities organized around specific topics. Further, there is a positive correlation between the community reception to a contribution and the style similarity to that community, but not so for topic similarity.

1 Introduction

Online discussion forums provide a rich source of data for studying people's language usage patterns. Discussion platforms take on various forms: articles on many news sites have a comment section, many websites are dedicated to question answering (www.quora.com), and other platforms let users share personal stories, news, and random discoveries (www.reddit.com). Like their offline counterparts, online communities are often comprised of people with similar interests and opinions. Online communication, however, differs from in-person communication in an interesting aspect: explicit and quantifiable feedback. Many discussion forums give their users the ability to upvote and/or downvote content posted by another user. These explicit reward/penalty labels provide valuable information on the reaction of users in a community. In this work, we take advantage of the available user response to explore the relationship between community reception and the degree of stylistic/topical coherence to such communities. Using hybrid n-grams and Latent Dirichlet Allocation (LDA) topic models to represent style and topic for a series of classification tasks, we confirm that there exists a *community language style*, which is not simply characterized by the *topics* that online communities are organized around. Moreover, we show that language style is better at discriminating communities, especially between different communities that happen to discuss similar issues. In addition, we found a positive, statistically significant, correlation between the community feedback to comments and their style, but interestingly not with their topic. Finally, we analyze community language on the user level and show that more successful users (in terms of positive community reception) tend to be more *specialized*; in other words, analogous to offline communities, it is rare for a person to be an expert in multiple areas.

2 Related Work

It is well known that conversation partners become more linguistically similar to each other as their dialogue evolves, via many aspects such as lexical, syntactic, as well as acoustic characteristics (Niederhoffer and Pennebaker, 2002; Levitan et al., 2011). This pattern is observed even when the conversation is fictional (Danescu-Niculescu-Mizil and Lee, 2011), or happening on social media (Danescu-Niculescu-Mizil et al., 2011). Regarding the language of online discussions in particular, it has been shown that individual users' linguistic patterns evolve to match

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1030–1035,
Austin, Texas, November 1-5, 2016.

those of the community they participate in, reaching "linguistic maturity" over time (Nguyen and Rosé, 2011; Danescu-Niculescu-Mizil et al., 2013). In a multi-community setting, Tan and Lee (2015) found that users tend to explore more in different communities as they mature, adopting the language of these new communities. These works have mainly focused on the temporal evolution of users' language. Our work differs in that we use different language models to explore the role of topic and style, while also considering users in multiple communities. In addition, we look at community language in terms of its correlation with reception of posted content.

Other researchers have looked at the role of language in combination with other factors in Reddit community reception. Lakkaraju et al. (2013) proposed a community model to predict the popularity of a resubmitted content, revealing that its title plays a substantial role. Jaech et al. (2015) considered timing and a variety of language features in ranking comments for popularity, finding significant differences across different communities. In our work, we focus on community language, but explore different models to account for it.

3 Data

Reddit is a popular forum with thousands of subcommunities known as *subreddits*, each of which has a specific theme. We will refer to *subreddits* and *communities* interchangeably. Redditors can submit content to initiate a discussion thread whose root text we will refer to as a *post*. Under each *post*, users can discuss the post by contributing a *comment*. Both posts and comments can be upvoted and downvoted, and the net feedback is referred to as *karma* points.

We use eight subreddits that reflect Reddit's diverse topics, while limiting the amount of data to a reasonable size. In addition, we create an artificial distractor merged_others that serves as an open class in our classification tasks and for normalizing scores in correlation analysis. Statistics are listed in Table 1. The merged_others set includes 9 other subreddits that are similar in size and content diversity to the previous ones: books, chicago, nyc, seattle, explainlikeimfive, science, running, nfl, and todayilearned. Among these extra subreddits, the smallest in size is nyc (1.5M tokens, 76K comments), and the largest is todayilearned (88M tokens, 5M comments). All data is from the period between January 1, 2014 and January 31, 2015. In each subreddit, 20% of the threads are held out for testing.

subreddit	# posts	# cmts	% $k \leq 0$
askmen	4.5K	1.1M	10.6
askscience	0.9K	0.3M	9.1
askwomen	3.6K	0.8M	7.5
atheism	3.1K	1.0M	15.2
changemyview	2.3K	0.5M	16.7
fitness	2.4K	0.9M	8.6
politics	4.9K	2.2M	20.8
worldnews	9.9K	6.0M	23.6
merged_others	28.0K	14.2M	13.2

Table 1: Reddit dataset statistics

We use discussion threads with at least 100 comments, hypothesizing that smaller threads will not elicit enough community personality for our study. (Virtually all threads kept had only upvotes.) For training our models, we also exclude individual comments with non-positive karma ($k \leq 0$) in order to learn only from content that is less likely to be downvoted by the Reddit communities; percentages are noted in Table 1.

4 Models

We wish to characterize community language via *style* and *topic*. For modeling style, a popular approach has been combining the selected words with part-of-speech (POS) tags to construct models for genre detection (Stamatatos et al., 2000; Feldman et al., 2009; Bergsma et al., 2012) and data selection (Iyer and Ostendorf, 1999; Axelrod, 2014). For topic, a common approach is Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We follow such approaches in our work, acknowledging the challenge of completely separating style/genre and topic factors raised previously (Iyer and Ostendorf, 1999; Sarawgi et al., 2011; Petrenz and Webber, 2011; Axelrod, 2014), which also comes out in our analysis. Generative language models are used for characterizing both style and topic, since they are well suited to handling texts of widely varying lengths.

4.1 Representing Style

Replacing words with POS tags reduces the possibility that the style model is learning topic, but re-

placing too many words loses useful community jargon. To explore this tradeoff, we compared four trigram language models representing different uses of words vs. POS tags in the vocabulary:

- `word_only`: a regular token-based language model (vocabulary: 156K words)
- `hyb-15k`: a hybrid word-POS language model over a vocabulary of 15K most frequent words across all communities in our data; all other words are converted to POS tags (vocabulary: 15K words + 38 tags)
- `hyb-500.30`: a hybrid word-POS language model over a vocabulary of 500 most frequent words in a subset of data balanced across communities, combined with the union of the 30 next most common words from each of the 17 subreddits; all other words are converted to POS tags (vocabulary: 854 words + 38 tags)
- `tag_only`: a language model using only POS tags as its vocabulary (vocabulary: 38 tags)

The hybrid models represent two intermediate sample points between the extremes of word-only and tag-only n-grams. For the `hyb-500.30` model, the mix of general and community-specific words was designed to capture distinctive community jargon. The general words include punctuation, function words, and words that are common in many subreddits (e.g., *sex, culture, see, dumb, simply*). The subreddit-specific words seem to reflect both broad topical themes and jargon or style words, as in (themes vs. style/jargon):

askmen: *wife, single* vs. *whatever, interested*
askwomen: *mom, husband* vs. *especially, totally*
askscience: *particle, planet* vs. *basically, x*
fitness: *exercises, muscles* vs. *cardio, reps, rack*

Tokenization and tagging are done using Stanford coreNLP (Manning et al., 2014). Punctuation is separated from the words and treated as a word. All language models are trigrams trained using the SRILM toolkit (Stolcke, 2002); modified Kneser-Ney smoothing is applied to the `word_only` language model, while Witten-Bell smoothing is applied to the `tag_only` and both hybrid models.

4.2 Representing Topic

We train 100- and 200-dimensional LDA topic models (Blei et al., 2003) using `gensim` (Řehůřek and Sojka, 2010). We remove all stopwords (250 words) and use tf-idf normalized word counts in each comment (as documents). The vocabulary consists of 156K words, similar to the vocabulary of the `word_only` language model. The topic models were trained on a subset of the training data, using all collected subreddits but randomly excluding roughly 15% of the training data of larger subreddits worldnews, todayilearned, and nfl.

ID	Frequent words
19	-lsb-, -rsb-, -rrb-, -lrb-, **, reddit, comment, confirmed, spanish, fair
29	sex, pilots, child, women, abortion, mail, birth, want, episodes, children
32	tax, government, taxes, iraq, pay, cia, land, money, income, people
34	africa, war, nation, global, germans, rebels, corruption, nations, fuel, world

Table 2: Examples of broadly used topics.

The topics learned exhibit a combination of ones that reflect general characteristics of online discussions or topics that arise in many forums, some that have more specific topics, and others that do not seem particularly coherent. Topics (from LDA-100) that consistently have high probability in all subreddits are shown in Table 2 with their top 10 words by frequency (normalized by the topic average). Topic 19 is likely capturing Reddit's writing conventions and formatting rules. Broadly used topics reflect women's issues (29) and news events (32, 34).

Online communities are typically organized around a common theme, but multiple topics might fall under that theme, particularly since some of the "topics" actually reflect style. A subreddit as a whole is characterized by a distribution of topics as learned via LDA, but any particular discussion thread would not necessarily reflect the full distribution. Therefore, we characterize each subreddit with multiple topic vectors. Specifically, we compute LDA topic vectors for each discussion thread in a subreddit, and learn 50 representative topic vectors for each subreddit via k-means clustering.

5 Community Classification

One method for exploring the relative importance of topic vs. style in online communication is through community classification experiments: given a discussion thread (or a user's comments), can we iden-

tify the community that it comes from more easily using style characteristics or topic characteristics? We formulate this task as a multi-class classification problem (8 communities and "other"), where samples are either at the discussion thread level or the user level. At the thread level, all comments (from multiple people) and the post in a discussion thread are aggregated and treated as a document to be classified. At the user level, we aggregate all comments made by a user in a certain subreddit and treat the collection (which may reflect multiple topics) as a document to be classified.

We classify document d_i to a subreddit according to $\hat{j} = \arg\max_j s_{i,j}$, where $s_{i,j}$ is a score of the similarity of d_i to community j. For the style models, $s_{i,j}$ is the log-probability under the respective trigram language model of community j. For the topic model, $s_{i,j}$ is computed using d_i's topic vector v_i as follows. For a subreddit j, we compute the cosine similarities $sim_{j,k}$ between v_i and the subreddit's topic vectors $w_{j,k}$ for $k = 1, \ldots, 50$. The final topic similarity score $s_{i,j}$ is the mean of the top 3 highest similarities: $s_{i,j} = (sim_{j,[1]} + sim_{j,[2]} + sim_{j,[3]})/3$, where $[\cdot]$ denotes the sorted cosine similarities' indices. The top-3 average captures the most prominent subreddit topics (as in a nearest-neighbor classifier). Averaging over all 50 $sim_{j,k}$ is ill suited to subreddits with broad topic coverage, and leads to poor classification results.

Table 3 summarizes the community classification results (as average accuracy across all subreddits) for each model described in Section 4. While all models beat the random baseline of 11%, the poor performance of the `tag_only` model confirms that POS tags alone are insufficient to characterize the community. Both for classifying threads and authors, `hyb-500.30` yields the best average classification accuracy, due to its ability to generalize POS structure while covering sufficient lexical content to capture the community's jargon and key topical themes. Neither topic model beats `hyb-500.30`, indicating that topic alone is not discriminative enough for community identification, even though specific communities coalesce around certain common topics. The `word_only` and `hyb-15k` models have performance on the threads that is similar to the topic models, since word features are sensitive to topic, as shown in (Petrenz and Webber, 2011).

Model	by thread	by author
random	11.1%	11.1%
word_only	68.9%	46.8%
tags_only	27.6%	18.8%
hyb-15k	69.4%	46.6%
hyb-500.30	**86.5%**	**51.0%**
topic-100	**71.1%**	27.5%
topic-200	69.6%	**27.7%**

Table 3: Average accuracy for classifying by posts and authors

Classifying authors is harder than classifying threads. Two factors are likely to contribute. First, treating a whole discussion thread as a document yields more data to base the decision on than a collection of author comments, since there are many authors who only post a few comments. Second, authors that have multi-community involvement may be less adapted to a specific community. The fact that word-based style models outperform topic models may be because the comments are from different threads so not matching typical topic distributions.

Subreddit confusion statistics indicate that certain communities are easier to identify than others. Both style and topic models do well in recognizing askscience: classification accuracy for threads is as much as 97%. Communities that were most confusable are intuitively similar: politics and worldnews, askmen and askwomen.

6 Community Feedback Correlation

In this section, we investigate whether the style and/or topic scores of a discussion or user are correlated with community response. For thread-level feedback, we use karma points of the discussion thread itself; for the user-level feedback, we compute each user's subreddit-dependent k-index (Jaech et al., 2015), defined similarly to the well-known h-index (Hirsch, 2005). Specifically, a user's k-index k_j in subreddit j is the maximum integer k such that the user has at least k comments with karma greater than k in that subreddit. User k-index scores have Zipfian distribution, as illustrated in Figure 1 for the worldnews subreddit.

We compute a *normalized* community similarity score $\tilde{s}_{i,j} = s_{i,j} - s_{i,m}$, where $s_{i,m}$ is the corresponding score from the subreddit merged_others. The correlation between $\tilde{s}_{i,j}$ and community feedback is reported for three models in Table 4 for the

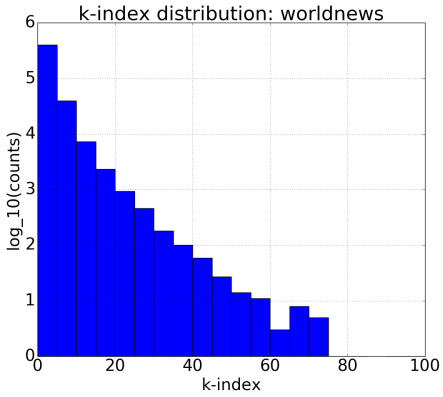


Figure 1: Distribution (log base 10 counts) of user k-index scores for the worldnews subreddit.

subreddit	hyb-500.30	word_only	topic-100
askmen	0.392*	0.222*	0.055
askscience	0.321*	-0.110	-0.166*
askwomen	0.501*	0.388*	0.005
atheism	0.137*	-0.229*	-0.251
chgmyvw	0.167*	-0.121*	-0.306*
fitness	0.130*	0.017	-0.313*
politics	0.533*	0.341*	0.011
worldnews	0.374*	0.148*	-0.277*

Table 4: Spearman rank correlation of thread $\tilde{s}_{i,j}$ with karma scores. (*) indicates statistical significance ($p < 0.05$).

thread level, and in Table 5 for the user level. On the thread level, the `hyb-500.30` style model consistently finds positive, statistically significant, correlation between the post's stylistic similarity score and its karma. This result suggests that language style adaptation does contribute to being well-received by the community. None of the other models explored in the previous section had this property, and for the topic models the correlation is mostly negative. On the user level, *all* correlations between a user's k-index and their style/topic match are statistically significant, though the `hyb-500.30` style model shows more positive correlation than other models. In both cases, the `word_only` model gives results between the style and topic models. The `hyb-15k` model has results that are similar to the `word_only` model, and the `tag_only` model has mostly negative correlation.

Examining users' multi-community involvement, we also find that users with high k-indices tend to participate in fewer subreddits. Among relatively

subreddit	hyb-500.30	word_only	topic-100
askmen	0.402	0.215	0.167
askscience	0.343	0.106	0.042
askwomen	0.451	0.260	0.165
atheism	0.296	0.024	0.107
chgmyvw	0.446	0.020	0.091
fitness	0.309	0.286	0.127
politics	0.453	0.317	0.177
worldnews	0.421	0.330	0.166

Table 5: Spearman rank correlation of authors' $\tilde{s}_{i,j}$ with their k-indices. All values are statistically significant ($p < 0.05$).

active users (having at least 100 comments), those with a max k-index of at least 100 participated in a median of 3 communities, while those with a max k-index of at most 5 participated in a median of 6 subreddits. Of the 42 users with max k-index of at least 100, only 4 achieve a k-index of at least 50 in one other community, and only 6 achieve a k-index of at least 20 in one other community.

7 Conclusion

In this work, we use hybrid n-grams and topic models to characterize style and topic of language in online communities. Since communities center on a common theme, topic characteristics are reflected in language style, but we find that the best model for determining community identity uses very few words and mostly relies on POS patterns. Using Reddit's community response system (karma), we also show that discussions and users with higher community endorsement are more likely to match the language style of the community, where the language model that best classifies the community is also most correlated with community response. In addition, online users tend to have more positive community response when they specialize in fewer subreddits. These results have implications for detecting newcomers in a community and the popularity of posts, as well as for language generation.

Acknowledgments

This paper is based on work supported by the DARPA DEFT Program. Views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We thank the reviewers for their helpful feedback.

References

Amittai Axelrod. 2014. *Data Selection for Statistical Machine Translation*. Ph.D. thesis, University of Washington, Seattle.

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 327–337. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialog. In *Proceedings of the ACL Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words! Linguistic style accommodation in social media. In *Proceedings of WWW*.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of WWW*.

Sergey Feldman, Alex Marin, Mari Ostendorf, and Maya Gupta. 2009. Part-of-speech histogram features for genre classification of text. In *Proc. ICASSP*, pages 4781–4784.

Jorge E. Hirsch. 2005. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569–16572.

Rukmini Iyer and Mari Ostendorf. 1999. Relevance weighting for combining multi-domain data for n-gram language modeling. *Comput. Speech Lang.*, 13(3):267–282, July.

Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Talking to the crowd: What do people react to in online discussions? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2026–2031, Lisbon, Portugal, September. Association for Computational Linguistics.

Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. 2013. What's in a name? Understanding the interplay between titles, content, and communities in social media. In *International AAAI Conference on Web and Social Media*.

Rivka Levitan, Agustín Gravano, and Julia Hirschberg. 2011. Entrainment in speech preceding backchannels. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 113–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Dong Nguyen and Carolyn P. Rosé. 2011. Language use as a reflection of socialization in online communities. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 76–85. Association for Computational Linguistics.

Kate Niederhoffer and James Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21:337–360.

Philipp Petrenz and Bonnie Webber. 2011. Stable classification of text genres. *Computational Linguistics*, 37(2):385–393.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: Tracing stylometric evidence beyond topic and genre. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 78–86. Association for Computational Linguistics.

Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. 2000. Text genre detection using common word frequencies. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 808–814. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286.

Chenhao Tan and Lillian Lee. 2015. All who wander: On the prevalence and characteristics of multi-community engagement. In *Proceedings of WWW*.

Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts

Masashi Yoshikawa and **Hiroyuki Shindo** and **Yuji Matsumoto**
Graduate School of Information and Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, 630-0192, Japan
{ masashi.yoshikawa.yh8, shindo, matsu }@is.naist.jp

Abstract

Joint dependency parsing with disfluency detection is an important task in speech language processing. Recent methods show high performance for this task, although most authors make the unrealistic assumption that input texts are transcribed by human annotators. In real-world applications, the input text is typically the output of an automatic speech recognition (ASR) system, which implies that the text contains not only disfluency noises but also recognition errors from the ASR system. In this work, we propose a parsing method that handles both disfluency and ASR errors using an incremental shift-reduce algorithm with several novel features suited to ASR output texts. Because the gold dependency information is usually annotated only on transcribed texts, we also introduce an alignment-based method for transferring the gold dependency annotation to the ASR output texts to construct training data for our parser. We conducted an experiment on the Switchboard corpus and show that our method outperforms conventional methods in terms of dependency parsing and disfluency detection.

1 Introduction

Spontaneous speech is different from written text in many ways, one of which is that it contains disfluencies, that is, parts of the utterance that are corrected by the speaker during the utterance. NLP system performance is reported to deteriorate when there are disfluencies, for example, with SMT (Cho et al., 2014). Therefore, it is desirable to preprocess the speech before passing it to other NLP tasks.

There are a number of studies that address the problem of detecting disfluencies. Some of these studies include dependency parsing (Honnibal and Johnson, 2014; Wu et al., 2015; Rasooli and Tetreault, 2014), whereas others are dedicated systems (Qian and Liu, 2013; Ferguson et al., 2015; Hough and Purver, 2014; Hough and Schlangen, 2015; Liu et al., 2003). Among these studies, Honnibal (2014) and Wu (2015) address this problem by adding a new action to transition-based dependency parsing that removes the disfluent parts of the input sentence from the stack. Using this approach, they achieved high performance in terms of both dependency parsing and disfluency detection on the Switchboard corpus.

However, the authors assume that the input texts to parse are transcribed by human annotators, which, in practice, is unrealistic. In real-world applications, in addition to disfluencies, the input texts contain ASR errors; these issues might degrade the parsing performance. For example, proper nouns that are not contained in the ASR system vocabulary may break up into smaller pieces, yielding a difficult problem for the parsing unit (Cheng et al., 2015):

REF: what can we get at **Litanfeeth**
HYP: what can we get *it leaks on feet*

In this work, we propose a method for joint dependency parsing and disfluency detection that can robustly parse ASR output texts. Our parser handles both disfluencies and ASR errors using an incremental shift-reduce algorithm, with novel features that consider recognition errors of the ASR system.

Furthermore, to evaluate dependency parsing per-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1036–1041,
Austin, Texas, November 1-5, 2016.

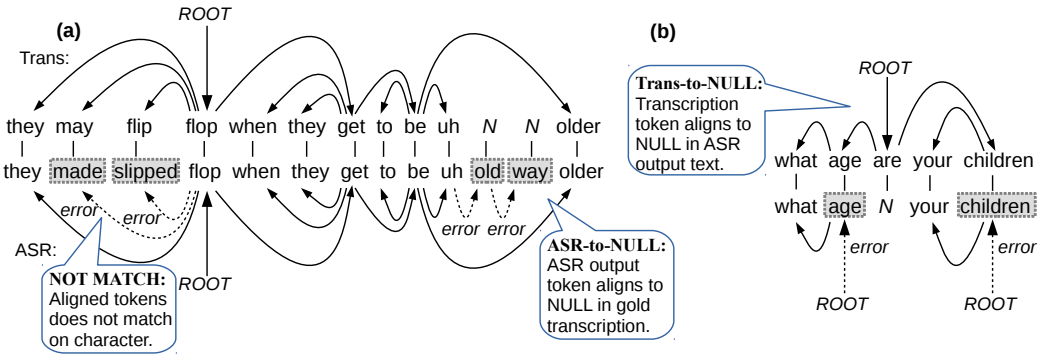


Figure 1: Examples of three problematic cases. Above shows the gold transcription and its tree, below shows the aligned ASR output and its newly transferred tree, where the dotted edges are ASR error edges.

formance on real human utterances, we create a tree-annotated corpus that contains ASR errors. [1]

2 Data Creation

To evaluate dependency parsing performance on real speech texts, we must create a tree-annotated corpus of ASR output texts.

Given a corpus that consists of speech data, transcription text and its syntactic annotation (e.g., the Switchboard corpus), we first apply the ASR system to the speech data. Next, we perform alignment between the ASR output texts and the transcription. Then, we transfer the gold syntactic annotation to the ASR output texts based on this alignment (Figure 1). The alignment is performed by minimizing the edit distance between the two sentences. We include "NULL" tokens in this alignment to allow for some tokens not having an aligned conterpart ("*N*" tokens in the Figure 1).

In the constructed trees, there are three problematic cases based on how an ASR output text and its transcription are aligned with each other: (1) a word in the ASR output text aligns with a NULL token in the transcription (**ASR-to-NULL**), (2) a word in the gold transcription aligns with a NULL in the ASR output (**Trans-to-NULL**), and (3) two words align, but do not match exactly in terms of characters (**NOT MATCH**). To create a consistent dependency tree that spans the entire sentence, we must address each of these cases.

[1]There are also studies that tackle the problem of disfluency detection in the context of speech recognition such as (Liu et al., 2003). Our work is novel in that our aim is to extend the joint method of disfluency detection with dependency parsing so that it can be applicable to the output of ASR system.

2.1 ASR-to-NULL

In the case of ASR-to-NULL, a token from the ASR system has no corresponding token in the gold transcription. In this case, we automatically annotate a dependency relation with an "error" label such that the token's head becomes the previous word token.

Figure 1(a) shows an example of this case. In the figure, the words "old" and "way" have no corresponding words in the gold transcription. Thus, we automatically annotate the dependency relations between ("old", "uh") and ("way", "old"), respectively, with the "error" label.

2.2 Trans-to-NULL

Although NULL tokens are introduced to facilitate alignment, as these tokens in the ASR output are not actual words, we must remove them in the final tree. Without any treatment, the gold transcription tokens aligned to these tokens are also deleted along with them. This causes the child tokens in the sentence not to have heads; consequently, these child tokens are not included in the syntactic tree. To avoid this problem, we instead attach them to the head of the deleted token.

For example, in Figure 1(b), the word "are" is missing in the ASR hypothesis. Then, this token's children lose their head in the transfer process. Thus, we rescue these children by attaching them to the head of "are", which, in this case, is ROOT token.

If the head of the removed token is also of the Trans-to-NULL type, then we look for an alternative head by climbing the tree in a recursive manner, until reaching ROOT. We also label the newly created edges in this process as "error".

2.3 NOT MATCH

In cases in which two aligned tokens do not match exactly on the character level, the mismatch is regarded as an instance of a substitution type of ASR error. Therefore, we encode this fact in the label of the arc from the token to its head.

In Figure 1(a), the words "made" and "slipped" in the ASR hypothesis do not match the gold transcription tokens, "may" and "flip", respectively. Therefore, we automatically re-label the arc from each token to its head as "error".

3 Transition-based Dependency Parsing

To parse texts that contain disfluencies and ASR errors, we extend the ArcEager shift-reduce dependency parser of (Zhang and Nivre, 2011). Our proposed parser adopts the same *Shift*, *Reduce*, *LeftArc*, and *RightArc* actions as ArcEager. To this parser we add three new actions, i.e., *Edit*, *LeftArcError*, and *RightArcError*, to handle disfluencies and ASR errors.

Edit action removes a disfluent token when it is the first element of the stack. This is different from Honnibal (2014)'s *Edit* action: theirs accumulates consecutive disfluent tokens on the top of the stack and removes them all at once, whereas our method removes this kind of token one-by-one. Use of this *Edit* action guarantees that the length of the action sequence is always $2n-1$. This property is advantageous because the parser can use the standard beam search and does not require normalization, such as those adopted in (Honnibal and Johnson, 2014) and (Zhu et al., 2013).

LeftArcError and *RightArcError* act in the same way as *LeftArc* and *RightArc*, except that these act only on ASR error tokens, whereas the original *LeftArc* and *RightArc* are reserved for non ASR error tokens. Using two different kinds of *Arc* actions for the two types of tokens (ASR error or not) allows for the weights not to be shared between them, and is expected to yield improved performance.

In the experiment below, we train all of the models using structured perceptron with max violation (Huang et al., 2012). The feature set is mainly based on (Honnibal and Johnson, 2014), such as the disfluency capturing features to inquire whether the token sequence inside the two specific spans match on word forms or POS tags. We adjusted these features to inspect the content of the buffer more carefully, because our parser decides if the word token is disfluent or not every time new token is shifted and hints for the decision lies much more in the buffer.

3.1 Backoff Action Feature

With the newly proposed *LeftArcError* and *RightArcError* actions, we fear that the relatively low frequency of "error" tokens may cause the weights for these actions to be updated too infrequently to be accurately generalized. We resort to using the "backoff action feature" to avoid this situation. This means that, for each action $a \in \{LeftArc, LeftArcError\}$, the score of performing it in a state s is calculated as follow:

$$SCORE(a, s) = \mathbf{w} \cdot \mathbf{f}(a, s) + \mathbf{w} \cdot \mathbf{f}(a', s) \quad (1)$$

where a' = *LeftArcBackoff*, $\mathbf{w}$ is the weight vector and $\mathbf{f}(\cdot, \cdot)$ is the feature representation, respectively. *LeftArcBackoff* is not actual action performed by our parser, rather it is used to provide the common feature representation which both *LeftArc* and *LeftArcError* can "back off" to. *RightArc* and *RightArcError* actions also calculate their scores as in Eq.(1), with a' = *RightArcBackoff*. The scores for all the other actions are calculated in the normal way: $SCORE(a, s) = \mathbf{w} \cdot \mathbf{f}(a, s)$.

3.2 WCN Feature

To better capture which parts of the texts are likely to be ASR errors, we use additional features extracted from a word confusion network (WCN) generated by ASR models. Marin (2015) reports his observation that WCN slots with more arcs tend to correspond to erroneous region. Following (Marin, 2015), we use mean and standard deviation of arc posteriors and the highest arc posterior in each WCN slot corresponding to each word token. We include in the feature vector these real-valued features for tokens on top of the stack and the first three elements of the buffer.

4 Experiment

We conducted experiments using both the proposed parsing method and the tree-annotated corpus based on the ASR output texts. Our experiments were performed using the Switchboard corpus (Godfrey et

al., 1992). This corpus consists of speech data and its transcription texts, and subset of which is annotated with POS tags, syntactic trees and disfluency information (repair, discourse marker and interjection) based on (Shriberg, 1994). [2]

4.1 ASR Settings

To obtain the ASR output texts of the corpus, we used the off-the-shelf NeuralNet recipe (Zhang et al., 2014) presented by Kaldi.[3] We used the jackknife method to obtain the ASR output texts throughout the syntactically annotated part of the corpus. [4]

From these ASR output texts, we created the tree-annotated corpus by applying the data creation method introduced in §2. Out of all 857,493 word tokens, there are 32,606 ASR-to-NULL, 34,952 Trans-to-NULL, and 93,138 NOT MATCH cases, meaning 15.6% of all word tokens had "error" labeled arcs.

4.2 Parsing Settings

We assigned POS tags to the created corpus using the Stanford POS tagger (Toutanova et al., 2003) trained on a part of the gold Switchboard corpus. [5]

We adopt the same train/dev/test split as in (Honnibal and Johnson, 2014), although the data size reduces slightly during the process of data creation. We report the unlabeled attachment score (UAS), which indicates how many heads of fluent tokens are correctly predicted. As for disfluency detection, we report precision/recall/F1-score values following the previous work in the literature.

As a baseline (To which we refer as *Base* in the following), we use an ArcEager parser with our proposed *Edit* action and the disfluency capturing features, trained on the train part of the gold Switchboard corpus. Using this parser on ASR output test data can be seen as reproducing the typical situation, in which a parser is trained on ASR-error-free texts, but nevertheless needs to parse the ASR output texts.

[2]We converted the phrase structure trees to dependency ones using the Stanford converter (de Marneffe et al., 2006).

[3]http://kaldi-asr.org/

[4]The average Word Error Rate of resulting models were 13.9 % on the Switchboard part of HUB5 evaluation dataset: https://catalog.ldc.upenn.edu/LDC2002S09

[5]We used a part of the corpus that is annotated with POS information but not syntactic one. The performance of the tagger is evaluated on the syntactically annotated part of the corpus; the tagger has an accuracy score of 95.0%.

Model	Dep	Disfl		
	UAS	Prec.	Rec.	F1
Base	72.7	58.6	**62.2**	60.3
+ *ErrorAct*	76.3	66.0	57.6	61.5
+ *Backoff*	**76.4**	65.6	57.3	61.1
+ *WCN*	76.2	**67.9**	57.9	**62.5**

Table 1: Dependency parsing and disfluency detection results of the proposed methods. We used our created corpus as both train and test data.

Train	Test	Model	Dep	Disfl		
			UAS	Prec.	Rec.	F1
$Trans$	$Trans$	*Base*	89.7	90.4	76.8	83.1
$Trans$	ASR	*Base*	74.7	58.5	**65.6**	61.8
ASR	ASR	*Base*	72.7	58.6	62.2	60.3
ASR	ASR	*Ours*	**76.2**	**67.9**	57.9	**62.5**

Table 2: Parsing result on different train-test settings. $Trans$ refers to original Switchboard transcription text, ASR the text created through the data creation in §4.1. *Ours* is our proposed parser: *Base* + *ErrorAct* + *Backoff* + *WCN*.

4.3 Results and Analysis

In Table 1, based on the baseline *Base* parser, we report scores with the additional (and additive) use of *Left/RightArcError* actions (*ErrorAct*), the WCN feature (*WCN*), and the backoff action feature (*Backoff*), on our created corpus. Using *ErrorAct* resulted in 3.6% and 1.2% improvement in UAS and disfluency detection F1, respectively. *Backoff* contributes to further improved UAS, whereas *WCN* cause an increase in disfluency detection accuracy.

Table 2 reports performance on various train and test data settings. In Table 2, the Train and Test columns represent which data to use in training and testing; $Trans$ refers to the gold transcription text of the Switchboard corpus, and ASR the text created through the data creation in §4.1. When evaluated on the ASR texts, the parser trained on the ASR texts showed degraded performance compared to the parser trained on the gold transcription ($(\text{Train}, \text{Test}) = (ASR, ASR)$). Although both the train and test data are ASR texts and share characteristics, we did not observe domain adaptation effect. We hypothesized that the drop in the performance is due to the noisy nature of our corpus, which is created from the texts with ASR errors. Having ASR-

error-specific actions, *Left/RightArcError* mitigates this problem by separately treating the ASR error tokens and non ASR error tokens. Finally, with the newly proposed features, the parser trained on ASR texts outperforms the parser trained on the transcription texts with the improvement of 1.5% and 0.7% for UAS and disfluency detection, respectively.

However, when compared with the case of $(\text{Train}, \text{Test}) = (Trans, Trans)$, we observe significant decreases in performance in both of the tasks conducted on ASR texts. This result clearly poses a new challenge for the disfluency detection community.

5 Conclusion

In this work, we have proposed a novel joint transition-based dependency parsing method with disfluency detection. Using new actions, and new feature set, the proposed parser can parse ASR output texts robustly. We have also introduced a data construction method to evaluate dependency parsing and disfluency detection performance for real speech data. As the experimental results for ASR texts is significantly lower than that achieved for the gold transcription texts, we have clarified the need to develop a method that is robust to recognition errors in the ASR system.

6 Acknowledgements

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of this paper. This work was supported by JSPS KAKENHI Grant Number 15K16053, 26240035.

References

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746. Association for Computational Linguistics.

Eunah Cho, Jan Niehues, and Alex Waibel. 2014. Tight integration of speech disfluency removal into smt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers (EACL)*, pages 43–47. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 257–262. Association for Computational Linguistics.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. " switchboard: Telephone speech corpus for research and development ". In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on (Volume:1)*. Proc. IEEE Int. Conf. Acoust. Speech Sig. Proc.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. In *Transactions of the Association of Computational Linguistics Volume 2, Issue 1 (TACL)*, pages 131–142. Association for Computational Linguistics.

Julian Hough and Matthew Purver. 2014. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89. Association for Computational Linguistics.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. Interspeech 2015.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Yang Liu, Elizabeth Shriberg, and Andreas Stolcke. 2003. Automatic disfluency identification in coversational speech using multiple knowledge sources. In *In Proceedings of the 8th Eurospeech Conference*.

Marius Alexandru Marin. 2015. In *Effective Use of Cross-Domain Parsing in Automatic Speech Recognition and Error Detection. Ph.D. thesis*. University of Washington.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel Tetreault. 2014. Non-monotonic parsing of fluent umm i mean disfluent sentences. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers (EACL)*,

pages 48–53. Association for Computational Linguistics.

Elizabeth Shriberg. 1994. In *Preliminaries to a Theory of Speech Disfluencies. Ph.D. thesis*. University of California, Berkeley.

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

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (ACL)*, pages 495–503. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 188–193. Association for Computational Linguistics.

Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2014. Improving deep neural network acoustic models using generalized maxout networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Miu Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 434–443. Association for Computational Linguistics.

Real-Time Speech Emotion and Sentiment Recognition for Interactive Dialogue Systems

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu,
Yan Wan, Ricky Ho Yin Chan and **Pascale Fung**
Human Language Technology Center
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[dbertero, fsiddique]@connect.ust.hk, b01901045@ntu.edu.tw,
ywanad@connect.ust.hk, eehychan@ust.hk, pascale@ece.ust.hk

Abstract

In this paper, we describe our approach of enabling an interactive dialogue system to recognize user emotion and sentiment in real-time. These modules allow otherwise conventional dialogue systems to have "empathy" and answer to the user while being aware of their emotion and intent. Emotion recognition from speech previously consists of feature engineering and machine learning where the first stage causes delay in decoding time. We describe a CNN model to extract emotion from raw speech input without feature engineering. This approach even achieves an impressive average of 65.7% accuracy on six emotion categories, a 4.5% improvement when compared to the conventional feature based SVM classification. A separate, CNN-based sentiment analysis module recognizes sentiments from speech recognition results, with 82.5 F-measure on human-machine dialogues when trained with out-of-domain data.

1 Introduction

Interactive dialogue systems and chatbots have been around for a while. Some, though not all, systems have statistical and machine learning modules to enable them to improve overtime. With the pervasiveness of such systems on mobile devices, expectations of user experience have also increased. We expect human-machine dialogues to get closer to human-human dialogues. One important factor is that we expect machines to understand our emotions and intent and respond with empathy.

We propose a module of emotion and sentiment recognition for an interactive dialogue system. This module enables the system to assess the user's current emotional state and sentiment, and thereby decide the appropriate response at every dialogue state. The dialogue management system handles the mixed-initiative dialogues while taking into account user emotion and sentiment, in addition to query content. Emotion and sentiment recognition enables our system to handle user queries previously unseen in training data. Positive user queries containing positive emotion and sentiment label would have a positive response, and similarly a negatively labeled statement would have a negative response. Examples are shown below:

User: *I lost my job.*
Response: *Sorry to hear that. Success is in never giving up.*
User: *I just graduated from college!*
Response: *Congratulations! I am happy for you.*
User: *I went on a vacation last month and it was pretty bad, I lost all my luggage*
Response: *That doesn't sound so good. Hope your next vacation will be a good one.*
User: *My last vacation was amazing, I loved it!*
Response: *That sounds great. I would like to travel with you.*

Meanwhile, dialogue systems like this need to have real-time recognition of user emotion and sentiment. Previous approaches of emotion recognition from speech involve feature engineering (Schuller et al., 2009; Schuller et al., 2010) as a first step which invariably causes delay in decoding. So we are interested in investigating a method to avoid feature engineering and instead use a Convolutional Neural

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1042–1047,
Austin, Texas, November 1-5, 2016.

Network to extract emotion from raw audio input directly.

2 Speech Recognition

Our acoustic data is obtained from various public domain corpora and LDC corpora, comprised of 1385hrs of speech. We use Kaldi speech recognition toolkit (Povey et al., 2011) to train our acoustic models. We train deep neural network hidden Markov models (DNN-HMMs) using the raw audio together with encode-decode parallel audio. We apply layer-wise training of restricted Boltzmann machines (RBM) (Hinton, 2010), frame cross-entropy training with mini-batch stochastic gradient descent (SGD) and sequence discriminative training using state Minimum Bayes Risk (sMBR) criterion.

The text data, of approximately 90 million sentences, includes acoustic training transcriptions, filtered sentences of Google 1 billion word LM benchmark (Chelba et al., 2013), and other multiple domains (web news, music, weather). Our decoder allows streaming of raw audio or CELP encoded data through TCP/IP or HTTP protocol, and performs decoding in real time. The ASR system achieves 7.6% word error rate on our clean speech test data[1].

3 Real-Time Emotion Recognition from Time-Domain Raw Audio Input

In recent years, we have seen successful systems that gave high classification accuracies on benchmark datasets of emotional speech (Mairesse et al., 2007) or music genres and moods (Schermerhorn and Scheutz, 2011).

Most of such work consists of two main steps, namely feature extraction and classifier learning, which is tedious and time-consuming. Extracting high and low level features (Schuller et al., 2009), and computing over windows of audio signals typically takes a few dozen seconds to do for each utterance, making the response time less than real-time instantaneous, which users have come to expect from interactive systems. It also requires a lot of hand tuning. In order to bypass feature engineering, the current direction is to explore methods that can recognize emotion or mood directly from time-domain audio signals. One approach that has shown great potential is using Convolutional Neural Networks. In the following sections, we compare an approach of using CNN without feature engineering to a method that uses audio features with a SVM classifier.

3.1 Dataset

For our experiments on emotion recognition with raw audio, we built a dataset from the TED-LIUM corpus release 2 (Rousseau et al., 2014). It includes 207 hours of speech extracted from 1495 TED talks. We annotated the data with an existing commercial API followed by manual correction. We use these 6 categories: criticism, anxiety, anger, loneliness, happiness, and sadness. We obtained a total of 2389 segments for the criticism category, 3855 for anxiety, 12708 for anger, 3618 for loneliness, 8070 for happy and 1824 for sadness. The segments have an average length slightly above 13 seconds.

3.2 Convolutional Neural Network model

The Convolutional Neural Network (CNN) model using raw audio as input is shown in Figure 1. The raw audio samples are first down-sampled at 8 kHz, in order to optimize between the sampling rate and representation memory efficiency in case of longer segments. The CNN is designed with a single filter for real-time processing. We set a convolution window of size 200, which corresponds to 25 ms, and an overlapping step size of 50, equal to around 6 ms. The convolution layer performs the feature extraction, and models the variations among neighboring, overlapping frames. The subsequent max-pooling combines the contributions of all the frames, and gives as output a segment-based vector. This is then fed into a fully connected layer before the final softmax layer. These last layers perform a similar function as those of a fully connected Deep Neural Network (DNN), mapping the max-pooling output into a probabilistic distribution over the desired emotional output categories.

During decoding the processing time increases linearly with the length of the audio input segment. Thus the largest time contribution is due to the computations inside the network (He and Sun, 2015), which with a single convolution layer can be performed in negligible time for single utterances.

[1] https://catalog.ldc.upenn.edu/LDC94S13A

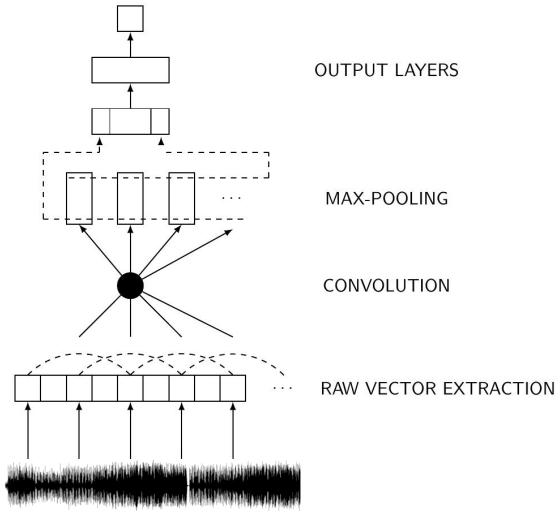


Figure 1: Convolutional Neural Network model for emotion classification from raw audio samples.

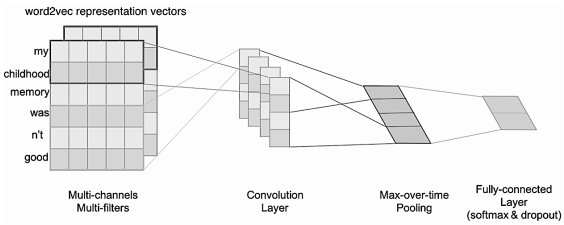


Figure 2: Convolutional neural network model for sentiment classification

4 Sentiment Inference from Speech and Text

Convolutional Neural Networks (CNNs) have recently achieved remarkably strong performance also on the practically important task of sentence classification (Johnson and Zhang, 2014; Kalchbrenner et al., 2014; Kim, 2014). In our approach, we use a CNN-based classifier with Word2Vec to analyze the sentiment of recognized speech.

We train a CNN with one layer of convolution and max pooling (Collobert et al., 2011) on top of word embedding vectors trained on the Google News corpus (Mikolov et al., 2013) of size 300. We apply on top of the word vectors a convolutional sliding window of size 3, 4 and 5 to represent multiple features. We then apply a max-pooling operation over the output vectors of the convolutional layer, that allows the model to pick up the most valuable information wherever it happens in the input sentence, and give as output a fixed-length sentence encoding vector.

Emotion class	SVM	CNN
Criticism/Cynicism	55.0	61.2
Defensiveness/Anxiety	56.3	62.0
Hostility/Anger	72.8	72.9
Loneliness/Unfulfillment	61.1	66.6
Love/Happiness	50.9	60.1
Sadness/Sorrow	71.1	71.4
Average	61.2	**65.7**

Table 1: Accuracy obtained, percentage, in the Convolutional Neural Network model for emotion classification from raw audio samples.

We employ two distinct CNN channels: the first uses word embedding vectors directly as input, while the second fine-tunes them via back propagation (Kim, 2014). All the hidden layer dimensions are set to 100. The final softmax layer takes as input the concatenated sentence encoding vectors of the two channels, and gives as output is the probability distribution over a binary classification for sentiment analysis of text transcribed from speech by our speech recognizer.

To improve the performance of sentiment classification in real time conversation, we compare the performance on the Movie Review dataset used in Kim (2014) with the Twitter sentiment 140[2] dataset. This twitter dataset contains a total of 1.6M sentences with positive and negative sentiment labels. Before training the CNN model we apply some preprocessing as mentioned in Go et al. (2009).

5 Experiments

5.1 Experimental setup

For the speech emotion detection module we setup our experiments as binary classification tasks, in which each segment is classified as either part of a particular emotion category or not. For each category the negative samples were chosen randomly from the clips that did not belong to the positive category. We took 80% of the data as training set, and 10% each as development and test set. The development set was used to tune the hyperparameters and determine the early stopping condition. We implemented our CNN with the THEANO framework

[2] www.sentiment140.com

Corpus	Average Length	Size	Vocabulary Size	Words in Word2vec
Movie Review	20	10662	18765	16448
Twitter	12.97	1600000	273761	79663

Table 2: Corpus statistics for text sentiment experiments with CNN.

Model	Accuracy	Precision	Recall	F-score
CNN model (trained on Movie Review dataset)	67.8%	91.2%	63.5%	74.8
LIWC (keyword based)	**73.5%**	**80.3%**	77.3%	77.7
CNN model (trained on Twitter dataset)	72.17%	78.64%	**86.69%**	**82.5**

Table 3: Sentiment analysis result on human-machine dialogue when trained from Twitter and Movie Review dataset

(Bergstra et al., 2010). We chose rectified linear as the non-linear function for the hidden layers, as it generally provided better performance over other functions. We used standard backpropagation training, with momentum set to 0.9 and initial learning rate to 10^{-5}. As a baseline we used a linear-kernel SVM model from the LibSVM (Chang and Lin, 2011) library with the INTERSPEECH 2009 emotion feature set (Schuller et al., 2009), extracted with openSMILE (Eyben et al., 2010). These features are computed from a series of input frames and output a single static summary vector, e.g, the smooth methods, maximum and minimum value, mean value of the features from the frames (Liscombe et al., 2003).

A similar one-layer CNN setup was used also for the sentiment module, again with rectified linear as the activation function. As our dataset contains many neutral samples, we trained two distinct CNNs: one for positive sentiment and one for negative, and showed the average results among the two categories. For each of the two training corpora we took 10% as development set. We used as baseline a method that uses positive and emotion keywords from the Linguistic Inquiry and Word Count (LIWC 2015) dictionary (Pennebaker et al., 2015).

5.2 Results and discussion

5.2.1 Speech emotion recognition

Results obtained by this module are shown in Table 1. In all the emotion classes considered our CNN model outperformed the SVM baseline, sometimes marginally (in the angry and sad classes), sometimes more significantly (happy and criticism classes). It is particularly important to point out that our CNN does not use any kind of preprocessed features. The lower results for some categories, even on the SVM baseline, may be a sign of inaccuracy in manual labeling. We plan to work to improve both the dataset, with hand-labeled samples, and periodically retrain the model as ongoing work.

Processing time is another key factor of our system. We ran an evaluation of the time needed to perform all the operations required by our system (down-sampling, audio samples extraction and classification) on a commercial laptop. The system we used is a Lenovo x250 laptop with a Intel i5 CPU, 8 Gb RAM, an SSD hard disk and running Linux Ubuntu 16.04. Our classifier took an average of 162 ms over 10 segments randomly chosen from our corpus of length greater than 13 s, which corresponds to 13 ms per second of speech, hence achieving real-time performance on typical utterances. The key of the low processing time is the lightweight structure of the CNN, which uses only one filter. We replicated the evaluations with the same 10 segments on a two-filter CNN, where the second filter spans over 250 ms windows. Although we obtained higher performance with this structure in our preliminary experiments, the processing time raised to 6.067 s, which corresponds to around 500 ms per second of speech. This is over one order of magnitude higher than the one filter configuration, making it less suitable for time constrained applications such as dialogue systems.

5.2.2 Sentiment inference from ASR

Results obtained by this module are shown in Table 3. Our CNN model got a 6.1% relative improvement on F-score over the baseline when trained with the larger Twitter dataset. The keyword based method got a slightly better accuracy and precision and a much lower recall on our relatively small human-machine dialogue dataset (821 short utter-

ances). However, we noticed that the keyword based method accuracy fell sharply when tested on the larger Twitter dataset we used to train the CNN, yielding only 45% accuracy. We also expect to improve our CNN model in the future training it with more domain specific data, something not possible with a thesaurus based method.

6 Conclusion

In this paper, we have introduced the emotion and sentiment recognition module for an interactive dialog system. We described in detail the two parts involved, namely speech emotion and sentiment recognition, and discussed the results achieved. We have shown how deep learning can be used for such modules in this architecture, ranging from speech recognition, emotion recognition to sentiment recognition from dialogue. More importantly, we have shown that by using a CNN with a single filter, it is possible to obtain real-time performance on speech emotion recognition at 65.7% accuracy, directly from time-domain audio input, bypassing feature engineering. Sentiment analysis with CNN also leads to a 82.5 F-measure when trained from out-of-domain data. This approach of creating emotionally intelligent systems will help future robots to acquire empathy, and therefore rather than committing harm, they can act as friends and caregivers to humans.

Acknowledgments

This work was partially funded by the Hong Kong Phd Fellowship Scheme, and partially by grant #16214415 of the Hong Kong Research Grants Council.

References

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

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

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

Kaiming He and Jian Sun. 2015. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360.

Geoffrey Hinton. 2010. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jackson Liscombe, Jennifer Venditti, and Julia Bell Hirschberg. 2003. Classifying subject ratings of emotional speech using acoustic features. In *Proceedings of Eurospeech*, pages 725–728. ISCA.

François Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*, pages 457–500.

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

J.W. Pennebaker, R.J. Booth, R.L. Boyd, and M.E. Francis. 2015. Linguistic inquiry and word count: Liwc2015. *Austin, TX: Pennebaker Conglomerates*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In

IEEE 2011 workshop on automatic speech recognition and understanding, number EPFL-CONF-192584. IEEE Signal Processing Society.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. 2014. Enhancing the ted-lium corpus with selected data for language modeling and more ted talks. In *LREC*, pages 3935–3939.

Paul Schermerhorn and Matthias Scheutz. 2011. Disentangling the effects of robot affect, embodiment, and autonomy on human team members in a mixed-initiative task. In *Proceedings from the International Conference on Advances in Computer-Human Interactions*, pages 236–241.

Björn Schuller, Stefan Steidl, and Anton Batliner. 2009. The interspeech 2009 emotion challenge. In *INTERSPEECH*, volume 2009, pages 312–315. Citeseer.

Björn Schuller, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian A Müller, and Shrikanth S Narayanan. 2010. The interspeech 2010 paralinguistic challenge. In *INTERSPEECH*, volume 2010, pages 2795–2798.

A Neural Network Architecture for Multilingual Punctuation Generation

Miguel Ballesteros[1] **Leo Wanner**[1,2]
[1]NLP Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Catalan Institute for Research and Advanced Studies (ICREA)
miguel.ballesteros@upf.edu leo.wanner@upf.edu

Abstract

Even syntactically correct sentences are perceived as awkward if they do not contain correct punctuation. Still, the problem of automatic generation of punctuation marks has been largely neglected for a long time. We present a novel model that introduces punctuation marks into raw text material with transition-based algorithm using LSTMs. Unlike the state-of-the-art approaches, our model is language-independent and also neutral with respect to the intended use of the punctuation. Multilingual experiments show that it achieves high accuracy on the full range of punctuation marks across languages.

1 Introduction

Although omnipresent in (language learner) grammar books, punctuation received much less attention in linguistics and natural language processing (Krahn, 2014). In linguistics, punctuation is generally acknowledged to possess different functions. Its traditionally most studied function is that to encode prosody of oral speech, i.e., the prosodic *rhetorical* function; see, e.g., (Kirchhoff and Primus, 2014) and the references therein. In particular the comma is assumed to possess a strong rhetorical function (Nunberg et al., 2002). Its other functions are the grammatical function, which leads it to form a separate (along with semantics, syntax, and phonology) grammatical submodule (Nunberg, 1990), and the syntactic function (Quirk et al., 1972), which makes it reflect the syntactic structure of a sentence.

The different functions of punctuation are also reflected in different tasks in natural language processing (NLP): introduction of punctuation marks into a generated sentence that is to be read aloud, restoration of punctuation in speech transcripts, parsing under consideration of punctuation, or generation of punctuation in written discourse. Our work is centered in the last task. We present a novel punctuation generation algorithm that is based on the transition-based algorithm with long short-term memories (LSTMs) by Dyer et al. (2015) and character-based continuous-space vector embeddings of words using bidirectional LSTMs (Ling et al., 2015b; Ballesteros et al., 2015). The algorithm takes as input raw material without punctuation and effectively introduces the full range of punctuation symbols. Although intended, first of all, for use in sentence generation, the algorithm is function- and language-neutral, which makes it different, compared to most of the state-of-the-art approaches, which use function- and/or language-specific features.

2 Related Work

The most prominent punctuation-related NLP task has been so far introduction (or restoration) of punctuation in speech transcripts. Most often, classifier models are used that are trained on n-gram models (Gravano et al., 2009), on n-gram models enriched by syntactic and lexical features (Ueffing et al., 2013) and/or by acoustic features (Baron et al., 2002; Kolář and Lamel, 2012). Tilk and Alumäe (2015) use a lexical and acoustic (pause duration) feature-based LSTM model for the restoration of periods and commas in Estonian speech transcripts. The grammatical and syntactic functions of punctuation have been addressed in the context of written

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1048–1053,
Austin, Texas, November 1-5, 2016.

language. Some of the proposals focus on the grammatical function (Doran, 1998; White and Rajkumar, 2008), while others bring the grammatical and syntactic functions together and design rule-based grammatical resources for parsing (Briscoe, 1994) and surface realization (White, 1995; Guo et al., 2010). Guo et al. (2010) is one of the few works that is based on a statistical model for the generation of punctuation in the context of Chinese sentence generation, trained on a variety of syntactic features from LFG f-structures, preceding punctuation bigrams and cue words.

Our proposal is most similar to Tilk and Alumäe (2015), but our task is more complex since we generate the full range of punctuation marks. Furthermore, we do not use any acoustic features. Compared to Guo et al. (2010), we do not use any syntactic features either since our input is just raw text material.

3 Model

Our model is inspired by a number of recent works on neural architectures for structure prediction: Dyer et al. (2015)'s transition-based parsing model, Dyer et al. (2016)'s generative language model and phrase-structure parser, Ballesteros et al. (2015)'s character-based word representation for parsing, and Ling et al. (2015b)'s part-of-speech tagging .

3.1 Algorithm

We define a transition-based algorithm that introduces punctuation marks into sentences that do not contain any punctuation. In the context of NLG, the input sentence would be the result of the surface realization task (Belz et al., 2011). As in transition-based parsing (Nivre, 2004), we use two data structures: Nivre's *queue* is in our case the *input buffer* and his *stack* is in our case the *output buffer*. The algorithm starts with an input buffer full of words and an empty output buffer. The two basic actions of the algorithm are SHIFT, which moves the first word from the input buffer to the output buffer, and GENERATE, which introduces a punctuation mark after the first word in the output buffer. Figure 1 shows an example of the application of the two actions.

At each stage t of the application of the algorithm, the state, which is defined by the contents of the output and input buffers, is encoded in terms of a vector $\mathbf{s}_t$; see Section 3.3 for different alternatives of state representation. As Dyer et al. (2015), we use $\mathbf{s}_t$ to compute the probability of the action at time t as:

Transition	Output	Input
	[ ]	[No it was not]
SHIFT	[No]	[it was not]
GENERATE(",")	[No ,]	[it was not]
SHIFT	[No , it]	[was not]
SHIFT	[No , it was]	[not]
SHIFT	[No , it was not]	[ ]
GENERATE(".")	[No, it was not .]	[ ]

Figure 1: Transition sequence for the input sequence *No it was not* – with the output *No, it was not.*

$$p(z_t \mid \mathbf{s}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{s}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}} \exp\left(\mathbf{g}_{z'}^\top \mathbf{s}_t + q_{z'}\right)} \quad (1)$$

where $\mathbf{g}_z$ is a vector representing the embedding of the action z, and q_z is a bias term for action z. The set $\mathcal{A}$ represents the actions (either SHIFT or GENERATE(p)).[1] $\mathbf{s}_t$ encodes information about previous actions (since it may include the history with the actions taken and the generated punctuation symbols are introduced in the output buffer, see Section 3.3), thus the probability of a sequence of actions $\boldsymbol{z}$ given the input sequence is:

$$p(\boldsymbol{z} \mid \boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{z}|} p(z_t \mid \mathbf{s}_t). \quad (2)$$

As in (Dyer et al., 2015), the model greedily chooses the best action to take given the state with no backtracking.[2]

3.2 Word Embeddings

Following the tagging model of Ling et al. (2015b) and the parsing model of Ballesteros et al. (2015), we compute character-based continuous-space vector embeddings of words using bidirectional LSTMs (Graves and Schmidhuber, 2005) to learn similar representation for words that are similar from an orthographic/morphological point of view.

[1] Note that GENERATE(p) includes all possible punctuations that the language in question has, and thus the number of classes the classifier predicts in each time step is #punctuations + 1.

[2] For further optimization, the model could be extended, for instance, by beam-search.

The character-based representations may be also concatenated with a fixed vector representation from a neural language model. The resulting vector is passed through a component-wise rectifier linear unit (ReLU). We experiment with and without pretrained word embeddings. To pretrain the fixed vector representations, we use the skip n-gram model introduced by Ling et al. (2015a).

3.3 Representing the State

We work with two possible representations of the input and output buffers (i.e, the state $\mathbf{s}_t$): (i) a look-ahead model that takes into account the immediate context (two embeddings for the input and two embeddings for the output), which we use as a baseline, and (ii) the LSTM model, which encodes the entire input sequence and the output sentence with LSTMs.

3.3.1 Baseline: Look-ahead Model

The look-ahead model can be interpreted as a 4-gram model in which two words belong to the input and two belong to the output. The representation takes the average of the two first embeddings of the output and the two first embeddings at the front of the input. The word embeddings contain all the richness provided by the character-based LSTMs and the pretrained skip n-gram model embeddings (if used). The resulting vector is passed through a component-wise ReLU and a softmax transformation to obtain the probability distribution over the possible actions given the state $\mathbf{s}_t$; see Section 3.1.

3.3.2 LSTM Model

The baseline look-ahead model considers only the immediate context for the input and output sequences. In the proposed model, we apply recurrent neural networks (RNNs) that encode the entire input and output sequences in the form of LSTMs. LSTMs are a variant of RNNs designed to deal with the vanishing gradient problem inherent in RNNs (Hochreiter and Schmidhuber, 1997; Graves, 2013). RNNs read a vector $\mathbf{x}_t$ at each time step and compute a new (hidden) state $\mathbf{h}_t$ by applying a linear map to the concatenation of the previous time step's state $\mathbf{h}_{t-1}$ and the input, passing then the outcome through a logistic sigmoid non-linearity.

We use a simplified version of the stack LSTM model of Dyer et al. (2015). The input buffer is encoded as a stack LSTM, into which we PUSH the entire sequence at the beginning and POP words from it at each time step. The output buffer is a sequence, encoded by an LSTM, into which we PUSH the final output sequence. As in (Dyer et al., 2015), we include a third sequence with the history of actions taken, which is encoded by another LSTM. As already mentioned above, the three resulting vectors are passed through a component-wise ReLU and a softmax transformation to obtain the probability distribution over the possible actions that can be taken (either to shift or to generate a punctuation mark), given the current state $\mathbf{s}_t$; see Section 3.1.

4 Experiments

To test our models, we carried experiments on five languages: Czech, English, French, German, and Spanish. English, French and Spanish are generally assumed to be characterized by prosodic punctuation, while for German the syntactic punctuation is more dominant (Kirchhoff and Primus, 2014). Czech punctuation also leans towards syntactic punctuation (Kolář et al., 2004), but due to its rather free word order we expect it to reflect prosodic punctuation as well.

The punctuation marks that the models attempt to predict (and that also occur in the training sets) for each language are listed in Table 1.[3] Commas represent around 55% and periods around 30% of the total number of marks in the datasets.

Czech	‘.’, ‘,’, ‘–’, ‘(’, ‘)’, ‘:’, ‘/’, ‘?’, ‘%’, ‘*’, ‘=’, ‘|’, ‘”’, ‘+’, ‘;’, ‘!’, ‘o’, ‘“’, ‘&’, ‘[’, ‘]’, ‘§’
English	‘–’, ‘(’, ‘)’, ‘,’, ‘ ” ’, ‘.’, ‘…’, ‘:’, ‘;’, ‘?’, ‘ “ ’, ‘}’, ‘{’
French	‘ ” ’, ‘,’, ‘–’, ‘:’, ‘?’, ‘(’, ‘)’, ‘.’, ‘!’, ‘…’
German	‘ ” ’, ‘(’, ‘)’, ‘,’, ‘.’, ‘/’, ‘:’, ‘–’, ‘…’, ‘?’, ‘ “ ’
Spanish	‘ ” ’, ‘(’, ‘)’, ‘,’, ‘–’, ‘.’, ‘:’, ‘?’, ‘¿’, ‘!’, ‘¡’

Table 1: Punctuation marks covered in our experiments.

4.1 Setup

The stack LSTM model uses two layers, each of dimension 100 for each input sequence. For both the

[3]The consideration of some of the symbols listed in Table 1 as punctuation marks may be questioned (see, e.g., ‘+’ or ‘§’ for Czech). However, all of them are labeled as punctuation marks in the corresponding tag sets, such that we include them.

Commas															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	78.79	43.54	56.09	75.60	38.52	51.04	54.00	22.76	32.02	68.87	32.89	44.52	63.17	19.15	29.39
LookAhead + Pre	–	–	–	75.94	40.81	53.09	–	–	–	71.30	39.62	50.94	58.03	26.67	36.54
LSTM	**80.79**	**68.30**	**74.02**	78.88	70.02	74.19	**61.73**	**44.52**	**51.73**	73.78	65.45	69.37	64.01	42.73	51.25
LSTM + Pre	–	–	–	**80.83**	**74.81**	**77.70**	–	–	–	**76.56**	**69.19**	**72.69**	**65.65**	**45.33**	**53.63**
Periods															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	82.62	**95.64**	88.65	88.51	97.76	92.91	71.34	94.61	81.34	77.10	97.76	86.21	73.13	99.13	84.17
LookAhead + Pre	–	–	–	87.44	97.71	92.29	–	–	–	78.26	95.93	86.20	73.16	**99.29**	84.25
LSTM	**89.39**	93.66	**91.48**	93.07	**98.31**	95.62	**76.38**	**95.47**	**84.86**	84.75	98.18	90.97	**74.70**	98.65	**85.02**
LSTM + Pre	–	–	–	**94.44**	98.06	**96.22**	–	–	–	**85.65**	**98.39**	**91.58**	74.24	98.57	84.69
Average															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	80.90	58.57	67.95	82.72	52.72	64.40	60.67	32.33	42.18	75.82	52.58	62.10	67.50	33.88	45.12
LookAhead + Pre	–	–	–	81.83	53.90	64.99	–	–	–	75.75	54.57	63.65	64.80	38.58	48.36
LSTM	**82.42**	**69.11**	**75.18**	**84.89**	71.23	77.46	**65.34**	**45.52**	**53.66**	80.03	65.90	72.28	67.78	47.80	56.06
LSTM + Pre	–	–	–	83.72	**74.56**	**78.87**	–	–	–	**81.60**	**67.47**	**73.87**	**68.09**	**49.21**	**57.13**

Table 2: Results of the LSTM model and the Baseline (Look-ahead model) for precision, recall and F score for commas, periods and micro average for all punctuation symbols (including commas and periods) listed in Table 1. +Pre refers to models that include pretrained word embeddings.

look-ahead and the stack LSTM models, character-based embeddings, punctuation embeddings and pretrained embeddings (if used) also have 100 dimensions. Both models are trained to maximize the conditional log-likelihood (Eq. 2) of output sentences, given the input sequences.

For Czech, English, German, and Spanish, we use the wordforms from the treebanks of the CoNLL 2009 Shared Task (Hajič et al., 2009); the French dataset is by Candito et al. (2010). Development sets are used to optimize the model parameters; the results are reported for the held-out test sets.

4.2 Results and Discussion

Table 2 displays the outcome of the experiments for periods and commas in all five languages and summarizes the overall performance of our algorithm in terms of the micro-average figures. In order to test whether pretrained word embeddings provide further improvements, we incorporate them for English, Spanish and German.[4]

The figures show that the LSTMs that encode the entire context of a punctuation mark are better than a strong baseline that takes into account a 4-gram sliding window of tokens. They also show that character-based representations are already useful for the punctuation generation task on their own, but when concatenated with pretrained vectors, they are even more useful.

The model is capable of providing good results for all languages, being more consistent for English, Czech and German. Average sentence length may indicate why the model seems to be worse for Spanish and French, since sentences are longer in the Spanish (29.8) and French (27.0) datasets, compared to German (18.0), Czech (16.8) or English (24.0). The training set is also smaller in Spanish and French compared to the other languages. It is worth noting that the results across languages are not directly comparable since the datasets are different, and as shown in Table 1, the sets of punctuation marks that are to be predicted diverge significantly.

The figures in Table 2 cannot be directly compared with the figures reported by Tilk and Alumäe (2015) for their LSTM-model on period and comma restoration in speech transcripts: the tasks and datasets are different.

Our results prove that the state representation (through LSTMs, which have already been shown to be effective for syntax (Dyer et al., 2015; Dyer et al., 2016)) and character-based representations (which allow similar embeddings for words that are mor-

[4]Word embeddings for English, Spanish and German are trained using the AFP portion of the English Gigaword corpus (version 5), the German monolingual training data from the 2010 Machine Translation Workshop, and the Spanish Gigaword version 3 respectively.

phologically similar (Ling et al., 2015b; Ballesteros et al., 2015)) are capturing strong linguistic clues to predict punctuation.

5 Conclusions

We presented an LSTM-based architectured that is capable of adding punctuation marks to sequences of tokens as produced in the context of surface realization without punctuation with high quality and linear time.[5] Compared to other proposals in the field, the architecture has the advantage to operate on sequences of word forms, without any additional syntactic or acoustic features. This tool could be used for ASR (Tilk and Alumäe, 2015) and grammatical error correction (Ng et al., 2014). In the future, we plan to create cross-lingual models by applying multilingual word embeddings (Ammar et al., 2016).

Acknowledgments

This work was supported by the European Commission under the contract numbers FP7-ICT-610411 (MULTISENSOR) and H2020-RIA-645012 (KRISTINA).

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Don Baron, Elizabeth Shriberg, and Andreas Stolcke. 2002. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Proceedings of the International Conference on Spoken Language Processing*, pages 949–952, Denver, CO.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226.

Ted Briscoe. 1994. Parsing (with) punctuation. Technical report, Rank Xerox Research Centre, Grenoble, France.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical French dependency parsing: treebank conversion and first results. In *Proceedings of the LREC*.

Christine D. Doran. 1998. *Incorporating Punctuation into the Sentence Grammar*. Ph.D. thesis, University of Pennsylvania.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*.

Agustín Gravano, Martin Jansche, and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *Proceedings of the ICASSP 2009*, pages 4741–4744.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2010. A linguistically inspired statistical model for chinese punctuation generation. *ACM Transactions on Asian Language Information Processing*, 9(2).

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

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

Frank Kirchhoff and Beatrice Primus. 2014. The architecture of punctuation systems. A historical case study of the comma in German. *Written Language and Literacy*, 17(2):195–224.

Jáchym Kolář and Lori Lamel. 2012. Development and Evaluation of Automatic Punctuation for French and English Speech-to-Text. In *Proceedings of the 13th Interspeech Conference*, Portland, OR.

[5] The code is available at `https://github.com/miguelballesteros/LSTM-punctuation`

Jáchym Kolář, Jan Švec, and Josef Psutka. 2004. Automatic Punctuation Annotation in Czech Broadcast News Speech. In *Proceedings of the 9th Conference Speech and Computer*, St. Petersburg, Russia.

Albert Edward Krahn. 2014. *A New Paradigm for Punctuation*. Ph.D. thesis, University of Wisconsin-Milwaukee.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Geoffrey Nunberg, Ted Briscoe, and Rodney Huddleston. 2002. Punctuation. In *The Cambridge Grammar of the English Language*, pages 1723–1764. Cambridge University Press, Cambridge.

Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. CSLI Publications, Stanford, CA.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1972. *A Grammar of Contemporary English*. Longman, London.

Ottokar Tilk and Tanel Alumäe. 2015. LSTM for Punctuation Restoration in Speech Transcripts. In *Proceedings of the 16th Interspeech Conference*, Dresden, Germany.

Nicola Ueffing, Maximilian Bisani, and Paul Vozila. 2013. Improved models for automatic punctuation prediction for spoken and written text. In *Proceedings of the 14th Interspeech Conference*, Lyon, France.

Michael White and Rajakrishnan Rajkumar. 2008. A More Precise Analysis of Punctuation for Broad-Coverage Surface-Realization with CCG. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 17–24, Manchester, UK.

Michael White. 1995. Presenting punctuation. In *Proceedings of the 5th European Workshop on Natural Language Generation*, pages 107–125, Lyon, France.

Neural Headline Generation on Abstract Meaning Representation

Sho Takase[†] **Jun Suzuki**[‡] **Naoaki Okazaki**[†] **Tsutomu Hirao**[‡] **Masaaki Nagata**[‡]
Graduate School of Information Sciences, Tohoku University[†]
NTT Communication Science Laboratories, NTT Corporation[‡]
{takase, okazaki}@ecei.tohoku.ac.jp
{suzuki.jun, hirao.tsutomu, nagata.masaaki}@lab.ntt.co.jp

Abstract

Neural network-based encoder-decoder models are among recent attractive methodologies for tackling natural language generation tasks. This paper investigates the usefulness of structural syntactic and semantic information additionally incorporated in a baseline neural attention-based model. We encode results obtained from an abstract meaning representation (AMR) parser using a modified version of Tree-LSTM. Our proposed attention-based AMR encoder-decoder model improves headline generation benchmarks compared with the baseline neural attention-based model.

1 Introduction

Neural network-based encoder-decoder models are cutting-edge methodologies for tackling natural language generation (NLG) tasks, *i.e.*, machine translation (Cho et al., 2014), image captioning (Vinyals et al., 2015), video description (Venugopalan et al., 2015), and headline generation (Rush et al., 2015).

This paper also shares a similar goal and motivation to previous work: improving the encoder-decoder models for natural language generation. There are several directions for enhancement. This paper respects the fact that NLP researchers have expended an enormous amount of effort to develop fundamental NLP techniques such as POS tagging, dependency parsing, named entity recognition, and semantic role labeling. Intuitively, this structural, syntactic, and semantic information underlying input text has the potential for improving the quality of NLG tasks. However, to the best of our knowledge, there is no clear evidence that syntactic and semantic information can enhance the recently developed encoder-decoder models in NLG tasks.

To answer this research question, this paper proposes and evaluates a headline generation method based on an encoder-decoder architecture on *Abstract Meaning Representation (AMR)*. The method is essentially an extension of *attention-based summarization (ABS)* (Rush et al., 2015). Our proposed method encodes results obtained from an AMR parser by using a modified version of Tree-LSTM encoder (Tai et al., 2015) as additional information of the baseline ABS model. Conceptually, the reason for using AMR for headline generation is that information presented in AMR, such as predicate-argument structures and named entities, can be effective clues when producing shorter summaries (headlines) from original longer sentences. We expect that the quality of headlines will improve with this reasonable combination (ABS and AMR).

2 Attention-based summarization (ABS)

ABS proposed in Rush et al. (2015) has achieved state-of-the-art performance on the benchmark data of headline generation including the DUC-2004 dataset (Over et al., 2007). Figure 1 illustrates the model structure of ABS. The model predicts a word sequence (summary) based on the combination of the neural network language model and an input sentence encoder.

Let V be a vocabulary. $\boldsymbol{x}_i$ is the i-th indicator vector corresponding to the i-th word in the input sentence. Suppose we have M words of an input sentence. $\boldsymbol{X}$ represents an input sentence, which

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1054–1059,
Austin, Texas, November 1-5, 2016.

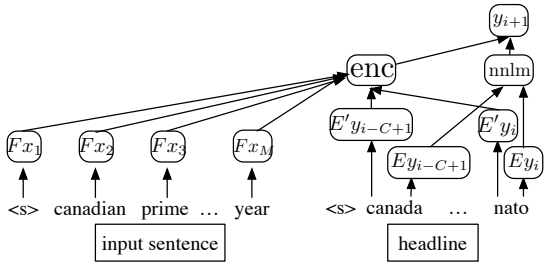


Figure 1: Model structure of 'attention-based summarization (ABS)'.

is represented as a sequence of indicator vectors, whose length is M. That is, $\boldsymbol{x}_i \in \{0,1\}^{|V|}$, and $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$. Similarly, let $\boldsymbol{Y}$ represent a sequence of indicator vectors $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L)$, whose length is L. Here, we assume $L < M$. $\boldsymbol{Y}_{C,i}$ is a short notation of the list of vectors, which consists of the sub-sequence in $\boldsymbol{Y}$ from $\boldsymbol{y}_{i-C+1}$ to $\boldsymbol{y}_i$. We assume a one-hot vector for a special start symbol, such as "$\langle$S$\rangle$", when $i < 1$. Then, ABS outputs a summary $\hat{\boldsymbol{Y}}$ given an input sentence $\boldsymbol{X}$ as follows:

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} \Big\{ \log p(\boldsymbol{Y}|\boldsymbol{X}) \Big\}, \quad (1)$$

$$\log p(\boldsymbol{Y}|\boldsymbol{X}) \approx \sum_{i=0}^{L-1} \log p(\boldsymbol{y}_{i+1}|\boldsymbol{X}, \boldsymbol{Y}_{C,i}), \quad (2)$$

$$p(\boldsymbol{y}_{i+1}|\boldsymbol{X}, \boldsymbol{Y}_{C,i}) \propto \exp\big(\mathrm{nnlm}(\boldsymbol{Y}_{C,i}) + \mathrm{enc}(\boldsymbol{X}, \boldsymbol{Y}_{C,i})\big), \quad (3)$$

where $\mathrm{nnlm}(\boldsymbol{Y}_{C,i})$ is a feed-forward neural network language model proposed in (Bengio et al., 2003), and $\mathrm{enc}(\boldsymbol{X}, \boldsymbol{Y}_{C,i})$ is an input sentence encoder with attention mechanism.

This paper uses D and H as denoting sizes (dimensions) of vectors for word embedding and hidden layer, respectively. Let $\boldsymbol{E} \in \mathbb{R}^{D\times|V|}$ be an embedding matrix of output words. Moreover, let $\boldsymbol{U} \in \mathbb{R}^{H\times(CD)}$ and $\boldsymbol{O} \in \mathbb{R}^{|V|\times H}$ be weight matrices of hidden and output layers, respectively[1]. Using the above notations, $\mathrm{nnlm}(\boldsymbol{Y}_{C,i})$ in Equation 3 can be written as follows:

$$\mathrm{nnlm}(\boldsymbol{Y}_{C,i}) = \boldsymbol{O}\boldsymbol{h}, \quad \boldsymbol{h} = \tanh(\boldsymbol{U}\tilde{\boldsymbol{y}}_c), \quad (4)$$

[1]Following Rush et al. (2015), we omit bias terms throughout the paper for readability, though each weight matrix also has a bias term.

where $\tilde{\boldsymbol{y}}_c$ is a concatenation of output embedding vectors from $i - C + 1$ to i, that is, $\tilde{\boldsymbol{y}}_c = (\boldsymbol{E}\boldsymbol{y}_{i-C+1} \cdots \boldsymbol{E}\boldsymbol{y}_i)$. Therefore, $\tilde{\boldsymbol{y}}_c$ is a (CD) dimensional vector.

Next, $\boldsymbol{F} \in \mathbb{R}^{D\times|V|}$ and $\boldsymbol{E}' \in \mathbb{R}^{D\times|V|}$ denote embedding matrices of input and output words, respectively. $\boldsymbol{O}' \in \mathbb{R}^{|V|\times D}$ is a weight matrix for the output layer. $\boldsymbol{P} \in \mathbb{R}^{D\times(CD)}$ is a weight matrix for mapping embedding of C output words onto embedding of input words. $\tilde{\boldsymbol{X}}$ is a matrix form of a list of input embeddings, namely, $\tilde{\boldsymbol{X}} = \big[\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M\big]$, where $\tilde{\boldsymbol{x}}_i = \boldsymbol{F}\boldsymbol{x}_i$. Then, $\mathrm{enc}(\boldsymbol{X}, \boldsymbol{Y}_{C,i})$ is defined as the following equations:

$$\mathrm{enc}(\boldsymbol{X}, \boldsymbol{Y}_{C,i}) = \boldsymbol{O}'\bar{\boldsymbol{X}}\boldsymbol{p}, \quad (5)$$

$$\boldsymbol{p} \propto \exp(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{P}\tilde{\boldsymbol{y}}'_c), \quad (6)$$

where $\tilde{\boldsymbol{y}}'_c$ is a concatenation of output embedding vectors from $i - C + 1$ to i similar to $\tilde{\boldsymbol{y}}_c$, that is, $\tilde{\boldsymbol{y}}'_c = (\boldsymbol{E}'\boldsymbol{y}_{i-C+1} \cdots \boldsymbol{E}'\boldsymbol{y}_i)$. Moreover, $\bar{\boldsymbol{X}}$ is a matrix form of a list of averaged input word embeddings within window size Q, namely, $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_M]$, where $\bar{\boldsymbol{x}}_i = \sum_{q=i-Q}^{i+Q} \frac{1}{Q}\tilde{\boldsymbol{x}}_q$.

Equation 6 is generally referred to as the attention model, which is introduced to encode a relationship between input words and the previous C output words. For example, if the previous C output words are assumed to align to $\boldsymbol{x}_i$, then the surrounding Q words $(\boldsymbol{x}_{i-Q}, \ldots, \boldsymbol{x}_{i+Q})$ are highly weighted by Equation 5.

3 Proposed Method

Our assumption here is that syntactic and semantic features of an input sentence can greatly help for generating a headline. For example, the meanings of subjects, predicates, and objects in a generated headline should correspond to the ones appearing in an input sentence. Thus, we incorporate syntactic and semantic features into the framework of headline generation. This paper uses an AMR as a case study of the additional features.

3.1 AMR

An AMR is a rooted, directed, acyclic graph that encodes the meaning of a sentence. Nodes in an AMR graph represent 'concepts', and directed edges represent a relationship between nodes. Concepts

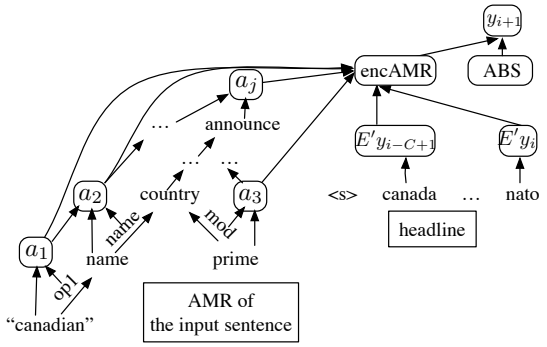


Figure 2: Model structure of our proposed attention-based AMR encoder; it outputs a headline using ABS and encoded AMR with attention.

consist of English words, PropBank event predicates, and special labels such as "person". For edges, AMR has approximately 100 relations (Banarescu et al., 2013) including semantic roles based on the PropBank annotations in OntoNotes (Hovy et al., 2006). To acquire AMRs for input sentences, we use the state-of-the-art transition-based AMR parser (Wang et al., 2015).

3.2 Attention-Based AMR Encoder

Figure 2 shows a brief sketch of the model structure of our attention-based AMR encoder model. We utilize a variant of child-sum Tree-LSTM originally proposed in (Tai et al., 2015) to encode syntactic and semantic information obtained from output of the AMR parser into certain fixed-length embedding vectors. To simplify the computation, we transform a DAG structure of AMR parser output to a tree structure, which we refer to as "*tree-converted AMR structure*". This transformation can be performed by separating multiple head nodes, which often appear for representing coreferential concepts, to a corresponding number of out-edges to head nodes. Then, we straightforwardly modify Tree-LSTM to also encode edge labels since AMR provides both node and edge labels, and original Tree-LSTM only encodes node labels.

Let $\boldsymbol{n}_j$ and $\boldsymbol{e}_j$ be N and E dimensional embeddings for labels assigned to the j-th node, and the out-edge directed to its parent node[2]. $\boldsymbol{W}_{in}$, $\boldsymbol{W}_{fn}$, $\boldsymbol{W}_{on}$, $\boldsymbol{W}_{un} \in \mathbb{R}^{D\times N}$ are weight matrices for node embeddings $\boldsymbol{n}_j$[3]. Similarly, $\boldsymbol{W}_{ie}$, $\boldsymbol{W}_{fe}$, $\boldsymbol{W}_{oe}$, $\boldsymbol{W}_{ue} \in \mathbb{R}^{D\times E}$ are weight matrices for edge embeddings $\boldsymbol{e}_j$. $\boldsymbol{W}_{ih}$, $\boldsymbol{W}_{fh}$, $\boldsymbol{W}_{oh}$, $\boldsymbol{W}_{uh} \in \mathbb{R}^{D\times D}$ are weight matrices for output vectors connected from child nodes. $B(j)$ represents a set of nodes, which have a direct edge to the j-th node in our tree-converted AMR structure. Then, we define embedding $\boldsymbol{a}_j$ obtained at node j in tree-converted AMR structure via Tree-LSTM as follows:

$$\tilde{\boldsymbol{h}}_j = \sum_{k\in B(j)} \boldsymbol{a}_k, \tag{7}$$

$$\boldsymbol{i}_j = \sigma\Big(\boldsymbol{W}_{in}\boldsymbol{n}_j + \boldsymbol{W}_{ie}\boldsymbol{e}_j + \boldsymbol{W}_{ih}\tilde{\boldsymbol{h}}_j\Big), \tag{8}$$

$$\boldsymbol{f}_{jk} = \sigma\Big(\boldsymbol{W}_{fn}\boldsymbol{n}_j + \boldsymbol{W}_{fe}\boldsymbol{e}_j + \boldsymbol{W}_{fh}\boldsymbol{a}_k\Big), \tag{9}$$

$$\boldsymbol{o}_j = \sigma\Big(\boldsymbol{W}_{on}\boldsymbol{n}_j + \boldsymbol{W}_{oe}\boldsymbol{e}_j + \boldsymbol{W}_{oh}\tilde{\boldsymbol{h}}_j\Big), \tag{10}$$

$$\boldsymbol{u}_j = \tanh\Big(\boldsymbol{W}_{un}\boldsymbol{n}_j + \boldsymbol{W}_{ue}\boldsymbol{e}_j + \boldsymbol{W}_{uh}\tilde{\boldsymbol{h}}_j\Big), \tag{11}$$

$$\boldsymbol{c}_j = \boldsymbol{i}_j \odot \boldsymbol{u}_j \sum_{k\in B(j)} \boldsymbol{f}_{jk} \odot \boldsymbol{c}_k, \tag{12}$$

$$\boldsymbol{a}_j = \boldsymbol{o}_j \odot \tanh(\boldsymbol{c}_j). \tag{13}$$

Let J represent the number of nodes in tree-converted AMR structure obtained from a given input sentence. We introduce $\boldsymbol{A} \in \mathbb{R}^{D\times J}$ as a matrix form of a list of hidden states $\boldsymbol{a}_j$ for all j, namely, $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_J]$. Let $\boldsymbol{O}'' \in \mathbb{R}^{|V|\times D}$ be a weight matrix for the output layer. Let $S \in \mathbb{R}^{D\times(CD)}$ be a weight matrix for mapping the context embedding of C output words onto embeddings obtained from nodes. Then, we define the attention-based AMR encoder '$\mathrm{encAMR}(\boldsymbol{A}, \boldsymbol{Y}_{C,i})$' as follows:

$$\mathrm{encAMR}(\boldsymbol{A}, \boldsymbol{Y}_{C,i}) = \boldsymbol{O}''\boldsymbol{A}\boldsymbol{s}, \tag{14}$$

$$\boldsymbol{s} \propto \exp(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}\tilde{\boldsymbol{y}}'_c). \tag{15}$$

Finally, we combine our attention-based AMR encoder shown in Equation 14 as an additional term of Equation 3 to build our headline generation system.

4 Experiments

To demonstrate the effectiveness of our proposed method, we conducted experiments on benchmark data of the abstractive headline generation task described in Rush et al. (2015).

[2]We prepare a special edge embedding for a root node.

[3]As with Equation 4, all the bias terms are omitted, though each weight matrix has one.

	DUC-2004			Gigaword test data used in (Rush et al., 2015)			Gigaword Our sampled test data		
Method	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
ABS (Rush et al., 2015)	26.55	7.06	22.05	30.88	12.22	27.77	–	–	–
ABS (re-run)	28.05	7.38	23.15	31.26	12.46	28.25	32.93	13.43	29.80
ABS+AMR	***28.80**	***7.83**	***23.62**	**31.64**	***12.94**	**28.54**	***33.43**	***13.93**	**30.20**
ABS+AMR(w/o attn)	28.28	7.21	23.12	30.89	12.40	27.94	31.32	12.83	28.46

Table 1: Results of methods on each dataset. We marked $*$ on the ABS+AMR results if we observed statistical difference ($p < 0.05$) between ABS (re-run) and ABS+AMR on the t-test. (R-1: ROUGE-1, R-2: ROUGE-2, R-L: ROUGE-L)

For a fair comparison, we followed their evaluation setting. The training data was obtained from the first sentence and the headline of a document in the annotated Gigaword corpus (Napoles et al., 2012)[4]. The development data is DUC-2003 data, and test data are both DUC-2004 (Over et al., 2007) and sentence-headline pairs obtained from the annotated Gigaword corpus as well as training data[5]. All of the generated headlines were evaluated by ROUGE (Lin, 2004)[6]. For evaluation on DUC-2004, we removed strings after 75-characters for each generated headline as described in the DUC-2004 evaluation. For evaluation on Gigaword, we forced the system outputs to be at most 8 words as in Rush et al. (2015) since the average length of headline in Gigaword is 8.3 words. For the preprocessing for all data, all letters were converted to lower case, all digits were replaced with '#', and words appearing less than five times with 'UNK'. Note that, for further evaluation, we prepared 2,000 sentence-headline pairs randomly sampled from the test data section of the Gigaword corpus as our additional test data.

In our experiments, we refer to the baseline neural attention-based abstractive summarization method described in Rush et al. (2015) as "**ABS**", and our proposed method of incorporating AMR structural information by a neural encoder to the baseline method described in Section 3 as "**ABS+AMR**". Additionally, we also evaluated the performance of the AMR encoder *without* the attention mechanism, which we refer to as "**ABS+AMR(w/o attn)**", to investigate the contribution of the attention mechanism on the AMR encoder. For the parameter estimation (training), we used stochastic gradient descent to learn parameters. We tried several values for the initial learning rate, and selected the value that achieved the best performance for each method. We decayed the learning rate by half if the log-likelihood on the validation set did not improve for an epoch. Hyper-parameters we selected were $D = 200$, $H = 400$, $N = 200$, $E = 50$, $C = 5$, and $Q = 2$. We re-normalized the embedding after each epoch (Hinton et al., 2012).

For ABS+AMR, we used the two-step training scheme to accelerate the training speed. The first phase learns the parameters of the ABS. The second phase trains the parameters of the AMR encoder by using 1 million training pairs while the parameters of the baseline ABS were fixed and unchanged to prevent overfitting.

Table 1 shows the recall of ROUGE (Lin, 2004) on each dataset. ABS (re-run) represents the performance of ABS re-trained by the distributed scripts[7]. We can see that the proposed method, ABS+AMR, outperforms the baseline ABS on all datasets. In particular, ABS+AMR achieved statistically significant gain from ABS (re-run) for ROUGE-1 and ROUGE-2 on DUC-2004. However in contrast, we observed that the improvements on Gigaword (the same test data as Rush et al. (2015)) seem to be limited compared with the DUC-2004 dataset. We assume that this limited gain is caused largely by the quality of AMR parsing results. This means that the

[4]Training data can be obtained by using the script distributed by the authors of Rush et al. (2015).

[5]Gigaword test data can be obtained from `https://github.com/harvardnlp/sent-summary`

[6]We used the ROUGE-1.5.5 script with option "`-n2 -m -b75 -d`", and computed the average of each ROUGE score.

[7]`https://github.com/facebook/NAMAS`

I(1): crown prince abdallah ibn abdel aziz left saturday at the head of saudi arabia 's delegation to the islamic summit in islamabad , the official news agency spa reported .
G: saudi crown prince leaves for islamic summit
A: crown prince leaves for islamic summit in saudi arabia
P: saudi crown prince leaves for islamic summit in riyadh

I(2): a massive gothic revival building once christened the lunatic asylum west of the <unk> was auctioned off for $ #.# million -lrb- euro# .# million -rrb- .
G: massive ##th century us mental hospital fetches $ #.# million at auction
A: west african art sells for $ #.# million in
P: west african art auctioned off for $ #.# million

I(3): brooklyn , the new bastion of cool for many new yorkers , is poised to go mainstream chic .
G: high-end retailers are scouting sites in brooklyn
A: new yorkers are poised to go mainstream with chic
P: new york city is poised to go mainstream chic

Figure 3: Examples of generated headlines on Gigaword. **I**: input, **G**: true headline, **A**: ABS (re-run), and **P**: ABS+AMR.

Gigaword test data provided by Rush et al. (2015) is already pre-processed. Therefore, the quality of the AMR parsing results seems relatively worse on this pre-processed data since, for example, many low-occurrence words in the data were already replaced with "UNK". To provide evidence of this assumption, we also evaluated the performance on our randomly selected 2,000 sentence-headline test data also taken from the test data section of the annotated Gigaword corpus. "Gigaword (randomly sampled)" in Table 1 shows the results of this setting. We found the statistical difference between ABS(re-run) and ABS+AMR on ROUGE-1 and ROUGE-2.

We can also observe that ABS+AMR achieved the best ROUGE-1 scores on all of the test data. According to this fact, ABS+AMR tends to successfully yield semantically important words. In other words, embeddings encoded through the AMR encoder are useful for capturing important concepts in input sentences. Figure 3 supports this observation. For example, ABS+AMR successfully added the correct modifier 'saudi' to "crown prince" in the first example. Moreover, ABS+AMR generated a consistent subject in the third example.

The comparison between ABS+AMR(w/o attn) and ABS+AMR (with attention) suggests that the attention mechanism is necessary for AMR encoding. In other words, the encoder without the attention mechanism tends to be overfitting.

5 Related Work

Recently, the Recurrent Neural Network (RNN) and its variant have been applied successfully to various NLP tasks. For headline generation tasks, Chopra et al. (2016) exploited the RNN decoder (and its variant) with the attention mechanism instead of the method of Rush et al. (2015): the combination of the feed-forward neural network language model and attention-based sentence encoder. Nallapati et al. (2016) also adapted the RNN encoder-decoder with attention for headline generation tasks. Moreover, they made some efforts such as hierarchical attention to improve the performance. In addition to using a variant of RNN, Gulcehre et al. (2016) proposed a method to handle infrequent words in natural language generation. Note that these recent developments do not conflict with our method using the AMR encoder. This is because the AMR encoder can be straightforwardly incorporated into their methods as we have done in this paper, incorporating the AMR encoder into the baseline. We believe that our AMR encoder can possibly further improve the performance of their methods. We will test that hypothesis in future study.

6 Conclusion

This paper mainly discussed the usefulness of incorporating structural syntactic and semantic information into novel attention-based encoder-decoder models on headline generation tasks. We selected abstract meaning representation (AMR) as syntactic and semantic information, and proposed an attention-based AMR encoder-decoder model. The experimental results of headline generation benchmark data showed that our attention-based AMR encoder-decoder model successfully improved standard automatic evaluation measures of headline generation tasks, ROUGE-1, ROUGE-2, and ROUGE-L. We believe that our results provide empirical evidence that syntactic and semantic information obtained from an automatic parser can help to improve the neural encoder-decoder approach in NLG tasks.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 140–149.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *CoRR*, abs/1207.0580.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 57–60.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the Association for Computational Linguistics Workshop*, pages 74–81.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 280–290.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing and Management*, 43(6):1506–1520.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 1556–1566.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015. Translating Videos to Natural Language Using Deep Recurrent Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 1494–1504.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR 2015)*, pages 3156–3164.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A Transition-based Algorithm for AMR Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 366–375.

Robust Gram Embeddings

Taygun Kekeç and **D.M.J. Tax**
Pattern Recognition and Bioinformatics Laboratory
Delft University of Technology
Delft, 2628CD, The Netherlands
taygunkekec@gmail.com,D.M.J.Tax@tudelft.nl

Abstract

Word embedding models learn vectorial word representations that can be used in a variety of NLP applications. When training data is scarce, these models risk losing their generalization abilities due to the complexity of the models and the overfitting to finite data. We propose a regularized embedding formulation, called *Robust Gram* (RG), which penalizes overfitting by suppressing the disparity between target and context embeddings. Our experimental analysis shows that the RG model trained on small datasets generalizes better compared to alternatives, is more robust to variations in the training set, and correlates well to human similarities in a set of word similarity tasks.

1 Introduction

Word embeddings represent each word as a unique vector in a linear vector space, encoding particular semantic and syntactic structure of the natural language (Arora et al., 2016). In various lingual tasks, these sequence prediction models shown superior results over the traditional count-based models (Baroni et al., 2014). Tasks such as sentiment analysis (Maas et al., 2011) and sarcasm detection (Ghosh et al., 2015) enjoys the merits of these features.

These word embeddings optimize features and predictors simultaneously, which can be interpreted as a factorization of the word cooccurence matrix C. In most realistic scenarios these models have to be learned from a small training set. Furthermore, word distributions are often skewed, and optimizing the reconstruction of $\hat{C}$ puts too much emphasis on the high frequency pairs (Levy and Goldberg, 2014). On the other hand, by having an unlucky and scarce data sample, the estimated $\hat{C}$ rapidly deviates from the underlying true cooccurence, in particular for low-frequency pairs (Lemaire and Denhire, 2008). Finally, noise (caused by stemming, removal of high frequency pairs, typographical errors, etc.) can increase the estimation error heavily (Arora et al., 2015).

It is challenging to derive a computationally tractable algorithm that solves all these problems. Spectral factorization approaches usually employ Laplace smoothing or a type of SVD weighting to alleviate the effect of the noise (Turney and Pantel, 2010). Alternatively, iteratively optimized embeddings such as Skip Gram (SG) model (Mikolov et al., 2013b) developed various mechanisms such as undersampling of highly frequent hub words apriori, and throwing rare words out of the training.

Here we propose a fast, effective and generalizable embedding approach, called Robust Gram, that penalizes complexity arising from the factorized embedding spaces. This design alleviates the need from tuning the aforementioned pseudo-priors and the preprocessing procedures. Experimental results show that our regularized model 1) generalizes better given a small set of samples while other methods yield insufficient generalization 2) is more robust to arbitrary perturbations in the sample set and alternations in the preprocessing specifications 3) achieves much better performance on word similarity task, especially when similarity pairs contains unique and hardly observed words in the vocabulary.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1060–1065,
Austin, Texas, November 1-5, 2016.

2 Robust Gram Embeddings

Let $|y| = V$ the vocabulary size and N be the total number of training samples. Denote x, y to be $V \times 1$ discrete word indicators for the context and target: corresponding to the context and word indicators c, w in word embedding literature. Define $\Psi_{d\times V}$ and $\Phi_{d\times V}$ as word and context embedding matrices. The projection on the matrix column space, Φx, gives us the embedding $\vec{x} \in R^d$. We use Φx and Φ_x interchangeably. Using a very general formulation for the regularized optimization of a (embedding) model, the following objective is minimized:

$$J = \sum_{i}^{N} \mathcal{L}(\Psi, \Phi, x_i, y_i) + g(\Psi, \Phi) \qquad (1)$$

where $\mathcal{L}(\Psi, \Phi, x_i, y_i)$ is the loss incurred by embedding example target y_i using context x_i and embedding parameters Ψ, Φ, and where $g(\Psi, \Phi)$ is a regularization of the embedding parameters. Different embedding methods differ in the form of specified loss function and regularization. For instance, the Skip Gram likelihood aims to maximize the following conditional:

$$\begin{aligned} \mathcal{L}(\Psi, \Phi, x, y) &= -\log p(y|x, \Phi, \Psi) \\ &= -\log \frac{\exp(\Psi_y^T \Phi_x)}{\sum_{y'} \exp(\Psi_{y'}^T \Phi_x)} \end{aligned} \qquad (2)$$

This can be interpreted as a generalization of Multinomial Logistic Regression (MLR). Rewriting $(\Psi y)^T(\Phi x) = (y^T \Psi^T \Phi x) = y^T W x = W_y x$ shows that the combination of Φ and Ψ become the weights in the MLR. In the regression the input x is transformed to directly predict y. The Skip Gram model, however, transforms both the context x and the target y, and can therefore be seen as a generalization of the MLR.

It is also possible to penalize the quadratic loss between embeddings (Globerson et al., 2007):

$$\mathcal{L}(.) = -\log \frac{\exp(-||\Psi_y - \Phi_x||^2)}{\sum_{y'} \exp(-||\Psi_{y'} - \Phi_x||^2)} \qquad (3)$$

Since these formulations predefine a particular embedding dimensionality d, they impose a low rank constraint on the factorization $W = \Psi^T \Phi$. This means that $g(\Psi, \Phi)$ contains $\lambda rank(\Phi^T \Psi)$ with a sufficiently large λ. The optimization with an explicit rank constraint is NP hard. Instead, approximate rank constraints are utilized with a Trace Norm (Fazel et al., 2001) or Max Norm (Srebro and Shraibman, 2005). However, adding such constraints usually requires semidefinite programs which quickly becomes computationally prohibitive even with a moderate vocabulary size.

Do these formulations penalize the complexity? Embeddings under quadratic loss are already regularized and avoids trivial solutions thanks to the second term. They also incorporate a bit weighted data-dependent ℓ_2 norm. Nevertheless, choosing a log-sigmoid loss for Equation 1 brings us to the Skip Gram model and in that case, ℓ_p regularization is not stated. Without such regularization, unbounded optimization of $2Vd$ parameters has potential to converge to solutions that does not generalize well.

To avoid this overfitting, in our formulation we choose g_1 as follows:

$$g_1 = \sum_{v}^{V} \lambda_1 \left(||\Psi_v||_2^2 + ||\Phi_v||_2^2 \right) \qquad (4)$$

where Ψ_v is the row vector of words.

Moreover, an appropriate regularization can also penalize the deviance between low rank matrices Ψ and Φ. Although there are words in the language that may have different context and target representations, such as *the* [1], it is natural to expect that a large proportion of the words have a shared representation in their context and target mappings. To this end, we introduce the following regularization:

$$g_2 = \lambda_2 ||\Psi - \Phi||_F^2 \qquad (5)$$

where F is the Frobenius matrix norm. This assumption reduces learning complexity significantly while a good representation is still retained, optimization under this constraint for large vocabularies is going to be much easier because we limit the degrees of freedom.

The Robust Gram objective therefore becomes:

$$LL + \lambda_1 \sum_{v}^{V} \left(||\Psi_v||_2^2 + ||\Phi_v||_2^2 \right) + \lambda_2 ||\Psi - \Phi||_F^2 \qquad (6)$$

[1] Consider prediction of *Suleiman* from *the*, and *the* from *oasis*. We expect *the* to have different vectorial representations.

where $LL = \sum_i^N \mathcal{L}(p(y_i|x_i, \Psi, \Phi))$ is the data log-likelihood, $p(y_i|x_i, \Psi, \Phi)$ is the loglinear prediction model, and $\mathcal{L}$ the cross entropy loss. Since we are in the pursuit of preserving/restoring low masses in $\hat{C}$, norms such as ℓ_2 allow each element to still possess a small probability mass and encourage smoothness in the factorized $\Psi^T\Phi$ matrix. As $\mathcal{L}$ is picked as the cross entropy, Robust Gram can be interpreted as a more principled and robust counterpart of Skip Gram objective.

One may ask what particular factorization Equation 6 induces. The objective searches for Ψ, Φ matrices that have similar eigenvectors in the vector space. A spectral PCA embedding obtains an asymmetric decomposition $W = U\Sigma V^T$ with $\Psi = U$ and $\Phi = \Sigma V$, albeit a convincing reason for embedding matrices to be orthonormal lacks. In the Skip Gram model, this decomposition is more symmetric since neither Ψ nor Φ are orthonormal and diagonal weights are distributed across the factorized embeddings. A symmetric factorization would be: $\Psi = U\Sigma^{0.5}$,$\Phi = \Sigma^{0.5}V^T$ as in (Levy and Goldberg, 2014). The objective in Eq. 6 converges to a more symmetric decomposition since $||\Psi - \Phi||$ is penalized. Still some eigenvectors across context and target maps are allowed to differ if they pay the cost. In this sense our work is related to power SVD approaches (Caron, 2000) in which one searches an a to minimize $||W - U\Sigma^a\Sigma^{1-a}V^T||$. In our formulation, if we enforce a solution by applying a strong constraint on $||\Psi - \Phi||_F^2$, then our objective will gradually converge to a symmetric powered decomposition such that $U \approx V$.

3 Experiments

The experiments are performed on a subset of the Wikipedia corpus containing approximately 15M words. For a systematic comparison, we use the same symmetric window size adopted in (Pennington et al., 2014), 10. Stochastic gradient learning rate is set to 0.05. Embedding dimensionality is set to 100 for model selection and sensitivity analysis. Unless otherwise is stated, we discard the most frequent 20 hub words to yield a final vocabulary of 26k words. To understand the relative merit of our approach [2] , Skip Gram model is picked as the baseline. To retain the learning speed, and avoid inctractability of maximum likelihood learning, we learn our embeddings with Noise Contrastive Estimation using a negative sample (Gutmann and Hyvärinen, 2012).

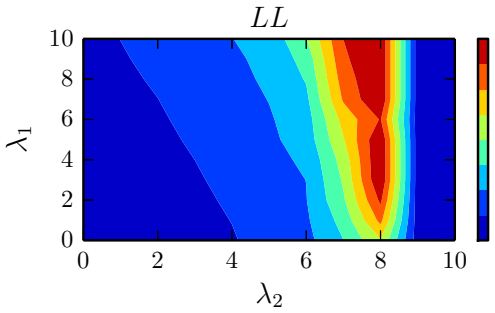


Figure 1: The LL objective for varying λ_1, λ_2.

3.1 Model Selection

For model selection, we are going to illustrate the log likelihood of different model instances. However, exact computation of the LL is computationally difficult since a full pass over the validation likelihood is time-consuming with millions of samples. Hence, we compute a stochastic likelihood with a few approximation steps. We first subsample a million samples rather than a full evaluation set, and then sample few words to predict in the window context similar to the approach followed in (Levy and Goldberg, 2014). Lastly, we approximate the normalization factor with one negative sample for each prediction score (Mnih and Kavukcuoglu, 2013)(Gutmann and Hyvärinen, 2012). Such an approximation works fine and gives smooth error curves. The reported likelihoods are computed by averaging over 5-fold cross validation sets.

Results. Figure 1 shows the likelihood LL obtained by varying $\{\lambda_1, \lambda_2\}$. The plot shows that there exits a unique minimum and both constraints contribute to achieve a better likelihood compared to their unregularized counterparts (for which $\lambda_1 = \lambda_2 = 0$). In particular, the regularization imposed by the differential of context and target embeddings g_2 contributes more than the regularization on the em-

[2] Our implementation can be downloaded from `github.com/taygunk/robust_gram_embeddings`

beddings Ψ and Φ separately. This is to be expected as g_2 also incorporates an amount of norm bound on the vectors. The region where both constraints are employed gives the best results. Observe that we can simply enhance the effect of g_2 by adding a small amount of bounded norm g_1 constraint in a stable manner. Doing this with pure g_2 is risky because it is much more sensitive to the selection of λ_2. These results suggest that the convex combination of stable nature of g_1 with potent regularizer of g_2, finally yields comparably better regularization.

3.2 Sensitivity Analysis

In order to test the sensitivity of our model and baseline Skip Gram to variations in the training set, we perform two sensitivity analyses. First, we simulate a missing data effect by randomly dropping out $\gamma \in [0, 20]$ percent of the training set. Under such a setting, robust models are expected to be effected less from the inherent variation. As an addition, we inspect the effect of varying the minimum cut-off parameter to measure the sensitivity. In this experiment, from a classification problem perspective, each instance is a sub-task with different number of classes (words) to predict. Instances with small cut-off introduces classification tasks with very few training samples. This cut-off choice varies in different studies (Pennington et al., 2014; Mikolov et al., 2013b), and it is usually chosen based on heuristic and storage considerations.

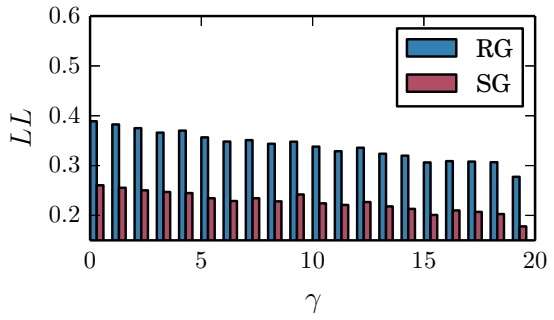


Figure 2: Training dropouts effect on LL.

Results. Figure 2 illustrates the likelihood of the Robust and Skip Gram model by varying the dropout ratio on the training set. As the training set shrinks, both models get lower LL. Nevertheless, likelihood decay of Skip Gram is relatively faster. When 20% drop is applied, the LL drops to 74% in the SG model. On the other hand the RG model not only starts with a much higher LL, the drop is also to 75.5%, suggesting that RG objective is more resistant to random variations in the training data.

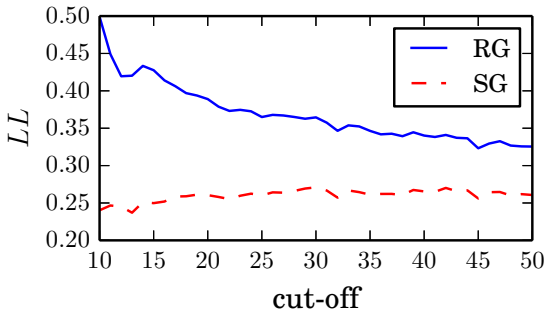


Figure 3: LL w.r.t the cut-off parameter.

Figure 3 shows the results of varying the rare-words cut-off threshold. We observe that the likelihood obtained by the Skip Gram is consistently lower than that of the Robust Gram. The graph shows that throwing out these rare words helps the objective of SG slightly. But for the Robust Gram removing the rare words actually means a significant decrease in useful information, and the performance starts to degrade towards the SG performance. RG avoids the overfitting occurring in SG, but still extracts useful information to improve the generalization.

3.3 Word Similarity Performance

The work of (Schnabel et al., 2015) demonstrates that intrinsic tasks are a better proxy for measuring the generic quality than extrinsic evaluations. Motivated by this observation, we follow the experimental setup of (Schnabel et al., 2015; Agirre et al., 2009), and compare word correlation estimates of each model to human estimated similarities with Spearman's correlation coefficient. The evaluation is performed on several publicly available word similarity datasets having different sizes. For datasets having multiple subjects annotating the word similarity, we compute the average similarity score from all subjects.

We compare our approach to set of techniques on the horizon of spectral to window based approaches. A fully spectral approach, HPCA, (Lebret and Le-

bret, 2013) extracts word embeddings by running a Hellinger PCA on the cooccurrence matrix. For this method, context vocabulary upper and lower bound parameters are set to $\{1, 10^{-5}\}$, as promoted by its author. GLoVe (Pennington et al., 2014) approach formulates a weighted least squares problem to combine global statistics of cooccurence and efficiency of window-based approaches. Its objective can be interpreted as an alternative to the cross-entropy loss of Robust Gram. The x_{max}, α values of the GLoVe objective is by default set to $100, 3/4$. Finally, we also compare to shallow representation learning networks such as Skip Gram and Continuous Bag of Words (CBoW) (Mikolov et al., 2013a), competitive state of the art window based baselines.

We set equal window size for all these models, and iterate three epochs over the training set. To yield more generality, all results obtained with 300 dimensional embeddings and subsampling parameters are set to 0. For Robust Gram approach, we have set $\lambda_1, \lambda_2 = \{0.3, 0.3\}$. To obtain the similarity results, we use the final Φ context embeddings.

Results. Table 1 depicts the results. The first observation is that in this setting, obtaining word similarity using HPCA and GLoVe methods are suboptimal. Frankly, we can conjecture that this scarce data regime is not in the favor of the spectral methods such as HPCA. Its poor performance can be attributed to its pure geometric reconstruction formulation, which runs into difficulties by the amount of inherent noise. Compared to these, CBoW's performance is moderate except in the RW dataset where it performs the worst. Secondly, the performance of the SG is relatively better compared to these approaches. Surprisingly, under this small data setting, RG outperforms all of its competitors in all datasets except for RG65, a tiny dataset of 63 words containing very common words. It is admissible that RG sacrifices a bit in order to generalize to a large variety of words. Note that it especially wins by a margin in MEN and Rare Words (RW) datasets, having the largest number of similarity query samples. As the number of query samples increases, RG embeddings' similarity modeling accuracy becomes clearly perceptible. The promising result Robust Gram achieves in RW dataset also sheds light on why CBoW performed worst on RW: CBOW overfits rapidly confirming the recent studies on the stability of CBoW (Luo et al., 2014). Finally, these word similarity results suggest that RG embeddings can yield much more generality under data scarcity.

	RG65	WS	WSS	WSR	MEN	RW
Size	63	353	203	252	3000	2034
CBoW	48.5	59.7	71.8	61.3	56.5	26.4
GloVe	48.9	56.2	61.5	59.1	53.0	30.0
SG	**59.2**	**71.7**	74.6	66.5	64.7	33.5
HPCA	32.1	48.6	52.9	51.5	49.9	30.7
RG	59.0	**71.7**	**74.8**	**66.7**	**65.8**	**34.0**

Table 1: Spearman's ρ coefficient. Higher is better.

4 Conclusion

This paper presents a regularized word embedding approach, called Robust Gram. In this approach, the model complexity is penalized by suppressing deviations between the embedding spaces of the target and context words. Various experimental results show that RG maintains a robust behaviour under small sample size situations, sample perturbations and it reaches a higher word similarity performance compared to its competitors. The gain from Robust Gram increases notably as diverse test sets are used to measure the word similarity performance.

In future work, by taking advantage of the promising results of Robust Gram, we intend to explore the model's behaviour in various settings. In particular, we plan to model various corpora, i.e. predictive modeling of sequentially arriving network packages. Another future direction might be encoding available domain knowledge by additional regularization terms, for instance, knowledge on synonyms can be used to reduce the degrees of freedom of the optimization. We also plan to enhance the underlying optimization by designing Elastic constraints (Zou and Hastie, 2005) specialized for word embeddings.

Acknowledgments

The authors acknowledge funding by the Dutch Organization for Scientific Research (NWO; grant 612.001.301). We also would like to thank Hamdi Dibeklioglu and Mustafa Unel for their kind support during this work.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *CoRR*, abs/1502.03520.

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

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, June.

John Caron. 2000. Experiments with lsa scoring: Optimal rank and basis. In *Proc. of SIAM Computational Information Retrieval Workshop*.

Maryam Fazel, Haitham Hindi, and Stephen P. Boyd. 2001. A rank minimization heuristic with application to minimum order system approximation. In *In Proceedings of the 2001 American Control Conference*, pages 4734–4739.

Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan. 2015. Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *EMNLP*, pages 1003–1012. The Association for Computational Linguistics.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *J. Mach. Learn. Res.*, 8:2265–2295.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1):307–361, February.

Rémi Lebret and Ronan Lebret. 2013. Word emdeddings through hellinger PCA. *CoRR*, abs/1312.5542.

Benot Lemaire and Guy Denhire. 2008. Effects of high-order co-occurrences on word semantic similarities. *CoRR*, abs/0804.0143.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

Qun Luo, Weiran Xu, and Jun Guo. 2014. A study on the cbow model's overfitting and stability. In *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, Web-KR '14, pages 9–12. ACM.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*, pages 298–307.

Nathan Srebro and Adi Shraibman. 2005. Rank, trace-norm and max-norm. In *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 545–560. Springer.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

SimpleScience: Lexical Simplification of Scientific Terminology

Yea-Seul Kim and **Jessica Hullman**
University Washington
Information School
yeaseul1, jhullman@uw.edu

Matthew Burgess
University of Michigan
Computer Science Department
mattburg@umich.edu

Eytan Adar
University of Michigan
School of Information
eadar@umich.edu

Abstract

Lexical simplification of scientific terms represents a unique challenge due to the lack of a standard parallel corpora and fast rate at which vocabulary shift along with research. We introduce SimpleScience, a lexical simplification approach for scientific terminology. We use word embeddings to extract simplification rules from a parallel corpora containing scientific publications and Wikipedia. To evaluate our system we construct SimpleSciGold, a novel gold standard set for science-related simplifications. We find that our approach outperforms prior context-aware approaches at generating simplifications for scientific terms.

1 Introduction

Lexical simplification, the process of reducing the complexity of words by replacing them with simpler substitutes (e.g., *sodium* in place of *Na*; *insects* in place of *lepidopterans*) can make scientific texts more accessible to general audiences. Human-in-the-loop interfaces present multiple possible simplifications to a reader (on demand) in place of jargon and give the reader familiar access points to understanding jargon (Kim et al., 2015). Unfortunately, simplification techniques are not yet of high enough quality for fully automated scenarios.

Currently lexical simplification pipelines for scientific texts are rare. The vast majority of prior methods assume a domain independent context, and rely on Wikipedia and Simple English Wikipedia, a subset of Wikipedia using simplified grammar and terminology, to learn simplifications (Biran et al., 2011; Paetzold and Specia, 2015), with translation-based approaches using an aligned version (Coster and Kauchak, 2011; Horn et al., 2014; Yatskar et al., 2010). However, learning simplifications from Wikipedia is not well suited to lexical simplification of scientific terms. Though generic or established terms may appear in Wikipedia, novel terms associated with new advances may not be reflected. Wikipedia's editing rules also favor generality over specificity and eliminate redundancy, both of which are problematic in providing a rich training set that matches simple and complex terms. Further, some approaches work by detecting all pairs of words in a corpus and filtering to isolate synonym or hypernym-relationship pairs using WordNet (Biran et al., 2011). Like Wikipedia, WordNet is a general purpose semantic database (Miller, 1995), and does not cover all branches of science nor integrate new terminology quickly.

Word embeddings do not require the use of prebuilt ontologies to identify associated terms like simplifications. Recent work indicates that they may improve results for simplification *selection*: determining which simplifications for a given complex word can be used without altering the meaning of the text (Paetzold and Specia, 2015). Embeddings have also been explored to extract hypernym relations from general corpora (Rei and Briscoe, 2014). However, word embeddings have not been used for *generating* lexical simplifications. We provide a novel demonstration of how using embeddings on a scientific corpus is better suited to learning scientific term simplifications than prior approaches that use WordNet as a filter and Wikipedia as a corpus.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1066–1071,
Austin, Texas, November 1-5, 2016.

INPUT: Finally we show that the transient immune activation that renders mosquitoes resistant to the human malaria parasite has little to no effect on mosquito fitness as a measure of survival or fecundity under laboratory conditions. **CANDIDATE RULES**: {fecundity→fertility} {fecundity→productivity} **OUTPUT**: Finally we show that the transient immune activation that renders mosquitoes resistant to the human malaria parasite has little to no effect on mosquito fitness as a measure of survival or (**fertility**; **productivity**) under laboratory conditions.

Table 1: Input sentence, candidate rules and output sentence. (Further examples provided as supplementary material.)

We introduce SimpleScience, a novel lexical simplification pipeline for scientific terms, which we apply to a scientific corpus of nearly 500k publications in Public Library of Science (PLOS) and PubMed Central (PMC) paired with a general corpus from Wikipedia. We validate our approach using SimpleSciGold, a gold standard set that we create using crowdsourcing that contains 293 sentences containing scientific terms with an average of 21 simplifications per term. We show how the SimpleScience pipeline achieves better performance (F-measure: 0.285) than the prior approach to simplification generation applied to our corpus (F-measure: 0.136). We further find that the simplification selection techniques used in prior work to determine which simplifications are a good fit for a sentence do not improve performance when our generation pipeline is used. [1]

2 Parallel corpora: Scientific and General

We assembled a *scientific corpus* of papers from the entire catalog of PLOS articles and the National Library of Medicine's Pubmed Central (PMC) archive (359,324 fulltext articles). The PLOS corpus of 125,378 articles includes articles from PLOS One and each speciality PLOS journal (e.g., Pathogens, Computational Biology). Our *general corpus* includes all 4,776,093 articles from the Feb. 2015 English Wikipedia snapshot. We chose Wikipedia as it covers many scientific concepts and usually contains descriptions of those concepts using simpler language than the research literature. We obtained all datasets from DeepDive (Ré and Zhang, 2015).

[1]Data and source code are available at: https://github.com/yeaseulkim/SimpleScience

3 SimpleScience Design

3.1 Generating Simplifications

Our goal is to learn simplification rules in the form *complex word→simple word*. One approach identifies all pairwise permutations of 'content' terms and then applies semantic (i.e., WordNet) and simplicity filters to eliminate pairs that are not simplifications(Biran et al., 2011). We adopt a similar pipeline but leverage distance metrics on word embeddings and a simpler frequency filter in place of WordNet. Embeddings identify words that share context in an unsupervised, scalable way and are more efficient than constructing co-occurrence matrices (Biran et al., 2011). As our experiments demonstrate, our approach improves performance on a scientific test set over prior work.

3.1.1 Step 1: Generating Word Embeddings

We used the Word2Vec system (Mikolov et al., 2013) to learn word vectors for each content word in the union of vocabulary of the scientific and general corpus. While other approaches exist (Pennington et al., 2014; Levy and Goldberg, 2014), Word2Vec has been shown to produce both fast and accurate results (Mikolov et al., 2013). We set the embedding *dimension* to 300, the *context-window* to 10, and use the skip-gram architecture with negative-sampling,which is known to produce quality results for rare entities (Mikolov et al., 2013).

3.1.2 Step 2: Filtering Pairs

Given the set of all pairwise permutations of words, we retain a simplification rule of two words w_1, w_2 if the cosine similarity $cos(w_1, w_2)$ between the word vectors is greater than a threshold a. We use grid search, described below, to parameterize a.

We then apply additional filtering rules. To avoid rules comprised of words with the same stem (e.g., *permutable*, *permutation*) we stem all words (using the Porter stemmer in the Python NLTK library (Bird et al., 2009)). The POS of each word is determined by Morphadorner (Burns, 2013) and pairs that differ in POS are omitted (e.g., *permutation* (noun), *change(d)* (verb)); Finally, we omit rules where one word is a prefix of the other and the suffix is one of *s, es, ed, ly, er*, or *ing*.

To retain only rules of the form *complex word* →

simple word we calculate the *corpus complexity*, C (Biran et al., 2011) of each word w as the ratio between the frequency (f) in the scientific versus general corpus: $C_w = f_{w,scientific}/f_{w,general}$. The *lexical complexity*, L, of a word is calculated as the word's character length, and the final complexity of the word as $C_w \times L_w$. We require that the final complexity score of the first word in the rule be greater than the second.

While this simplicity filter has been shown to work well in general corpora (Biran et al., 2011), it is sensitive to very small differences in the frequencies with which both words appear in the corpora. This is problematic given the distribution of terms in our corpora, where many rarer scientific terms may appear in small numbers in both corpora.

We introduce an additional constraint that requires that the second (simple) word in the rule occur in the general corpus at least k times. This helps ensure that we do not label words that are at a similar complexity level as simplifications. We note that this filter aligns with prior work that suggests that features of the hypernym in hypernym-hyponym relations influence performance more than features of the hyponym (Rei and Briscoe, 2014).

Parameterization: We use a grid search analysis to identify which measures of the set including $cos(w_1, w_2)$, $f_{w1,scientific}$, $f_{w2,scientific}$, $f_{w1,general}$, and $f_{w2,general}$ most impact the F-measure when we evaluate our generation approach against our scientific gold standard set (Sec. 4), and to set the specific parameter values. Using this method we identify a=0.4 for cosine similarity and k=3,000 for the frequency of the simple term in the general corpus. Full results are available in supplementary material.

3.2 Applying Simplifications

In prior context-aware simplification systems, the decision of whether to apply a simplification rule in an input sentence is complex, involving several similarity operations on word co-occurrence matrices (Biran et al., 2011) or using embeddings to incorporate co-occurrence context for pairs generated using other means (Paetzold and Specia, 2015). However, the SimpleScience pipline already considers the context of appearance for each word in deriving simplifications via word embeddings learned from a large corpus. We see no additional improvements in F-measure when we apply two variants of context similarity thresholds to decide whether to apply a rule to an input sentence. The first is the cosine similarity between the distributed representation of the simple word and the sum of the distributed representations of all words within a window l surrounding the complex word in the input sentence (Paetzold and Specia, 2015). The second is the cosine similarity of a minimum shared frequency co-occurrence matrix for the words in the pair and the co-occurrence matrix for the input sentence (Biran et al., 2011).

In fully automated applications, the top rule from the ranked candidate rules is used. We find that ranking by the cosine similarity between the word embeddings for the complex and simple word in the rule leads to the best performance at the top slot (full results in supplementary material).

4 Evaluation

4.1 SimpleSciGold Test Set

To evaluate our pipeline, we develop SimpleSciGold, a scientific gold standard set of sentences containing complex scientific terms which is modeled after the general purpose gold standard set created by Horn et al. (2014).

To create SimpleSciGold, we start with scientific terms from two sources: we utilized all 304 complex terms from unigram rules by (Vydiswaran et al., 2014), and added another 34,987 child terms from rules found by mining direct parent-child relations for unigrams in the Medical Subject Headings (MeSH) ontology (United States National Library of Medicine, 2015). We chose complex terms with pre-existing simplifications as it provided a means by which we could informally check the crowd generated simplifications for consistency.

To obtain words in context, we extracted 293 sentences containing unique words in this set from PLOS abstracts from PLOS Biology, Pathology, Genetics, and several other journals. We present 10 MTurk crowdworkers with a task ("HIT") showing one of these sentences with the complex word bolded. Workers are told to read the sentence, consult online materials (we provide direct links to a Wikipedia search, a standard Google search, and

		SimpleSciGold			
Method	**Corpus (Complex, Simple)**	**Number of Simplifications**	**Pot.**	**Prec.**	**F**
Biran et al. 2011	Wikipedia, SEW	17	0.059	0.036	0.044
	PLOS/PMC, Wikip.	588	0.352	0.084	0.136
SimpleScience ($cos \geq .4, f_{w,simple} \geq 3000$)	PLOS/PMC, Wikip.	2,322	0.526	0.196	**0.285**
SimpleScience ($cos \geq .4, f_{w,simple} \geq 0$)	PLOS/PMC, Wikip.	10,910,536	0.720	0.032	0.061

Table 2: Simplification Generation Results. SimpleScience achieves the highest F-measure with a cosine threshold of 0.4 and a frequency of the simple word in the general corpus of 3000.

a Google "define" search on the target term), and add their simplification suggestions. Crowdworkers first passed a multiple choice practice qualification in which they were presented with sentences containing three complex words in need of simplification along with screenshots of Wikipedia and dictionary pages for the terms. The workers were asked to identify which of 5 possible simplifications listed for each complex word would preserve the meaning while simplifying. 108 workers took part in the gold standard set creation task, completing an average of 27 HITs each. The resulting SimpleSciGold standard set consists of an average of 20.7 simplifications for each of the 293 complex words in corresponding sentences.

4.2 Simplification Generation

We compare our word embedding generation process (applied to our corpora) to Biran et al.'s (2011) approach (applied to the Wikipedia and Simple English Wikipedia corpus as well as our scientific corpora). Following the evaluation method used in Paetzold and Specia (2015), we calculate *potential* as the proportion of instances for which at least one of the substitutions generated is present in the gold standard set, *precision* as the proportion of generated instances which are present in the gold standard set, and *F-measure* as their harmonic mean.

Our SimpleScience approach outperforms the original approach by Biran et al. (2011) applied to the Wikipedia and SEW corpus as well as to the scientific corpus (Table 1).

4.3 Applying Simplifications

We find that neither prior selection approaches yield performance improvements over our generation process. We evaluate the performance of ranking candidate rules by cosine similarity (to find the top rule for a fully automated application), and achieve precision of 0.389 at the top slot. In our supplementary materials, we provide additional results for potential, precision and F-measure at varying numbers of slots (up to 5), where we test ranking by cosine similarity of embeddings as well as by the second filter used in our pair generation step: the frequency of the simple word in the simple corpus.

4.4 Antonym Prevalence Analysis

A risk of using Word2Vec in place of WordNet is that the simpler terms generated by our approach may represent terms with opposite meanings (antonyms). While a detailed analysis is beyond the scope of this paper, we compared the likelihood of seeing antonyms in our results using a gold standard set of antonyms for biology, chemistry, and physics terms from WordNik (Wordnik, 2009). Specifically, we created an antonym set consisting of the 304 terms from the biology, chemistry, and physics categories in Wictionary for which at least one antonym is listed in WordNik. We compared antonym pairs with rules that produced by the SimpleScience pipeline (Fig. 1). We observed that 14.5% of the time (44 out of 304 instances), an antonym appeared at the top slot among results. 51.3% of the time (156 out of 304 instances), no antonyms in the list appeared within the top 100 ranked results. These results suggest that further filters are necessary to ensure high enough quality results for fully automated applications of scientific term simplification.

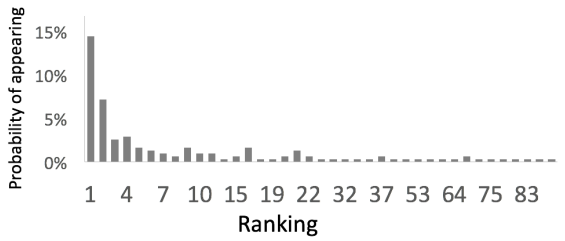


Figure 1: Probability of an antonym in our test set occurring as a suggested simpler term in the top 100 slots in the SimpleScience pipeline.

5 Limitations and Future Work

A risk of using Word2Vec to find related terms, rather than querying a lexical database like WordNet, is that generated rules may include antonyms. Adding techniques to filter antonym rules, such as using co-reference chains (Adel and Schütze, 2014), is important in future work.

We achieve a precision of 0.389 at the top slot on our SimpleSciGold standard set when we apply our generation method and rank candidates by cosine similarity. This level of precision is higher than that achieved by various prior ranking methods used in Lexenstein (Paetzold and Specia, 2015), with the exception of using machine learning techniques like SVM (Paetzold and Specia, 2015). Future work should explore how much the precision of our SimpleScience pipeline can be improved by adopting more sophisticated ranking methods. However, we suspect that even the highest precision obtained on general corpora and gold standard sets in prior work is not sufficient for fully automated simplification. An exciting area for future work is in applying the SimpleScience pipeline in interactive simplification suggestion interfaces for those reading or writing about science (Kim et al., 2015).

6 Conclusion

In this work, we introduce SimpleScience, a lexical simplification approach to address the unique challenges of simplifying scientific terminology, including a lack of parallel corpora, shifting vocabulary, and mismatch with using general purpose databases for filtering. We use word embeddings to extract simplification rules from a novel parallel corpora that contains scientific publications and Wikipedia. Using SimpleSciGold, a gold standard set that we created using crowdsourcing, we show that using embeddings and simple frequency filters on a scientific corpus outperforms prior approaches to simplification generation, and renders the best prior approach to simplification selection unnecessary.

References

[Adel and Schütze2014] Heike Adel and Hinrich Schütze. 2014. Using mined coreference chains as a resource for a semantic task. In *EMNLP*, pages 1447–1452.

[Biran et al.2011] Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: A context-aware approach to lexical simplification. In *ACL '11*. Association for Computational Linguistics.

[Bird et al.2009] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

[Burns2013] Philip R Burns. 2013. Morphadorner v2: a java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL*.

[Coster and Kauchak2011] William Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9. Association for Computational Linguistics.

[Horn et al.2014] Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

[Kim et al.2015] Yea-Seul Kim, Jessica Hullman, and Eytan Adar. 2015. Descipher: A text simplification tool for science journalism. In *Computation+Journalism Symposium*.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

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

[Miller1995] George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

[Paetzold and Specia2015] Gustavo Henrique Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. *ACL-IJCNLP 2015*, 1(1):85.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Ré and Zhang2015] Christopher Ré and Ce Zhang. 2015. Deepdive open datasets. `http://deepdive.stanford.edu/opendata`.

[Rei and Briscoe2014] Marek Rei and Ted Briscoe. 2014. Looking for hyponyms in vector space. In *CoNLL*, pages 68–77.

[United States National Library of Medicine2015] United States National Library of Medicine. 2015. Medical subject headings.

[Vydiswaran et al.2014] V.G.Vinod Vydiswaran, Qiaozhu Mei, David A. Hanauer, and Kai Zheng. 2014. Mining consumer health vocabulary from community-generated text. In *AMIA '14*.

[Wordnik2009] Wordnik. 2009. Wordnik online english dictionary. `https://www.wordnik.com/`.

[Yatskar et al.2010] Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *NAACL '10*. Association for Computational Linguistics.

Automatic Features for Essay Scoring – An Empirical Study

Fei Dong and **Yue Zhang**
Singapore University of Technology and Design
fei_dong@mymail.sutd.edu.sg and yue_zhang@sutd.edu.sg

Abstract

Essay scoring is a complicated processing requiring analyzing, summarizing and judging expertise. Traditional work on essay scoring focused on automatic handcrafted features, which are expensive yet sparse. Neural models offer a way to learn syntactic and semantic features automatically, which can potentially improve upon discrete features. In this paper, we employ convolutional neural network (CNN) for the effect of automatically learning features, and compare the result with the state-of-art discrete baselines. For in-domain and domain-adaptation essay scoring tasks, our neural model empirically outperforms discrete models.

1 Introduction

Automatic essay scoring (AES) is the task of building a computer-based grading system, with the aim of reducing the involvement of human raters as far as possible. AES is challenging since it relies not only on grammars, but also on semantics, discourse and pragmatics. Traditional approaches treat AES as a classification (Larkey, 1998; Rudner and Liang, 2002), regression (Attali and Burstein, 2004; Phandi et al., 2015), or ranking classification problem (Yannakoudakis et al., 2011; Chen and He, 2013), addressing AES by supervised learning. Features are typically bag-of-words, spelling errors and lengths, such word length, sentence length and essay length, etc. Some grammatical features are considered to assess the quality of essays (Yannakoudakis et al., 2011). A drawback is feature engineering, which can be time-consuming, since features need to be carefully handcrafted and selected to fit the approriate model. A further drawback of manual feature templates is that they are sparse, instantiated by discrete pattern-matching. As a result, parsers and semantic analyzers are necessary as a preprocessing step to offer syntactic and semantic patterns for feature extraction. Given variable qualities of student essays, such analyzers can be highly unreliable.

Neural network approaches have been shown to be capable of inducing dense syntactic and semantic features automatcially, giving competitive results to manually designed features for several tasks (Kalchbrenner et al., 2014; Johnson and Zhang, 2014; dos Santos and Gatti, 2014). In this paper, we empirically investigate a neural network method to learn features automatically for AES, without the need of external pre-processing. In particular, we build a hierarchical CNN model, with one lower layer representing sentence structures and one upper layer representing essay structure based on sentence representations. We compare automatically-induced features by the model with state-of-art baseline handcrafted features. Empirical results show that neural features learned by CNN are very effective in essay scoring task, covering more high-level and abstract information compared to manual feature templates.

2 Related Work

Following the first AES system Project Essay Grade (PEG) been developed in 1966 (Page, 1994), a number of commercial systems have come out, such as IntelliMetric 2, Intelligent Essay Assessor (IEA) (Foltz et al., 1999) and e-rater system (Attali and Burstein, 2004). The e-rater system now plays a

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1072–1077,
Austin, Texas, November 1-5, 2016.

second human rater's role in the Test of English as a Foreign Language (TOEFL) and Graduate Record Examination (GRE). The e-rater extracts a number of complex features, such as grammatical error and lexical complexity, and uses stepwise linear regression. IEA uses Latent Semantic Analysis (LSA) (Landauer et al., 1998) to create semantic vectors for essays and measure the semantic similarity between the vectors.

In the research literature, Larkey (1998) and Rudner and Liang (2002) treat AES as classification using bag-of-words features. Other recent work formulates the task as a preference ranking problem (Yannakoudakis et al., 2011; Chen and He, 2013). Yannakoudakis et al. (2011) formulated AES as a pairwise ranking problem by ranking the order of pair essays based on their quality. Features consist of word, POS n-grams features, complex grammatical features and so on. Chen and He (2013) formulated AES into a listwise ranking problem by considering the order relation among the whole essays and features contain syntactical features, grammar and fluency features as well as content and prompt-specific features. Phandi et al. (2015) use correlated Bayesian Linear Ridge Regression (cBLRR) focusing on domain-adaptation tasks. All these previous methods use discrete handcrafted features.

Recently, Alikaniotis et al. (2016) also employ a neural model to learn features for essay scoring automatically, which leverages a score-specific word embedding (SSWE) for word representations and a two-layer bidirectional long-short term memory network (LSTM) to learn essay representations. Alikaniotis et al. (2016) show that by combining SSWE, LSTM outperforms traditional SVM model. On the other hand, using LSTM alone does not give significantly more accuracies compared to SVM. This conforms to our preliminary experiments with the LSTM structure. Here, we use CNN without any specific embeddings and show that our neural models could outperform baseline discrete models on both in-domain and cross-domain senarios.

CNN has been used in many NLP applications, such as sequence labeling (Collobert et al., 2011) , sentences modeling (Kalchbrenner et al., 2014), sentences classification (Kim, 2014), text categorization (Johnson and Zhang, 2014; Zhang et al., 2015) and sentimental analysis (dos Santos and Gatti, 2014), etc. In this paper, we explore CNN representation ability for AES tasks on both in-domain and domain-adaptation settings.

Feature Type	Feature Description
Length	Number of characters, words, sentences, etc.
POS	Relative and absolute number of bad POS n-grams
Prompt	Relative and absolute number of words and their synonyms in the essay appearing in the prompt
Bag-of-words	Count of useful unigrams and bigrams (unstemmed, stemmed and spell corrected)

Table 1: Feature description used by EASE.

3 Baseline

Bayesian Linear Ridge Regression (BLRR) and Support Vector Regression (SVR) (Smola and Vapnik, 1997) are chosen as state-of-art baselines. Feature templates follow (Phandi et al., 2015), extracted by EASE[1], which are briefly listed in Table 1. "Useful n-grams" are determined using the Fisher test to separate the good scoring essays and bad scoring essays. Good essays are essays with a score greater than or equal to the average score, and the remainder are considered as bad scoring essays. The top 201 n-grams with the highest Fisher values are chosen as the bag of features and these top 201 n-grams constitute useful n-grams. Correct POS tags are generated using grammatically correct texts, which is done by EASE. The POS tags that are not included in the correct POS tags are treated as bad POS tags, and these bad POS tags make up the "bad POS n-grams" features.

The features tend to be highly useful for the in-domain task since the discrete features of same prompt data share the similar statistics. However, for different prompts, features statistics vary significantly. This raises challenges for discrete feature patterns.

ML-ρ (Phandi et al., 2015) was proposed to address this issue. It is based on feature augmentation, incorporating explicit correlation into augmented feature spaces. In particular, it expands baseline feature vector $\mathbf{x}$ to be $\Phi^s(\mathbf{x}) = (\rho\mathbf{x}, (1-\rho^2)^{1/2}\mathbf{x})$ and $\Phi^t(\mathbf{x}) = (\mathbf{x}, \mathbf{0_p})$ for source and target domain data

[1] https://github.com/edx/ease

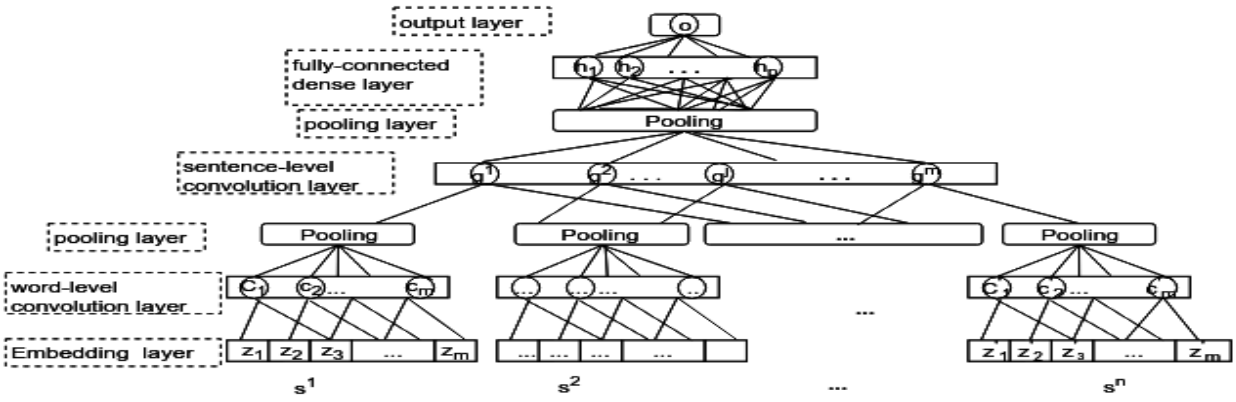


Figure 1: Hierarchical CNN structure

in R^{2p} respectively, with ρ being the correlation between source and target domain data. Then BLRR and maximum likelihood estimation are used to the optimize correlation. All the baseline models require POS-tagging as a pre-processing step, extracting syntactic features based on POS-tags.

4 Model

Word Representations We use word embedding with an embedding matrix $E_w \in R^{d_w \times V_w}$ where d_w is the embedding dimension, and V_w represents words vocabulary size. A word vector z_i is represented by $z_i = E_w w_i$ where w_i is the i-th word in a sentence. In contrast to the baseline models, our CNN model does not rely on POS-tagging or other pre-processing.

CNN Model We take essay scoring as a regression task and employ a two-layer CNN model, in which one convolutional layer is used to extract sentences representations, and the other is stacked on sentence vectors to learn essays representations. The architecture is depicted in Figure 1. Given an input sentence z_1, z_2, ..., z_n, a convolution layer with a *filter* $\mathbf{w} \in R^{h \times k}$ is applied to a window of h words to produce n-grams features. For instance, a feature c_i is generated from a window of words $z_{i:i+h-1}$ by $c_i = f(\mathbf{w} \cdot z_{i:i+h-1} + b)$, $b \in R$ is the bias term and f is the non-linear activation function rectified linear unit (ReLU).

The filter is applied to the all possible windows in a sentence to produce a feature map $\mathbf{c} = [c_1, c_2, ..., c_{m-h+1}]$. For $\mathbf{c}^j$ of the j-th sentence in an essay, max-pooling and average pooling function are used to produce the sentence vector $s^j = max\{\mathbf{c}^j\} \oplus avg\{\mathbf{c}^j\}$. The second convolutional layer takes s^1, s^2,..., s^n as inputs, followed by pooling layer (max-pooling and average-pooling) and a fully-connected hidden layer. The hidden layer directly connects to output layer which generates a score.

Set	#Essays	Genre	Avg Len.	Range	Med.
1	1783	ARG	350	2-12	8
2	1800	ARG	350	1-6	3
3	1726	RES	150	0-3	1
4	1772	RES	150	0-3	1
5	1805	RES	150	0-4	2
6	1800	RES	150	0-4	2
7	1569	NAR	250	0-30	16
8	723	NAR	650	0-60	36

Table 2: Details of the ASAP data; the last two columns are score range and median scores. For genre, ARG specifies *argumentative* essays, RES means *response* essays and NAR denotes *narrative* essays.

5 Experiments

5.1 Setup

Data We use the Automated Student Assessment Prize (ASAP)[2] dataset as evaluation data for our task, which contains 8 prompts of different genres as listed in Table 2. The essay scores are scaled into the range from 0 to 1. The settings of data preparation follow (Phandi et al., 2015). We use quadratic weighted kappa (QWK) as the metric. For domain-adaptation (cross-domain) experiments, we follow (Phandi et al., 2015), picking four pairs of essay prompts, namely, 1→2, 3→4, 5→6 and 7→8, where 1→2 denotes prompt 1 as source domain and prompt

[2]https://www.kaggle.com/c/asap-aes/data

Parameter	Parameter Name	Value
d_w	Word embedding dimension	100
h_{wrd}	Word context window size	5
h_{sent}	Sentence context window size	3
k_{wrd}	Word convolution units	50
k_{sent}	Sentence convolution units	50
p	Hidden size	50
drop_rate	Dropout rate	0.5
batch_size	Batch size	4
λ	Learning rate	0.01

Table 3: Neural Model Hyper-parameters

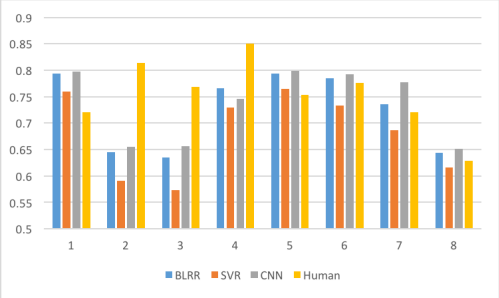


Figure 2: In-domain results

2 as target domain. All source domain essays are used as training data. Target domain data are randomly divided into 5 folds, where one fold is used as test data, and other 4 folds are collected together to sub-sample target domain train data. The sub-sampled sizes are 10, 25, 50, 100, with the larger sampled sets containing the smaller ones. And we repeated sub-sampling 5 times for each target training number to alleviate bias.

Hyper-parameters We use Adagrad for optimization. Word embeddings are randomly initialized and the hyper-parameter settings are listed in Table 3.

5.2 Results

In-domain The in-domain results are shown in Figure 2. The average values of all 8 prompt sets are listed in Table 4. For the in-domain task, CNN outperforms the baseline model SVR on all prompts of essay sets, and is competitive to BLRR. For the statistical significance, neural model is significantly better than baseline models with the p-value less than 10^{-5} at the confidence level of 95%. The average kappa value over 8 prompts is close to that of human raters.

Cross-domain The domain-adaptation results are shown in Table 5. It can be seen that our CNN

Model	BLRR	SVR	CNN	Human
Avg	0.725	0.682	0.734	0.754
Std dev	0.0025	0.0033	0.0029	—

Table 4: Indomain average kappa value and standard deviation over all 8 prompts.

Pairs	Method	$n_t = 10$	25	50	100
1→2	ML-ρ	0.365	0.437	0.521	**0.559**
	CNN	**0.546**	**0.569**	**0.563**	**0.559**
3→4	ML-ρ	0.435	0.540	0.590	0.619
	CNN	**0.628**	**0.656**	**0.659**	**0.662**
5→6	ML-ρ	0.415	0.600	0.678	0.718
	CNN	**0.647**	**0.700**	**0.714**	**0.750**
7→8	ML-ρ	0.328	0.438	0.496	0.551
	CNN	**0.570**	**0.590**	**0.568**	**0.587**

Table 5: Cross-domain results.

model outperforms ML-ρ on almost all pairs of adaptation experiments. ML-ρ domain-adaptation method's performance improves as the size of target domain training data increases. However, compared to ML-ρ, target training data size has less impact on our neural model. Even if the target training size is small, the neural model still gives strong performance. This results from the fact that neural model could learn more high-level and abstract features compared to traditional models with handcrafted discrete features. We plot the confusion matrix between truth and model prediction on test data in Figure 4, which shows that prediction scores of neural model tend to be closer to true values, which is very important in our task.

5.3 Feature Analysis

To visualize the features learned by our model, we use t-distributed stochastic neighbor embedding (t-SNE) (Van der Maaten and Hinton, 2008), projecting 50-dimensional features into 2-dimensional space. We take two domain pairs 3→4 and 5→6 as examples on the cross-domain task, extracting fully-connected hidden-layer features for target domain data using model trained on source domain data. The results are showed in Figure 3. The baseline discrete features are more concentrated, which shows that patterns on source prompt are weak in differentiating target prompt essays. By using ML-ρ and leveraging 100 target prompt training examples, the discrete features patterns are more scattered, increasing the differentiating power. In contrast, CNN

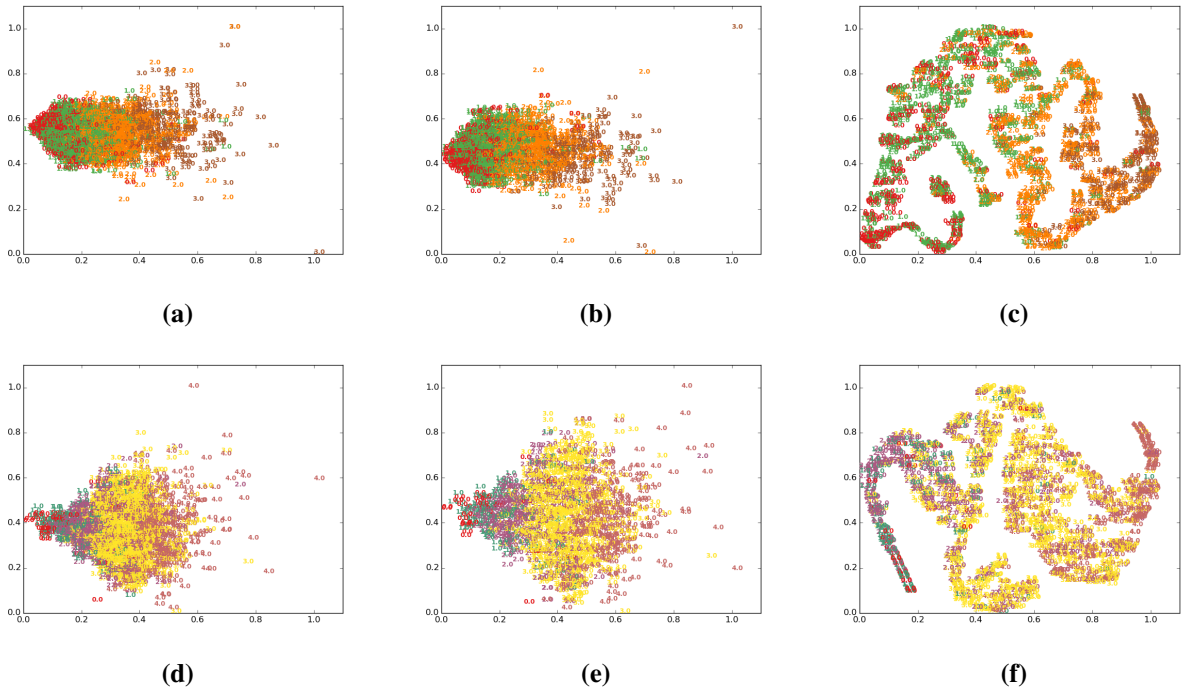


Figure 3: Visualization of discrete and neural features using t-SNE (each value represents an essay of the corresponding score). Top: Set 4 (3→4), Bottom: Set 6 (5→6). (a) discrete features; (b) ML-ρ features, $n_t = 100$; (c) neural features; (d) discrete features; (e) ML-ρ features, $n_t = 100$; (f) neural features.

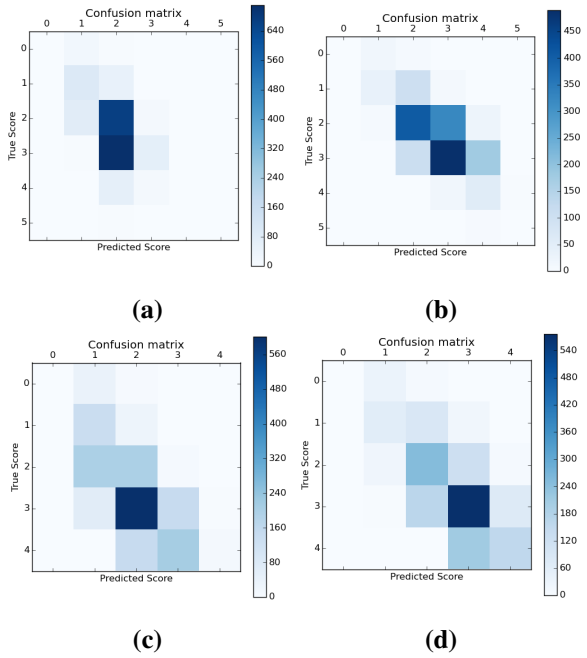


Figure 4: Confusion matrix of true and prediction scores by two different models on test data when target training size $n_t = 10$. (a) ML-ρ on 1→2; (b) CNN model on 1→2; (c) ML-ρ on 5→6; (d) CNN model on 5→6.

features trained on source prompt are sparse when used directly on the target prompt. This shows that neural features learned by the CNN model can better differentiate essays of different qualities. Without manual templates, such features automatically capture subtle and complex information that is relevant to the task.

6 Conclusion

We empirically investigated a hierarchical CNN model for automatic essay scoring, showing automatically learned features competitive to discrete handcrafted features for both in-domain and domain-adaptation tasks. The results demonstrate large potential for deep learning in AES.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. *arXiv preprint arXiv:1606.04289*.

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2):i–21.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *EMNLP*, pages 1741–1752.

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

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. Automated essay scoring: Applications to educational technology. In *proceedings of EdMedia*, volume 99, pages 40–64.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95. ACM.

Ellis Batten Page. 1994. Computer grading of student prose, using modern concepts and software. *The Journal of experimental education*, 62(2):127–142.

Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression.

Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2).

Alex Smola and Vladimir Vapnik. 1997. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Semantic Parsing with Semi-Supervised Sequential Autoencoders

Tomáš Kočiský[†‡] **Gábor Melis**[†] **Edward Grefenstette**[†]
Chris Dyer[†] **Wang Ling**[†] **Phil Blunsom**[†‡] **Karl Moritz Hermann**[†]
[†]Google DeepMind [‡]University of Oxford
{tkocisky,melisgl,etg,cdyer,lingwang,pblunsom,kmh}@google.com

Abstract

We present a novel semi-supervised approach for sequence transduction and apply it to semantic parsing. The unsupervised component is based on a generative model in which latent sentences generate the unpaired logical forms. We apply this method to a number of semantic parsing tasks focusing on domains with limited access to labelled training data and extend those datasets with synthetically generated logical forms.

1 Introduction

Neural approaches, in particular attention-based sequence-to-sequence models, have shown great promise and obtained state-of-the-art performance for sequence transduction tasks including machine translation (Bahdanau et al., 2015), syntactic constituency parsing (Vinyals et al., 2015), and semantic role labelling (Zhou and Xu, 2015). A key requirement for effectively training such models is an abundance of supervised data.

In this paper we focus on learning mappings from input sequences x to output sequences y in domains where the latter are easily obtained, but annotation in the form of (x, y) pairs is sparse or expensive to produce, and propose a novel architecture that accommodates semi-supervised training on sequence transduction tasks. To this end, we augment the transduction objective ($x \mapsto y$) with an autoencoding objective where the input sequence is treated as a latent variable ($y \mapsto x \mapsto y$), enabling training from both labelled pairs and unpaired output sequences. This is common in situations where we encode natural language into a logical form governed by some grammar or database.

While such an autoencoder could in principle be constructed by stacking two sequence transducers, modelling the latent variable as a series of discrete symbols drawn from multinomial distributions creates serious computational challenges, as it requires marginalising over the space of latent sequences Σ_x^*. To avoid this intractable marginalisation, we introduce a novel differentiable alternative for draws from a softmax which can be used with the reparametrisation trick of Kingma and Welling (2014). Rather than drawing a discrete symbol in Σ_x from a softmax, we draw a distribution over symbols from a logistic-normal distribution at each time step. These serve as continuous relaxations of discrete samples, providing a differentiable estimator of the expected reconstruction log likelihood.

We demonstrate the effectiveness of our proposed model on three semantic parsing tasks: the GEOQUERY benchmark (Zelle and Mooney, 1996; Wong and Mooney, 2006), the SAIL maze navigation task (MacMahon et al., 2006) and the Natural Language Querying corpus (Haas and Riezler, 2016) on OpenStreetMap. As part of our evaluation, we introduce simple mechanisms for generating large amounts of unsupervised training data for two of these tasks.

In most settings, the semi-supervised model outperforms the supervised model, both when trained on additional generated data as well as on subsets of the existing data.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1078–1087,
Austin, Texas, November 1-5, 2016.

Dataset	Example
GEO	what are the high points of states surrounding mississippi answer(high_point_1(state(next_to_2(stateid('mississippi'))))))
NLMAPS	Where are kindergartens in Hamburg? query(area(keyval('name','Hamburg')),nwr(keyval('amenity','kindergarten')),qtype(latlong))
SAIL	turn right at the bench into the yellow tiled hall $(1, 6, 90)$ FORWARD - FORWARD - RIGHT - STOP $(3, 6, 180)$

Table 1: Examples of natural language x and logical form y from the three corpora and tasks used in this paper. Note that the SAIL corpus requires additional information in order to map from the instruction to the action sequence.

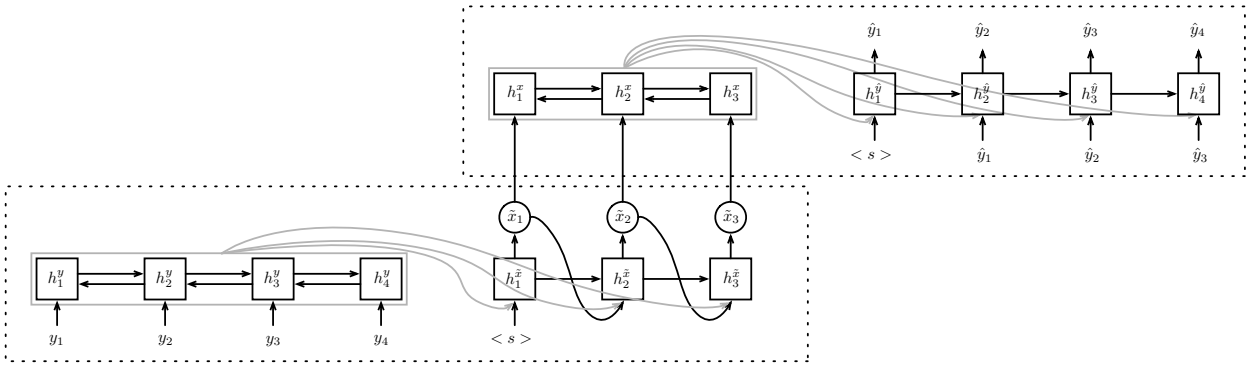


Figure 1: SEQ4 model with attention-sequence-to-sequence encoder and decoder. Circle nodes represent random variables.

2 Model

Our sequential autoencoder is shown in Figure 1. At a high level, it can be seen as two sequence-to-sequence models with attention (Bahdanau et al., 2015) chained together. More precisely, the model consists of four LSTMs (Hochreiter and Schmidhuber, 1997), hence the name SEQ4. The first, a bidirectional LSTM, encodes the sequence y; next, an LSTM with stochastic output, described below, draws a sequence of distributions $\tilde{x}$ over words in vocabulary Σ_x. The third LSTM encodes these distributions for the last one to attend over and reconstruct y as $\hat{y}$. We now give the details of these parts.

2.1 Encoding y

The first LSTM of the encoder half of the model reads the sequence y, represented as a sequence of one-hot vectors over the vocabulary Σ_y, using a bidirectional RNN into a sequence of vectors $h^y_{1:L_y}$ where L_y is the sequence length of y,

$$h^y_t = \left(f^{\rightarrow}_y(y_t, h^{y,\rightarrow}_{t-1}); f^{\leftarrow}_y(y_t, h^{y,\leftarrow}_{t+1})\right), \qquad (1)$$

where $f^{\rightarrow}_y, f^{\leftarrow}_y$ are non-linear functions applied at each time step to the current token y_t and their recurrent states $h^{y,\rightarrow}_{t-1}$, $h^{y,\leftarrow}_{t+1}$, respectively.

Both the forward and backward functions project the one-hot vector into a dense vector via an embedding matrix, which serves as input to an LSTM.

2.2 Predicting a Latent Sequence $\tilde{x}$

Subsequently, we wish to predict x. Predicting a discrete sequence of symbols through draws from multinomial distributions over a vocabulary is not an option, as we would not be able to backpropagate through this discrete choice. Marginalising over the possible latent strings or estimating the gradient through naïve Monte Carlo methods would be a prohibitively high variance process because the number of strings is exponential in the maximum length (which we would have to manually specify) with the vocabulary size as base. To allow backpropagation, we instead predict a sequence of distributions $\tilde{x}$ over the symbols of Σ_x with an RNN attending over

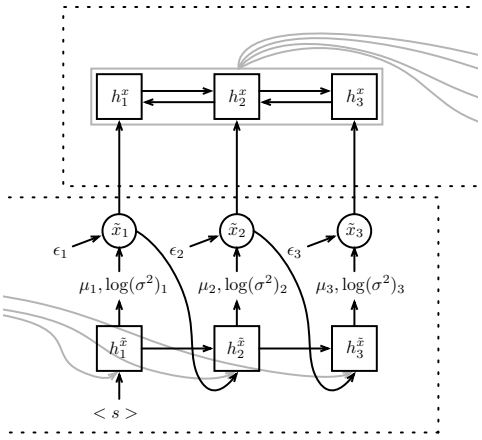


Figure 2: Unsupervised case of the SEQ4 model.

$h^y = h^y_{1:L_y}$, which will later serve to reconstruct y:

$$\tilde{x} = q(x|y) = \prod_{t=1}^{L_x} q(\tilde{x}_t|\{\tilde{x}_1, \cdots, \tilde{x}_{t-1}\}, h^y) \quad (2)$$

where $q(x|y)$ models the mapping $y \mapsto x$. We define $q(\tilde{x}_t|\{\tilde{x}_1, \cdots, \tilde{x}_{t-1}\}, h^y)$ in the following way:

Let the vector $\tilde{x}_t$ be a distribution over the vocabulary Σ_x drawn from a logistic-normal distribution[1], the parameters of which, $\mu_t, \log(\sigma^2)_t \in \mathbb{R}^{|\Sigma_x|}$, are predicted by attending by an LSTM attending over the outputs of the encoder (Equation 2), where $|\Sigma_x|$ is the size of the vocabulary Σ_x. The use of a logistic normal distribution serves to regularise the model in the semi-supervised learning regime, which is described at the end of this section. Formally, this process, depicted in Figure 2, is as follows:

$$h_t^{\tilde{x}} = f_{\tilde{x}}(\tilde{x}_{t-1}, h_{t-1}^{\tilde{x}}, h^y) \quad (3)$$

$$\mu_t, \log(\sigma_t^2) = l(h_t^{\tilde{x}}) \quad (4)$$

$$\epsilon \sim \mathcal{N}(0, I) \quad (5)$$

$$\gamma_t = \mu_t + \sigma_t \epsilon \quad (6)$$

$$\tilde{x}_t = \mathrm{softmax}(\gamma_t) \quad (7)$$

where the $f_{\tilde{x}}$ function is an LSTM and l a linear transformation to $\mathbb{R}^{2|\Sigma_x|}$. We use the reparametrisation trick from Kingma and Welling (2014) to draw from the logistic normal, allowing us to backpropagate through the sampling process.

[1]The logistic-normal distribution is the exponentiated and normalised (i.e. taking softmax) normal distribution.

2.3 Encoding x

Moving on to the decoder part of our model, in the third LSTM, we embed[2] and encode $\tilde{x}$:

$$h_t^x = \left(f_x^{\rightarrow}(\tilde{x}_t, h_{t-1}^{x,\rightarrow}); f_x^{\leftarrow}(\tilde{x}_t, h_{t+1}^{x,\leftarrow})\right) \quad (8)$$

When x is observed, during supervised training and also when making predictions, instead of the distribution $\tilde{x}$ we feed the one-hot encoded x to this part of the model.

2.4 Reconstructing y

In the final LSTM, we decode into y:

$$p(\hat{y}|\tilde{x}) = \prod_{t=1}^{L_y} p(\hat{y}_t|\{\hat{y}_1, \cdots, \hat{y}_{t-1}\}, h^{\tilde{x}}) \quad (9)$$

Equation 9 is implemented as an LSTM attending over $h^{\tilde{x}}$ producing a sequence of symbols $\hat{y}$ based on recurrent states $h^{\hat{y}}$, aiming to reproduce input y:

$$h_t^{\hat{y}} = f_{\hat{y}}(\hat{y}_{t-1}, h_{t-1}^{\hat{y}}, h^{\tilde{x}}) \quad (10)$$

$$\hat{y}_t \sim \mathrm{softmax}(l'(h_t^{\hat{y}})) \quad (11)$$

where $f_{\hat{y}}$ is the non-linear function, and the actual probabilities are given by a softmax function after a linear transformation l' of $h^{\hat{y}}$. At training time, rather than $\hat{y}_{t-1}$ we feed the ground truth y_{t-1}.

2.5 Loss function

The complete model described in this section gives a reconstruction function $y \mapsto \hat{y}$. We define a loss on this reconstruction which accommodates the unsupervised case, where x is not observed in the training data, and the supervised case, where (x, y) pairs are available. Together, these allow us to train the SEQ4 model in a semi-supervised setting, which experiments will show provides some benefits over a purely supervised training regime.

Unsupervised case When x isn't observed, the loss we minimise during training is the reconstruction loss on y, expressed as the negative log-likelihood $NLL(\hat{y}, y)$ of the true labels y relative to the predictions $\hat{y}$. To this, we add as a regularising

[2]Multiplying the distribution over words and an embedding matrix averages the word embedding of the entire vocabulary weighted by their probabilities.

term the KL divergence $KL[q(\gamma|y)\|p(\gamma)]$ which effectively penalises the mean and variance of $q(\gamma|y)$ from diverging from those of a prior $p(\gamma)$, which we model as a diagonal Gaussian $\mathcal{N}(0, I)$. This has the effect of smoothing the logistic normal distribution from which we draw the distributions over symbols of x, guarding against overfitting of the latent distributions over x to symbols seen in the supervised case discussed below. The unsupervised loss is therefore formalised as

$$\mathcal{L}_{unsup} = NLL(\hat{y}, y) + \alpha KL[q(\gamma|y)\|p(\gamma)] \quad (12)$$

with regularising factor α is tuned on validation, and

$$KL[q(\gamma|y)\|p(\gamma)] = \sum_{i=1}^{L_x} KL[q(\gamma_i|y)\|p(\gamma)] \quad (13)$$

We use a closed form of these individual KL divergences, described by Kingma and Welling (2014).

Supervised case When x is observed, we additionally minimise the prediction loss on x, expressed as the negative log-likelihood $NLL(\tilde{x}, x)$ of the true labels x relative to the predictions $\tilde{x}$, and do not impose the KL loss. The supervised loss is thus

$$\mathcal{L}_{sup} = NLL(\tilde{x}, x) + NLL(\hat{y}, y) \quad (14)$$

In both the supervised and unsupervised case, because of the continuous relaxation on generating $\tilde{x}$ and the reparameterisation trick, the gradient of the losses with regard to the model parameters is well defined throughout SEQ4.

Semi-supervised training and inference We train with a weighted combination of the supervised and unsupervised losses described above. Once trained, we simply use the $x \mapsto y$ decoder segment of the model to predict y from sequences of symbols x represented as one-hot vectors. When the decoder is trained without the encoder in a fully supervised manner, it serves as our supervised sequence-to-sequence baseline model under the name S2S.

3 Tasks and Data Generation

We apply our model to three tasks outlined in this section. Moreover, we explain how we generated additional unsupervised training data for two of these tasks. Examples from all datasets are in Table 1.

3.1 GeoQuery

The first task we consider is the prediction of a query on the GEO corpus which is a frequently used benchmark for semantic parsing. The corpus contains 880 questions about US geography together with executable queries representing those questions. We follow the approach established by Zettlemoyer and Collins (2005) and split the corpus into 600 training and 280 test cases. Following common practice, we augment the dataset by referring to the database during training and test time. In particular, we use the database to identify and anonymise variables (cities, states, countries and rivers) following the method described in Dong and Lapata (2016).

Most prior work on the GEO corpus relies on standard semantic parsing methods together with custom heuristics or pipelines for this corpus. The recent paper by Dong and Lapata (2016) is of note, as it uses a sequence-to-sequence model for training which is the unidirectional equivalent to S2S, and also to the decoder part of our SEQ4 network.

3.2 Open Street Maps

The second task we tackle with our model is the NLMAPS dataset by Haas and Riezler (2016). The dataset contains 1,500 training and 880 testing instances of natural language questions with corresponding machine readable queries over the geographical OpenStreetMap database. The dataset contains natural language question in both English and German but we focus only on single language semantic parsing, similar to the first task in Haas and Riezler (2016). We use the data as it is, with the only pre-processing step being the tokenization of both natural language and query form[3].

3.3 Navigational Instructions to Actions

The SAIL corpus and task were developed to train agents to follow free-form navigational route instructions in a maze environment (MacMahon et al., 2006; Chen and Mooney, 2011). It consists of a small number of mazes containing features such as objects, wall and floor types. These mazes come together with a large number of human instructions paired with the required actions[4] to reach the goal

[3] We removed quotes, added spaces around `()`, and separated the question mark from the last word in each question.

[4] There are four actions: `LEFT`, `RIGHT`, `GO`, `STOP`.

state described in those instructions.

We use the sentence-aligned version of the SAIL route instruction dataset containing 3,236 sentences (Chen and Mooney, 2011). Following previous work, we accept an action sequence as correct if and only if the final position and orientation exactly match those of the gold data. We do not perform any pre-processing on this dataset.

3.4 Data Generation

As argued earlier, we are focusing on tasks where aligned data is sparse and expensive to obtain, while it should be cheap to get unsupervised, monomodal data. Albeit that is a reasonable assumption for real world data, the datasets considered have no such component, thus the approach taken here is to generate random database queries or maze paths, i.e. the machine readable side of the data, and train a semi-supervised model. The alternative not explored here would be to generate natural language questions or instructions instead, but that is more difficult to achieve without human intervention. For this reason, we generate the machine readable side of the data for GEOQUERY and SAIL tasks[5].

For GEOQUERY, we fit a 3-gram Kneser-Ney (Chen and Goodman, 1999) model to the queries in the training set and sample about 7 million queries from it. We ensure that the sampled queries are different from the training queries, but do not enforce validity. This intentionally simplistic approach is to demonstrate the applicability of our model.

The SAIL dataset has only three mazes. We added a fourth one and over 150k random paths, including duplicates. The new maze is larger (21×21 grid) than the existing ones, and seeks to approximately replicate the key statistics of the other three mazes (maximum corridor length, distribution of objects, etc). Paths within that maze are created by randomly sampling start and end positions.

4 Experiments

We evaluate our model on the three tasks in multiple settings. First, we establish a supervised baseline to compare the S2S model with prior work. Next, we train our SEQ4 model in a semi-supervised setting on the entire dataset with the additional monomodal training data described in the previous section.

Finally, we perform an "ablation" study where we discard some of the training data and compare S2S to SEQ4. S2S is trained solely on the reduced data in a supervised manner, while SEQ4 is once again trained semi-supervised on the same reduced data plus the machine readable part of the discarded data (SEQ4-) or on the extra generated data (SEQ4+).

Model	Accuracy
Zettlemoyer and Collins (2005)	79.3
Zettlemoyer and Collins (2007)	86.1
Liang et al. (2013)	87.9
Kwiatkowski et al. (2011)	88.6
Zhao and Huang (2014)	88.9
Kwiatkowski et al. (2013)	89.0
Dong and Lapata (2016)	84.6
Jia and Liang (2016)[6]	89.3
S2S	86.5
SEQ4	87.3

Table 2: Non-neural and neural model results on GEOQUERY using the train/test split from (Zettlemoyer and Collins, 2005).

Training We train the model using standard gradient descent methods. As none of the datasets used here contain development sets, we tune hyperparameters by cross-validating on the training data. In the case of the SAIL corpus we train on three folds (two mazes for training and validation, one for test each) and report weighted results across the folds following prior work (Mei et al., 2016).

4.1 GeoQuery

The evaluation metric for GEOQUERY is the accuracy of exactly predicting the machine readable query. As results in Table 2 show, our supervised S2S baseline model performs slightly better than the comparable model by Dong and Lapata (2016). The semi-supervised SEQ4 model with the additional generated queries improves on it further.

The ablation study in Table 3 demonstrates a widening gap between supervised and semi-

[5]Our randomly generated unsupervised datasets can be downloaded from `http://deepmind.com/publications`

[6]Jia and Liang (2016) used hand crafted grammars to generate additional supervised training data.

Sup. data	S2S	SEQ4-	SEQ4+
5%	21.9	30.1	26.2
10%	39.7	42.1	42.1
25%	62.4	70.4	67.1
50%	80.3	81.2	80.4
75%	85.3	84.1	85.1
100%	86.5	86.5	87.3

Table 3: Results of the GEOQUERY ablation study.

Model	Accuracy
Haas and Riezler (2016)	68.30
S2S	78.03

Table 4: Results on the NLMAPS corpus.

supervised as the amount of labelled training data gets smaller. This suggests that our model can leverage unlabelled data even when only small amount of labelled data is available.

4.2 Open Street Maps

We report results for the NLMAPS corpus in Table 4, comparing the supervised S2S model to the results posted by Haas and Riezler (2016). While their model used a semantic parsing pipeline including alignment, stemming, language modelling and CFG inference, the strong performance of the S2S model demonstrates the strength of fairly vanilla attention-based sequence-to-sequence models. It should be pointed out that the previous work reports the number of correct answers when queries were executed against the dataset, while we evaluate on the strict accuracy of the generated queries. While we expect these numbers to be nearly equivalent, our evaluation is strictly harder as it does not allow for reordering of query arguments and similar relaxations.

We investigate the SEQ4 model only via the ablation study in Table 5 and find little gain through the semi-supervised objective. Our attempt at cheaply generating unsupervised data for this task was not successful, likely due to the complexity of the underlying database.

4.3 Navigational Instructions to Actions

Model extension The experiments for the SAIL task differ slightly from the other two tasks in that the language input does not suffice for choosing an action. While a simple instruction such as *'turn left'* can easily be translated into the action sequence `LEFT-STOP`, more complex instructions such as *'Walk forward until you see a lamp'* require knowledge of the agent's position in the maze.

Sup. data	S2S	SEQ4-
5%	3.22	3.74
10%	17.61	17.12
25%	33.74	33.50
50%	49.52	53.72
75%	66.93	66.45
100%	78.03	78.03

Table 5: Results of the NLMAPS ablation study.

To accomplish this we modify the model as follows. First, when encoding action sequences, we concatenate each action with a representation of the maze at the given position, representing the maze-state akin to Mei et al. (2016) with a bag-of-features vector. Second, when decoding action sequences, the RNN outputs an action which is used to update the agent's position and the representation of that new position is fed into the RNN as its next input.

Training regime We cross-validate over the three mazes in the dataset and report overall results weighted by test size (cf. Mei et al. (2016)). Both our supervised and semi-supervised model perform worse than the state-of-the-art (see Table 6), but the latter enjoys a comfortable margin over the former. As the S2S model broadly reimplements the work of Mei et al. (2016), we put the discrepancy in performance down to the particular design choices that we did not follow in order to keep the model here as general as possible and comparable across tasks.

The ablation studies (Table 7) show little gain for the semi-supervised approach when only using data from the original training set, but substantial improvement with the additional unsupervised data.

5 Discussion

Supervised training The prediction accuracies of our supervised baseline S2S model are mixed with respect to prior results on their respective tasks. For GEOQUERY, S2S performs significantly better than the most similar model from the literature (Dong and Lapata, 2016), mostly due to the fact that y and x are

Input from unsupervised data (y)	Generated latent representation (x)
answer smallest city loc_2 state stateid _STATE_	what is the smallest city in the state of _STATE_ </S>
answer city loc_2 state next_to_2 stateid _STATE_	what are the cities in states which border _STATE_ </S>
answer mountain loc_2 countryid _COUNTRY_	what is the lakes in _COUNTRY_ </S>
answer state next_to_2 state all	which states longer states show peak states to </S>

Table 8: Positive and negative examples of latent language together with the randomly generated logical form from the unsupervised part of the GEOQUERY training. Note that the natural language (x) does not occur anywhere in the training data in this form.

Model	Accuracy
Chen and Mooney (2011)	54.40
Kim and Mooney (2012)	57.22
Andreas and Klein (2015)	59.60
Kim and Mooney (2013)	62.81
Artzi et al. (2014)	64.36
Artzi and Zettlemoyer (2013)	65.28
Mei et al. (2016)	69.98
S2S	58.60
SEQ4	63.25

Table 6: Results on the SAIL corpus.

Sup. data	S2S	SEQ4-	SEQ4+
5%	37.79	41.48	43.44
10%	40.77	41.26	48.67
25%	43.76	43.95	51.19
50%	48.01	49.42	55.97
75%	48.99	49.20	57.40
100%	49.49	49.49	58.28

Table 7: Results of the SAIL ablation study. Results are from models trained on *L* and *Jelly* maps, tested on *Grid* only, hence the discrepancy between the 100% result and S2S in Table 6.

encoded with bidirectional LSTMs. With a unidirectional LSTM we get similar results to theirs.

On the SAIL corpus, S2S performs worse than the state of the art. As the models are broadly equivalent we attribute this difference to a number of task-specific choices and optimisations[7] made in Mei et al. (2016) which we did not reimplement for the sake of using a common model across all three tasks.

For NLMAPS, S2S performs much better than the state-of-the-art, exceeding the previous best result by 11% despite a very simple tokenization method and a lack of any form of entity anonymisation.

Semi-supervised training In both the case of GEOQUERY and the SAIL task we found the semi-supervised model to convincingly outperform the fully supervised model. The effect was particularly notable in the case of the SAIL corpus, where performance increased from 58.60% accuracy to 63.25% (see Table 6). It is worth remembering that the supervised training regime consists of three folds of tuning on two maps with subsequent testing on the third map, which carries a risk of overfitting to the training maps. The introduction of the fourth unsupervised map clearly mitigates this effect. Table 8 shows some examples of unsupervised logical forms being transformed into natural language, which demonstrate how the model can learn to sensibly ground unsupervised data.

Ablation performance The experiments with additional unsupervised data prove the feasibility of our approach and clearly demonstrate the usefulness of the SEQ4 model for the general class of sequence-to-sequence tasks where supervised data is hard to come by. To analyse the model further, we also look at the performance of both S2S and SEQ4 when reducing the amount of supervised training data available to the model. We compare three settings: the supervised S2S model with reduced training data, SEQ4- which uses the removed training data in an unsupervised fashion (throwing away the natural language) and SEQ4+ which uses the randomly generated unsupervised data described in Section 3. The S2S model behaves as expected on all three tasks, its performance dropping with the size of the training data. The performance of SEQ4- and SEQ4+ requires more analysis.

In the case of GEOQUERY, having unlabelled data from the true distribution (SEQ4-) is a good thing

[7] In particular we don't use beam search and ensembling.

when there is enough of it, as clearly seen when only 5% of the original dataset is used for supervised training and the remaining 95% is used for unsupervised training. The gap shrinks as the amount of supervised data is increased, which is as expected. On the other hand, using a large amount of extra, generated data from an approximating distribution (SEQ4+) does not help as much initially when compared with the unsupervised data from the true distribution. However, as the size of the unsupervised dataset in SEQ4- becomes the bottleneck this gap closes and eventually the model trained on the extra data achieves higher accuracy.

For the SAIL task the semi-supervised models do better than the supervised results throughout, with the model trained on randomly generated additional data consistently outperforming the model trained only on the original data. This gives further credence to the risk of overfitting to the training mazes already mentioned above.

Finally, in the case of the NLMAPS corpus, the semi-supervised approach does not appear to help much at any point during the ablation. These indistinguishable results are likely due to the task's complexity, causing the ablation experiments to either have to little supervised data to sufficiently ground the latent space to make use of the unsupervised data, or in the higher percentages then too little unsupervised data to meaningfully improve the model.

6 Related Work

Semantic parsing The tasks in this paper all broadly belong to the domain of semantic parsing, which describes the process of mapping natural language to a formal representation of its meaning. This is extended in the SAIL navigation task, where the formal representation is a function of both the language instruction and a given environment.

Semantic parsing is a well-studied problem with numerous approaches including inductive logic programming (Zelle and Mooney, 1996), string-to-tree (Galley et al., 2004) and string-to-graph (Jones et al., 2012) transducers, grammar induction (Kwiatkowski et al., 2011; Artzi and Zettlemoyer, 2013; Reddy et al., 2014) or machine translation (Wong and Mooney, 2006; Andreas et al., 2013).

While a large number of relevant literature focuses on defining the grammar of the logical forms (Zettlemoyer and Collins, 2005), other models learn purely from aligned pairs of text and logical form (Berant and Liang, 2014), or from more weakly supervised signals such as question-answer pairs together with a database (Liang et al., 2011). Recent work of Jia and Liang (2016) induces a synchronous context-free grammar and generates additional training examples (x, y), which is one way to address data scarcity issues. The semi-supervised setup proposed here offers an alternative solution to this issue.

Discrete autoencoders Very recently there has been some related work on discrete autoencoders for natural language processing (Suster et al., 2016; Marcheggiani and Titov, 2016, *i.a.*) This work presents a first approach to using effectively discretised sequential information as the latent representation without resorting to draconian assumptions (Ammar et al., 2014) to make marginalisation tractable. While our model is not exactly marginalisable either, the continuous relaxation makes training far more tractable. A related idea was recently presented in Gülçehre et al. (2015), who use monolingual data to improve machine translation by fusing a sequence-to-sequence model and a language model.

7 Conclusion

We described a method for augmenting a supervised sequence transduction objective with an autoencoding objective, thereby enabling semi-supervised training where previously a scarcity of aligned data might have held back model performance. Across multiple semantic parsing tasks we demonstrated the effectiveness of this approach, improving model performance by training on randomly generated unsupervised data in addition to the original data.

Going forward it would be interesting to further analyse the effects of sampling from a logistic-normal distribution as opposed to a softmax in order to better understand how this impacts the distribution in the latent space. While we focused on tasks with little supervised data and additional unsupervised data in y, it would be straightforward to reverse the model to train it with additional labelled data in x, i.e. on the natural language side. A natural extension would also be a formulation where semi-supervised training was performed in both x and y.

For instance, machine translation lends itself to such a formulation where for many language pairs parallel data may be scarce while there is an abundance of monolingual data.

References

Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional Random Field Autoencoders for Unsupervised Structured Prediction. In *Proceedings of NIPS*.

Jacob Andreas and Dan Klein. 2015. Alignment-based Compositional Semantics for Instruction Following. In *Proceedings of EMNLP*, September.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic Parsing as Machine Translation. In *Proceedings of ACL*, August.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning Compact Lexicons for CCG Semantic Parsing. In *Proceedings of EMNLP*, October.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Jonathan Berant and Percy Liang. 2014. Semantic Parsing via Paraphrasing. In *Proceedings of ACL*, June.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

David L. Chen and Raymond J. Mooney. 2011. Learning to Interpret Natural Language Navigation Instructions from Observations. In *Proceedings of AAAI*, August.

Li Dong and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. *arXiv preprint arXiv:1601.01280*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, May.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *arXiv preprint arXiv:1503.03535*.

Carolin Haas and Stefan Riezler. 2016. A corpus and semantic parser for multilingual natural language querying of openstreetmap. In *Proceedings of NAACL*, June.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING 2012*, December.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised PCFG Induction for Grounded Language Learning with Highly Ambiguous Supervision. In *Proceedings of EMNLP-CoNLL*, July.

Joohyun Kim and Raymond Mooney. 2013. Adapting Discriminative Reranking to Grounded Language Learning. In *Proceedings of ACL*, August.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of EMNLP*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *In Proceedings of EMNLP*. Citeseer.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning Dependency-based Compositional Semantics. In *Proceedings of the ACL-HLT*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions. In *Proceedings of AAAI*.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of ACL*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences. In *Proceedings of AAAI*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Simon Suster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual Learning of Multi-sense Embeddings with Discrete Autoencoders. *CoRR*, abs/1603.09128.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Proceedings of NIPS*.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of NAACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries using Inductive Logic Programming. In *Proceedings of AAAI/IAAI*, pages 1050–1055, August.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI*, pages 658–666. AUAI Press.

Luke Zettlemoyer and Michael Collins. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of EMNLP-CoNLL*, June.

Kai Zhao and Liang Huang. 2014. Type-driven incremental semantic parsing with polymorphism. *arXiv preprint arXiv:1411.5379*.

Jie Zhou and Wei Xu. 2015. End-to-end Learning of Semantic Role Labeling Using Recurrent Neural Networks. In *Proceedings of ACL*.

EQUATION PARSING : Mapping Sentences to Grounded Equations

Subhro Roy **Shyam Upadhyay** **Dan Roth**
University of Illinois, Urbana Champaign
{sroy9, upadhya3, danr}@illinois.edu

Abstract

Identifying mathematical relations expressed in text is essential to understanding a broad range of natural language text from election reports, to financial news, to sport commentaries to mathematical word problems. This paper focuses on identifying and understanding mathematical relations described within a single sentence. We introduce the problem of Equation Parsing – given a sentence, identify noun phrases which represent variables, and generate the mathematical equation expressing the relation described in the sentence. We introduce the notion of projective equation parsing and provide an efficient algorithm to parse text to projective equations. Our system makes use of a high precision lexicon of mathematical expressions and a pipeline of structured predictors, and generates correct equations in 70% of the cases. In 60% of the time, it also identifies the correct noun phrase $\rightarrow$ variables mapping, significantly outperforming baselines. We also release a new annotated dataset for task evaluation.

1 Introduction

Understanding text often involves reasoning with respect to quantities mentioned in it. Understanding the news article statement in Example 1 requires identifying relevant entities and the mathematical relations expressed among them in text, and determining how to compose them. Similarly, solving a math word problem with a sentence like Example 2, requires realizing that it deals with a *single* number, knowing the meaning of *"difference"* and composing the right equation – *"25"* needs to be subtracted from a number only after it is multiplied by 3.

Example 1 *Emanuel's campaign contributions total three times those of his opponents put together.*
Example 2 *Twice a number equals 25 less than triple the same number.*
Example 3 *Flying with the wind , a bird was able to make 150 kilometers per hour.*
Example 4 *The sum of two numbers is 80.*
Example 5 *There are 54 5-dollar and 10-dollar notes.*

As a first step towards understanding such relations, we introduce the Equation Parsing task - given a sentence expressing a mathematical relation, the goal is to generate an equation representing the relation, and to map the variables in the equation to their corresponding noun phrases. To keep the problem tractable, in this paper we restrict the final output equation form to have at most two (possibly coreferent) variables, and assume that each quantity mentioned in the sentence can be used at most once in the final equation.[1] In example 1, the gold output of an equation parse should be $V_1 = 3 \times V_2$, with $V_1 =$ "Emanuel's campaign contributions" and $V_2 =$ "those of his opponents put together".

The task can be seen as a form of semantic parsing (Goldwasser and Roth, 2011; Kwiatkowski et al., 2013) where instead of mapping a sentence to a logical form, we want to map it to an equation. However,

[1] We empirically found that around 97% of sentences describing a relation have this property.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1088–1097,
Austin, Texas, November 1-5, 2016.

there are some key differences that make this problem very challenging in ways that differ from the "standard" semantic parsing. In Equation Parsing, not all the components of the sentence are mapped to the final equation. There is a need to identify noun phrases that correspond to variables in the relations and determine that some are irrelevant and can be dropped. Moreover, in difference from semantic parsing into logical forms, in Equation Parsing multiple phrases in the text could correspond to the same variable, and identical phrases in the text could correspond to multiple variables.

We call the problem of mapping noun phrases to variables the problem of *grounding variables*. Grounding is challenging for various reasons, key among them are that: (i) The text often does not mention *"variables"* explicitly, e.g., the sentence in example 3 describes a mathematical relation between the speed of bird and the speed of wind, without mentioning "speed" explicitly. (ii) Sometimes, multiple noun phrases could refer to the same variable. For instance, in example 2, both *"a number"* and *"the same number"* refer to the same variable. On the other hand, the same noun phrase might refer to multiple variables, as in example 4, where the noun phrase "two numbers" refer to two variables.

In addition, the task involves deciding which of the quantities identified in the sentence are relevant to the final equation generation. In example 5, both "5" and "10" are not relevant for the final equation "$V_1 + V_2 = 54$". Finally, the equation needs to be constructed from a list of relevant quantities and grounded variables. Overall, the output space becomes exponential in the number of quantities mentioned in the sentence.

Determining the final equation that corresponds to the text is an inference step over a very large space. To address this, we define the concept of "projectivity" - a condition where the final equation can be generated by combining adjacent numbers or variables, and show that most sentences expressing mathematical relations exhibit the projectivity property. Finally, we restrict our inference procedure to only search over equations which have this property.

Our approach builds on a pipeline of structured predictors that identify irrelevant quantities, recognize coreferent variables, and, finally, generate equations. We also leverage a high precision lexicon of mathematical expressions and develop a greedy lexicon matching strategy to guide inference. We discuss and exemplify the advantages of this approach and, in particular, explain where the "standard" NLP pipeline fails to support equation parsing, and necessitates the new approach proposed here. Another contribution of this work is the development of a new annotated data set for the task of equation parsing. We evaluate our method on this dataset and show that our method predicts the correct equation in 70% of the cases and that in 60% of the time we also ground all variables correctly.

The next section presents a discussion of related work. Next we formally describe the task of equation parsing. The following sections describe our equation representation and the concept of projectivity, followed by the description of our algorithm to generate the equations and variable groundings from text. We conclude with experimental results.

2 Related Work

The work most related to this paper is (Madaan et al., 2016), which focuses on extracting relation triples where one of the arguments is a number. In contrast, our work deals with multiple variables and complex equations involving them. There has been a lot of recent work in automatic math word problem solving (Kushman et al., 2014; Roy et al., 2015; Hosseini et al., 2014; Roy and Roth, 2015). These solvers cannot handle sentences individually. They require the input to be a complete math word problem, and even then, they only focus on retrieving a set of answer values without mentioning what each answer value corresponds to. Our work is also conceptually related to work on semantic parsing – mapping natural language text to a formal meaning representation (Wong and Mooney, 2007; Clarke et al., 2010; Cai and Yates, 2013; Kwiatkowski et al., 2013; Goldwasser and Roth, 2011). However, as mentioned earlier, there are some significant differences in the task definition that necessitate the development of a new approach.

3 The Equation Parsing Task

Equation parsing takes as input a sentence x describing a single mathematical equation, comprising one or two variables and other quantities mentioned in x.

Let N be the set of noun phrases in the sentence x. The output of the task is the mathematical equation described in x, along with a mapping of each variable in the equation to its corresponding noun phrase in N. We refer to this mapping as the "grounding" of the variable; the noun phrase represents what the variable stands for in the equation. Table 1 gives an example of an input and output for the equation parsing of the text in example 2. Since an equation can be written in various forms, we use the form which most agrees with text, as our target output. So, for example 1, we will choose $V_1 = 3 \times V_2$ and not $V_2 = V_1 \div 3$. In cases where several equation forms seem to be equally likely to be the target equation, we randomly choose one of them, and keep this choice consistent across the dataset.

The Equation Parsing Task	
Input	*Twice a number equals 25 less than triple the same number.*
Output	$2 \times V_1 = (3 \times V_1) - 25$ *(Equation)* V_1 = "a number" *(Grounding)*

Table 1: Input and output for Equation Parsing

3.1 Equation Parse Representation

In this section, we introduce an equation parse for a sentence. An equation parse of a sentence x is a pair (T, E), where T represents a set of *triggers* extracted from x, and E represents an *equation tree* formed with the set T as leaves. We now describe these terms in detail.

Trigger Given a sentence x mentioning a mathematical relation, a trigger can either be a *quantity trigger* expressed in x, or *variable trigger* which is a noun phrase in x corresponding to a variable. A *quantity trigger* is a tuple (q, s), where q is the numeric value of the quantity mentioned in text, and s is the span of text from the sentence x which refers to the quantity. A *variable trigger* is a tuple (l, s), where l represents the label of the variable, and s represents the noun phrase representing the variable. For example, for the sentence in Fig 1, the spans "Twice", "25", and "triple" generate quantity triggers, whereas "a number" and "the same number" generate variable triggers, with label V_1.

Trigger List The trigger list T for a sentence x contains one trigger for each variable mention and each numeric value used in the final equation expressed by the sentence x. The trigger list might consist of multiple triggers having the same label, or extracted from the same span of text. In the example sentence in Fig 1, the trigger list comprises two triggers having the same label V_1. The final trigger list for the example in Fig 1 is {(2, "2"), (V_1, "a number"), (25, "25"), (3, "triple"), (V_1, "the same number")}. Note that there can be multiple valid trigger lists. In our example, we could have chosen both variable triggers to point to the same mention "a number". Quantity triggers in the trigger list form the ***quantity trigger list***, and the variable triggers in trigger list form the ***variable trigger list***.

Notation	Definition
Quantity Trigger	Mention of a quantity in text
Variable Trigger	Noun phrase coupled with variable label
Trigger	Quantity or variable trigger
Quantity Trigger List	List of quantity triggers, one for each number mention in equation
Variable Trigger List	List of variable triggers, one for each variable mention in equation
Trigger List	Union of quantity and variable trigger list
Equation Tree	Binary tree representation of equation
$lc(n)$, $rc(n)$	Left and right child of node n
$\textsc{Expr}(n)$	Expression represented by node n
$\odot(n)$	Operation at node n
$\textsc{Order}(n)$	Order of operation at node n
$\text{Location}(n)$	Character offset of trigger representing leaf node n
$\text{Span-Start}(n)$, $\text{Span-End}(n)$	Start and end character offsets of span covered by node n

Table 2: Summary of notations used in the paper

Equation Tree An equation tree of a sentence x is a binary tree whose leaves constitute the trigger list of x, and internal nodes (except the root) are labeled with one of the following operations – *addition, subtraction, multiplication, division*. In addition, for nodes which are labeled with subtraction or division, we maintain a separate variable to determine order of its children. The root of the tree is always labeled with the operation *equal*.

An equation tree is a natural representation for an equation. Each node n in an equation tree represents an expression $\textsc{Expr}(n)$, and the label of the parent node determines how the expressions of its children are to be composed to construct its own expression. Let us denote the label for a non-leaf node

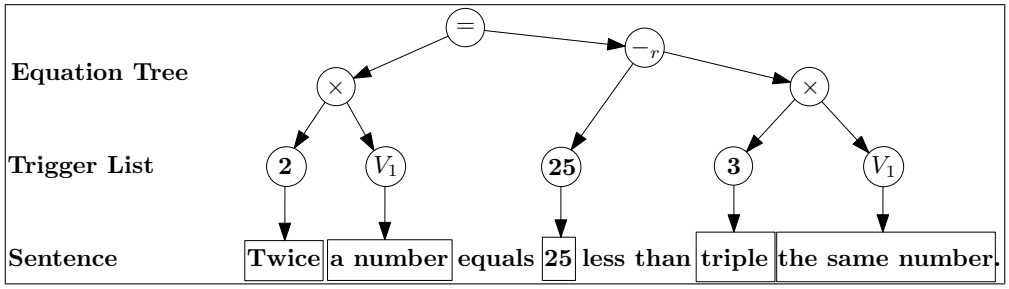


Figure 1: A sentence with its trigger list and equation tree. $-_r$ indicates subtraction with order rl.

n to be $\odot(n)$, where $\odot(n) \in \{+, -, \times, \div, =\}$ and the order of a node n's children by $\textsc{Order}(n)$ (defined only for subtraction and division nodes), which takes values lr (Left-Right) or rl (Right-Left). For a leaf node n, the expression $\textsc{Expr}(n)$ represents the variable label, if n is a variable trigger, and the numeric value of the quantity, if it is a quantity trigger. Finally, we use $lc(n)$ and $rc(n)$ to represent the left and right child of node n, respectively. The equation represented by the tree can be generated as follows. For all non-leaf nodes n, we have

$$\textsc{Expr}(n) = \begin{cases} \textsc{Expr}(lc(n)) \odot(n) \textsc{Expr}(rc(n)) \\ \quad \text{if } \odot(n) \in \{+, \times, =\} \\ \textsc{Expr}(lc(n)) \odot(n) \textsc{Expr}(rc(n)) \\ \quad \text{if } \odot(n) \in \{-, \div\} \wedge \textsc{Order}(n) = lr \\ \textsc{Expr}(rc(n)) \odot(n) \textsc{Expr}(lc(n)) \\ \quad \text{if } \odot(n) \in \{-, \div\} \wedge \textsc{Order}(n) = rl \end{cases} \quad (1)$$

Given an equation tree $\mathcal{T}$ of a sentence, the equation represented by it is the expression generated by the root of $\mathcal{T}$ (following Equation 1). Referring to the equation tree in Fig 1, the node marked "$-_r$" represents $(3 \times V_1) - 25$, and the root represents the full equation $2 \times V_1 = (3 \times V_1) - 25$.

4 Projectivity

For each leaf n of an equation tree T, we define a function $\text{Location}(\cdot)$, to indicate the position of the corresponding trigger in text. We also define for each node n of equation tree T, functions $\text{Span-Start}(n)$ and $\text{Span-End}(n)$ to denote the minimum span of text containing the leaves of the subtree rooted at n. We define them as follows:

$$\text{Span-Start}(n) = \begin{cases} \text{Location}(n) & \text{if } n \text{ is a leaf} \\ \min(\text{Span-Start}(lc(n)), \text{Span-Start}(rc(n))) & \\ \quad \text{otherwise} & \end{cases}$$

$$\text{Span-End}(n) = \begin{cases} \text{Location}(n) & \text{if } n \text{ is a leaf} \\ \max(\text{Span-End}(lc(n)), \text{Span-End}(rc(n))) & \\ \quad \text{otherwise} & \end{cases}$$

An equation tree T is called *projective* iff for every node n of T, either $\text{Span-End}(lc(n)) \leq \text{Span-Start}(rc(n))$ or $\text{Span-End}(rc(n)) \leq \text{Span-Start}(lc(n))$. In other words, the span of the left child and the right child cannot intersect in a projective equation tree[2].

The key observation, as our corpus analysis indicates, is that for most sentences, *there exists* a trigger list, such that the equation tree representing the relation in the sentence is projective. However this might involve mapping two mentions of the same variable to different noun phrases. Figure 1 shows an example of a projective equation tree, which requires different mentions of V_1 to be mapped to different noun phrases. If we had mapped both mentions of V_1 to same noun phrase "a number", the resulting equation tree would not have been projective. We collected 385 sentences which represent an equation with one or two mentions of variables, and each number in the sentence used at most once in the equation. We found that only one sentence among these could not generate a projective equation tree. (See Section 6.1 for details on dataset

[2]This is more general than the definition of projective trees used in dependency parsing (McDonald et al., 2005).

creation). Therefore, we develop an algorithmic approach for predicting projective equation trees, and show empirically that it compares favourably with ones which do not make the projective assumption.

5 Predicting Equation Parse

Equation parsing of a sentence involves predicting three components – Quantity Trigger List, Variable Trigger List and Equation Tree. We develop three structured prediction modules to predict each of the above components.

All our prediction modules take a similar form: given input x and output y, we learn a scoring function $f_w(x, y)$, which scores how likely is the output y given input x. The scoring function $f_w(x, y)$ is linear, $f_w(y) = w^T \phi(x, y)$, where $\phi(x, y)$ is a feature vector extracted from x and y. The inference problem, that is, the prediction y^* for an input x is then: $y^* = \arg\max_{y \in \mathcal{Y}} f_w(y)$, where $\mathcal{Y}$ is the set of all allowed values of y.

5.1 Predicting Quantity Trigger List

Given input text and the quantities mentioned in it, the role of this step is to identify , for each quantity in the text, whether it should be part of the final equation. For instance, in example 5 in Section 1, both "5" and "10" are not relevant for the final equation "$V_1 + V_2 = 54$". Similarly, in example 4, the number "two" is irrelevant for the equation "$V_1 + V_2 = 80$".

We define for each quantity q in the sentence, a boolean value $\mathrm{Relevance}(q)$, which is set to $true$ if q is relevant for the final equation, and to $false$ otherwise. For the structured classification, the input x is the sentence along with a set of recognized quantities mentioned in it, and the output y is the relevance values for all quantities in the sentence. We empirically found that predicting all relevance values jointly performs better than having a binary classifier predict each one separately. The feature function $\phi(x, y)$ used for the classification generates neighborhood features (from neighborhood of q) and quantity features (properties of the quantity mention). Details added to the appendix.

5.2 Predicting Variable Trigger List

The goal of this step is to predict the variable trigger list for the equation. Our structured classifier takes as input the sentence x, and the output y is either one or two noun-phrases, representing variables in the final equation. As we pointed out earlier, multiple groundings might be valid for any given variable, hence there can be multiple valid variable trigger lists. For every sentence x, we construct a set Y of valid outputs. Each element in Y corresponds to a valid variable trigger list. Finally, we aim to output only one of the elements of Y.

We modified the standard structured prediction algorithm to consider "superset supervision" and take into account multiple gold structures for an input x. We assume access to N training examples of the form : $(x_1, Y_1), (x_2, Y_2), \ldots, (x_N, Y_N)$, where each Y_i is a set of valid outputs for the sentence x_i. Since we want to output only one variable trigger list, we want to score at least one y from Y_i higher than all other possible outputs, for each x_i. We use a modified latent structured SVM to learn the weight vector w. The algorithm treats the best choice among all of Y_i as a latent variable. At each iteration, for all x_i, the algorithm chooses the best choice y_i^* from the set Y_i, according to the weight vector w. Then, w is updated by learning on all (x_i, y_i^*) by a standard structured SVM algorithm. The details of the algorithm are in Algorithm 1. The distinction from standard latent structural SVM is in line 5 of Algorithm 1. In order to get the best choice y_i^* for input x_i, we search only inside Y_i, instead of all of $\mathcal{Y}$. A similar formulation can be found in Björkelund and Kuhn

Algorithm 1 Structural SVM with Superset Supervision

Input: Training data $T = \{(x_1, Y_1), (x_2, Y_2), \ldots, (x_N, Y_N)\}$
Output: Trained weight vector w

1: $w \leftarrow w_0$
2: **repeat**
3: $\quad T' \leftarrow \emptyset$
4: $\quad$ **for all** $(x_i, Y_i) \in T$ **do**
5: $\quad\quad y_i^* \leftarrow \arg\max_{y \in Y_i} w^T \phi(x_i, y)$
6: $\quad\quad T' \leftarrow T' \cup \{(x_i, y_i^*)\}$
7: $\quad$ **end for**
8: $\quad$ Update w by running standard Structural SVM algorithm on T'
9: **until** convergence
10: **return** w

(2014). The features $\phi(x, y)$ used for variable trigger prediction include variable features (properties of noun phrase indicating variable) and neighborhood features (lexical features from neighborhood of variable mention). Details added to the appendix.

If the output of the classifier is a pair of noun phrases, we use a rule based variable coreference detector, to determine whether both noun phrases should have the same variable label or not. The rules for variable coreference are as follows :

1. If both noun phrases are the same, and they do not have the token "two" or "2", they have the same label.

2. If the noun phrases are different, and the noun phrase appearing later in the sentence contains tokens "itself", "the same number", they have the same label.

3. In all other cases, they have different labels.

Finally, each noun phrase contributes one variable trigger to the variable trigger list.

5.3 Predicting Equation Tree

It is natural to assume that the syntactic parse of the sentence could be very useful in addressing all the predictions we are making in the equation parsing tasks. However, it turns out that this is not the case – large portions of the syntactic parse will not be part of the equation parse, hence we need the aforementioned modules to address this. Nevertheless, in the next task of predicting the equation tree, we attempted to constraint the output space using guidance from the syntactic tree; we found, though, that even enforcing this weak level of output expectation is not productive. This was due to the poor performance of current syntactic parsers on the equation data (eg., in 32% of sentences, the Stanford parser made a mistake which does not allow recovering the correct equation).

The tree prediction module receives the trigger list predicted by the previous two modules, and the goal is to create an equation tree using the trigger list as the leaves of that tree. The input x is the sentence and the trigger list, and the output y is the equation tree representing the relation described in the sentence. We assume that the output will be a projective equation tree. For features $\phi(x, y)$, we extract for each non-leaf node n of the equation tree y, neighborhood features (from neighborhood of node spans of n's children), connecting text features (from text between the spans of n's children) and number features (properties of number in case of leaf nodes). Details are included in the appendix.

The projectivity assumption implies that the final equation tree can be generated by combining only adjacent nodes, once the set of leaves is sorted based on $\text{Span-Start}(\cdot)$ values. This allows us to use CKY algorithm for inference. A natural approach to further reduce the output space is to conform to the projective structure of the syntactic parse of the sentence. However, we found this to adversely affect performance, due to the poor performance of syntactic parser on equation data.

Lexicon To bootstrap the equation parsing process, we developed a high precision lexicon to translate mathematical expressions to operations and orders, like "sum of A and B" translates to "A+B", "A minus B" translates to "A-B", etc. (where A and B denote placeholder numbers or expressions). At each step of CKY, while constructing a node n of the equation tree, we check for a lexicon text expression corresponding to node n. If found, we allow only the corresponding operation (and order) for node n, and do not explore other operations or orders. We show empirically that reducing the space using this greedy lexicon matching help improve performance. We found that using the lexicon rules as features instead of hard constraints do not help as much. Note that our lexicon comprises only generic math concepts, and around 50% of the sentences in our dataset do not contain any pattern from the lexicon.

Finally, given input sentence, we first predict the quantity trigger and the variable trigger lists. Given the complete trigger list, we predict the equation tree relating the components of the trigger list.

5.4 Alternatives

A natural approach could be to jointly learn to predict all three components, to capture the dependencies among them. To investigate this, we developed a structured SVM which predicts all components jointly, using the union of the features of each component. We use approximate inference, first enumerating possible trigger lists, and then equation trees,

and find the best scoring structure. However, this method did not outperform the pipeline method. The worse performance of joint learning is due to: (1) search space being too large for the joint model to do well given our dataset size of 385, and (2) our independent classifiers being good enough, thus supporting better joint inference. This tradeoff is strongly supported in the literature (Punyakanok et al., 2005; Sutton and McCallum, 2007).

Another option is to enforce constraints between trigger list predictions, such as, variable triggers should not overlap with the quantity triggers. However, we noticed that often noun phrases returned by the Stanford parser were noisy, and would include neighboring numbers within the extracted noun phrases. This prevented us from enforcing such constraints.

6 Experimental Results

We now describe the data set, and the annotation procedure used. We then evaluate the system's performance on predicting trigger list, equation tree, and the complete equation parse.

6.1 Dataset

We created a new dataset consisting of 385 sentences extracted from algebra word problems and financial news headlines. For algebra word problems, we used the MIT dataset (Kushman et al., 2014), and two high school mathematics textbooks, Elementary Algebra (College of Redwoods) and Beginning and Intermediate Algebra (Tyler Wallace). Financial news headlines were extracted from The Latest News feed of MarketWatch, over the month of February, 2015. *All* sentences with information describing a mathematical relation among at most two (possibly coreferent) variables, were chosen. Next, we pruned sentences which require multiple uses of a number to create the equation. This only removed a few time related sentences like *"In 10 years, John will be twice as old as his son."*. We empirically found that around 97% of sentences describing a relation fall under the scope of our dataset.

The annotators were shown each sentence paired with the normalized equation representing the relation in the sentence. For each variable in the equation, the annotators were asked to mark spans of text which best describe what the variable represents. The annotation guidelines are provided in the appendix. We wanted to consider only noun phrase constituents for variable grounding. Therefore, for each annotated span, we extracted the noun phrase with maximum overlap with the span, and used it to represent the variables. Finally, a tuple with each variable being mapped to one of the noun phrases representing it, forms a valid output grounding (variable trigger list). We computed inter-annotator agreement on the final annotations where only noun phrases represent variables. The agreement (kappa) was 0.668, indicating good agreement. The average number of mention annotations per sentence was 1.74.

6.2 Equation Parsing Modules

In this section, we evaluate the performance of the individual modules of the equation parsing process. We report Accuracy - the fraction of correct predictions. Table 3 shows the 5-fold cross validation accuracy of the various modules. In each case, we also report accuracy by removing each feature group, one at a time. In addition, for equation tree prediction, we also show the effect of lexicon, projectivity, conforming to syntactic parse constraints, and using lexicon as features instead of hard constraints. For all our experiments, we use the Stanford Parser (Socher et al., 2013), the Illinois POS tagger (Roth and Zelenko, 1998) and the Illinois-SL structured prediction package (Chang et al., 2015).

6.3 Equation Parsing Results

In this section, we evaluate the performance of our system on the overall equation parsing task. We report Equation Accuracy - the fraction of sentences for which the system got the equation correct, and Equation+Grounding Accuracy - the fraction of sentences for which the system got both the equation and the grounding of variables correct. Table 4 shows the overall performance of our system, on a 5-fold cross validation. We compare against Joint Learning - a system which jointly learns to predict all relevant components of an equation parse (Section 5.4). We also compare with SPF (Artzi and Zettlemoyer, 2013), a publicly available semantic parser, which can learn from sentence-logical form pairs. We train SPF with sentence-equation pairs

Quantity Trigger List Prediction	Accuracy
All features	**95.3**
No Neighborhood features	42.5
No Quantity features	93.2

Variable Trigger List Prediction	Accuracy
All features	**75.5**
No Variable features	58.6
No Neighborhood features	70.3

Equation Tree Prediction	Accuracy
All features	**78.9**
No Neighborhood features	64.3
No Connecting Text features	70.2
No Number features	77.6
No Lexicon	72.7
No Projectivity	72.8
Conform with Syntactic Parse	70.2
Lexicon as Features	74.5

Table 3: Performance of system components

Source	Equation Accuracy	Equation + Grounding Accuracy
Our System	**71.3**	**61.2**
Joint Learning	60.9	50.0
SPF	3.1	N/A

Table 4: Performance on equation parsing

and a seed lexicon for mathematical terms (similar to ours), and report equation accuracy. Our structured predictors pipeline approach is shown to be superior to both Joint Learning and SPF.

SPF gets only a few sentences correct. We attribute this to the inability of SPF to handle overlapping mentions (like in Example 4), as well as its approach of parsing the whole sentence to the final output form. The developers of SPF also confirmed [3] that it is not suitable for equation parsing and that these results are expected. Since equation parsing is a more involved process, a slight adaptation of SPF does not seem possible, necessitating a more involved process , of the type we propose. Our approach, in contrast to SPF, can handle overlapping mentions, selects triggers from text, and parses the trigger list to form equations.

[3]Private communication

6.4 Error Analysis

For variable trigger list prediction, around 25% of the errors were due to the predictor choosing a span which is contained within the correct span, e.g., when the target noun phrase is "The cost of a child's ticket", our predictor chose only "child's ticket". Although this choice might be sufficient for downstream tasks, we consider it to be incorrect in our current evaluation. Another 25% of the errors were due to selection of entities which do not participate in the relation. For example, in "A rancher raises 5 times as many cows as horses.", our predictor chose "A rancher" and "cows" as variables, whereas the relation exists between "cows" and "horses". For the prediction of the equation tree, we found that 35% of the errors were due to rare math concepts expressed in text. For example, "7 dollars short of the price" represents 7 dollars should be subtracted from the price. These errors can be handled by carefully augmenting the lexicon. Another 15% of the errors were due to lack of world knowledge, requiring understanding of time, speed, and distance.

7 Conclusion

This paper investigates methods that identify and understand mathematical relations expressed in text. We introduce the equation parsing task, which involves generating an equation from a sentence and identifying what the variables represent. We define the notion of projectivity, and construct a high precision lexicon, and use these to reduce the equation search space. Our experimental results are quite satisfying and raise a few interesting issues. In particular, it suggests that predicting equation parses using a pipeline of structured predictors performs better than jointly trained alternatives. As discussed, it also points out the limitation of the current NLP tools in supporting these tasks. Our current formulation has one key limitation; we only deal with expressions that are described within a sentence. Our future work will focus on lifting this restriction, in order to allow relations expressed across multiple sentences and multiple relations expressed in the same sentence. Code and dataset are available at `http://cogcomp.cs.illinois.edu/page/publication_view/800`.

Acknowledgements

This work is funded by DARPA under agreement number FA8750-13-2-0008, and a grant from the Allen Institute for Artificial Intelligence (allenai.org).

A Features

A.1 Quantity Trigger List Prediction

The feature function $\phi(x, y)$ used for the classification generates the following features :

1. **Neighborhood features** : For each quantity q in the input sentence, we add unigrams and bigrams generated from a window around q, part of speech tags of neighborhood tokens of q. We conjoin these features with $\text{Relevance}(q)$.

2. **Quantity Features** : For each quantity q, we add unigrams and bigrams of the phrase representing the quantity. Also, we add a feature indicating whether the number is associated with number one or two, and whether it is the only number present in the sentence. These features are also conjoined with $\text{Relevance}(q)$.

A.2 Variable Trigger List Prediction

The features $\phi(x, y)$ used for variable trigger prediction are as follows:

1. **Variable features** : Unigrams and bigrams generated from the noun phrase representing variables, part of speech tags of tokens in noun phrase representing variables.

2. **Neighborhood Features** : Unigrams and POS tags from neighborhood of variables.

All the above features are conjoined with two labels, one denoting whether y has two variables or one, and the second denoting whether y has two variables represented by the same noun phrase.

A.3 Equation Tree Prediction

For features $\phi(x, y)$, we extract for each non-leaf node n of the equation tree y, the following:

1. **Neighborhood Features** : Unigrams, bigrams and POS tags from neighborhood of $\text{Span-Start}(lc(n))$, $\text{Span-Start}(rc(n))$, $\text{Span-End}(lc(n))$ and $\text{Span-End}(rc(n))$, conjoined with $\odot(n)$ and $\text{ORDER}(n)$.

2. **Connecting Text Features** : Unigrams, bigrams and part of speech tags between $\min(\text{Span-End}(lc(n)), \text{Span-End}(rc(n)))$ and $\max(\text{Span-Start}(lc(n)), \text{Span-Start}(rc(n)))$, conjoined with $\odot(n)$ and $\text{ORDER}(n)$.

3. **Number Features** : In case we are combining two leaf nodes representing quantity triggers, we add a feature signifying whether one number is larger than the other.

B Annotation Guidelines

The annotators were shown each sentence paired with the normalized equation representing the relation in the sentence. For each variable in the equation, the annotators were asked to mark spans of text which best describe what the variable represents. They were asked to annotate associated entities if exact variable description was not present. For instance, in example 3 (Section 1), the relation holds between the speed of bird and the speed of wind. However, *"speed"* is not explicitly mentioned in the sentence. In such cases, the annotators were asked to annotate the associated entities *"the wind"* and *"a bird"* as representing variables.

The guidelines also directed annotators to choose the longest possible mention, in case they feel the mention boundary is ambiguous. As a result, in the sentence, *"City Rentals rent an intermediate-size car for 18.95 dollars plus 0.21 per mile."*, the phrase *"City Rentals rent an intermediate-size car"* was annotated as representing variable. We allow multiple mentions to be annotated for the same variable. In example 2 (Section 1), both *"a number"* and *"the same number"* were annotated as representing the same variable.

References

[Artzi and Zettlemoyer2013] Yoav Artzi and Luke Zettlemoyer. 2013. UW SPF: The University of Washington Semantic Parsing Framework.

[Björkelund and Kuhn2014] Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents

and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[Cai and Yates2013] Qingqing Cai and Alexander Yates. 2013. Semantic Parsing Freebase: Towards Open-domain Semantic Parsing. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

[Chang et al.2015] Kai-Wei Chang, Shyam Upadhyay, Ming-Wei Chang, Vivek Srikumar, and Dan Roth. 2015. Illinoissl: A JAVA library for structured prediction. In *Arxiv Preprint*, volume abs/1509.07179.

[Clarke et al.2010] J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, 7.

[Goldwasser and Roth2011] D. Goldwasser and D. Roth. 2011. Learning from natural instructions. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[Hosseini et al.2014] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*.

[Kushman et al.2014] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*.

[Kwiatkowski et al.2013] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

[Madaan et al.2016] A. Madaan, A. Mittal, Mausam, G. Ramakrishnan, and S. Sarawagi. 2016. Numerical relation extraction with minimal supervision. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

[Punyakanok et al.2005] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129.

[Roth and Zelenko1998] D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics*, pages 1136–1142.

[Roy and Roth2015] S. Roy and D. Roth. 2015. Solving general arithmetic word problems. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Roy et al.2015] S. Roy, T. Vieira, and D. Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3.

[Socher et al.2013] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. In *ACL*.

[Sutton and McCallum2007] C. Sutton and A. McCallum. 2007. Piecewise pseudolikelihood for efficient training of conditional random fields. In Zoubin Ghahramani, editor, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 863–870. Omnipress.

[Wong and Mooney2007] Y.-W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

Automatic Extraction of Implicit Interpretations from Modal Constructions

Jordan Sanders and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
jordansanders3@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents an approach to extract implicit interpretations from modal constructions. Importantly, our approach uses a deterministic procedure to normalize eventualities and generate potential interpretations. An annotation effort demonstrates that these interpretations are intuitive to humans and most modal constructions convey at least one interpretation. Experimental results show that the task is challenging but can be automated.

1 Introduction

People use language to communicate not only facts, but also intentions, uncertain information and points of view. Modality can be broadly defined as a grammatical phenomenon used to express the speaker's opinion or attitude towards a proposition (Lyons, 1977). Modality has also been defined as "the category of meaning used to talk about possibilities and necessities, essentially, states of affairs beyond the actual." (Hacquard, 2011). Within computational linguistics, processing modality has proven useful for, among others, recognizing textual entailment (Snow et al., 2006; MacCartney et al., 2006), machine translation (Murata et al., 2005; Baker et al., 2012), and sentiment analysis (Wiebe et al., 2005).

In the absence of modality markers, it is understood that the author of a proposition agrees with it (Hengeveld and Mackenzie, 2008). Adding a modality marker—also referred to as cue—casts doubt on the truth of the proposition, e.g., *Mary got a new job last week* vs. *Mary likely got a new job last week*. Modality is surprisingly common (Morante and Sporleder, 2012), and notoriously difficult to annotate and process automatically (Rubinstein et al., 2013; Vincze et al., 2011). In MEDLINE, 11% of sentences contain speculative language (Light et al., 2004) and in biomedical abstracts, 18% (Vincze et al., 2008). Rubin (2006) reports that 59% of statements in 80 New York Times articles include epistemic modality. Despite modality being ubiquitous, there is not an agreed upon annotation schema.

In this paper, we extract implicit interpretations intuitively understood by humans when reading modal constructions. We do not follow any specific theory of modality. Instead, we manipulate modal constructions to automatically generate potential interpretations, and then assign factuality scores to them. Consider statement (1) below:

1. John likely contracted the disease when a mouse bit him in the Adirondacks.

Even though *likely* syntactically attaches to *contracted*, a natural reading suggests that *John contracted the disease* is factual; the only bit of uncertain information is how (or when) he contracted the disease. In other words, assuming that the author of statement (1) is truthful, event *contracted* occurred with AGENT *John* and THEME *the disease*, but the MANNER (or TIME) may not have been *when a mouse bit him in the Adirondacks*.

A key feature of the work presented in this paper is that the interpretations extracted from modal constructions are not tied to any syntactic or semantic representation. Given modal constructions in plain text, we extract implicit interpretations in plain text, and these interpretations can be processed with any existing NLP pipeline. The main contributions of

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1098–1107,
Austin, Texas, November 1-5, 2016.

this paper are: (1) procedure to automatically generate potential interpretations from modal constructions; (2) annotations assessing the factuality of potential interpretations generated from OntoNotes;[1] and (3) experimental results using several features.

2 Previous Work

Theoretical works in philosophy and linguistics have studied modality for decades (Palmer, 2001; Jespersen, 1992). Morante and Sporleder (2012) summarize some of these works and related phenomena, e.g., evidentiality, certainty, factuality, subjectivity. There are several expressions that have modal meanings (Fintel, 2006), including auxiliaries (must, should, etc.), adverbs (perhaps, possibly, etc.) nouns (possibility, chance, etc.) adjectives (necessary, possible, etc.) and conditionals (e.g., If the light is on, Sandy is home). Most previous works in computational linguistics target modal adverbs (Rubinstein et al., 2013; Carretero and Zamorano-Mansilla, 2013; de Waard and Maat, 2012), and some also target other modal triggers such as reporting verbs (e.g., The evidence *suggests* that he caused the fire), references, or all verbs (Diab et al., 2009). Following these previous works, we focus on modal adverbs.

Beyond theoretical works, there are many proposals to annotate modality. Doing so has proven challenging: following different annotations schemas on the same source text yields little overlap (Vincze et al., 2011), and Carretero and Zamorano-Mansilla (2013) present an analysis of disagreements when targeting modal adverbs. Annotation schemas typically include 3 tasks: identifying modality triggers, their scopes, and sources (Quaresma et al., 2014; Sánchez and Vogel, 2015). Many also classify the modality into several types (epistemic, circumstantial, ability, deontic, etc.) or a fine-grained taxonomy (Rubinstein et al., 2013; Nissim et al., 2013). In this paper, we are not concerned with modeling modality per se, or classifying instances of modality into predefined classes or hierarchies. Instead, we extract implicit interpretations from modal constructions in order to mirror intuitive readings.

FactBank is probably the best-known corpus for event factuality (Saurí and Pustejovsky, 2009). It was created following carefully crafted annotation guidelines and examples comprising 34 pages.[2] The guidelines detail a manual normalization step to "identify the full event that needs to be assessed in terms of its factuality" (p. 12), and the annotation process includes identifying the sources that are assessing factuality (p. 15). de Marneffe et al. (2012) reannotate a subset of FactBank with factuality values from the reader's perspective—they call it veridicality—using crowdsourcing. Both FactBank and de Marneffe et al. (2012), rely on manual normalization to identify the eventuality whose factuality is being annotated. Instead, we present an automated approach: we manipulate semantic roles and syntactic dependencies deterministically to generate several potential interpretations per modal construction, and then assess their factuality.

Many other efforts expand on FactBank using crowdsourced annotations, different annotation schemas (usually simpler) or other domains. Prabhakaran et al. (2012) use crowdsourcing to classify propositions into 5 modalities: ability, effort, intention, success and want. Soni et al. (2014) target the factuality of quotes (direct and indirect) in Twitter. Lee et al. (2015) detect events and assess factuality using easy-to-understand short instructions to crowdsource annotations. Unlike us, they annotate factuality at the individual token level, where annotated tokens are deemed events by annotators. Prabhakaran et al. (2015) define and annotate propositional heads with four categories: (1) non-belief propositions, or (2) committed, non-committed or reported belief. Instead of assessing factuality only for propositional heads (usually verbs, one assessment per proposition), we do so for potential interpretations automatically generated by manipulating verbs and their arguments deterministically.

All works cited in the previous two paragraphs either manually normalize text prior to assessing factuality—making automation from plain text impossible—or assess factuality for tokens deemed events (ordered, delay, agreed, etc.) or full propositions (a verb and all its arguments). Unlike them, we automatically generate potential interpretations from a single modal construction—or, equivalently, automatically generate several normalizations—and then assess their factuality.

[1] Available at www.sanders.tech

[2] https://catalog.ldc.upenn.edu/docs/LDC2009T23/annotationGuidelines.pdf

3 Terminology and Background

We use the term *modal construction* to refer to verb-argument structures modified by a modal adverb (possibly, probably, etc.). We use the term *implicit interpretation*, or *interpretation* to save space, to refer to meaning intuitively understood by humans when reading a modal construction. *Potential interpretations* are interpretations automatically generated whose factuality has yet to be determined. The *factuality* of an interpretation is a score indicating its likelihood—whether it is true, false or unknown given the modal construction.

We work on top of OntoNotes (Hovy et al., 2006) because it includes text from several genres (news, broadcast and telephone conversations, weblogs, etc.) and includes part-of-speech tags, parse trees, PropBank-style semantic roles and other linguistic information.[3] Very briefly, PropBank (Palmer et al., 2005) has two kinds of semantic roles: numbered roles (ARG$_0$, ARG$_1$, etc.), which are defined in verb-specific framesets, and argument modifiers (ARGM-TMP, ARGM-LOC, etc.), we refer the reader to the aforementioned reference, and the guidelines and framesets[4] for more details. We transformed the parse trees in OntoNotes into syntactic dependencies using Stanford CoreNLP (Manning et al., 2014).

4 Corpus Creation

We define a two-step procedure to create a corpus of modal constructions and the implicit interpretations intuitively understood by humans when reading them. First, we automatically generate potential interpretations from modal constructions by manipulating syntactic dependencies and semantic roles. Second, we manually score potential interpretations according to their likelihood. These interpretations and scores are later used to learn how to score potential interpretations automatically (Section 6).

4.1 Generating Potential Interpretations

Selecting Modal Constructions. OntoNotes is a large corpus containing 63,918 sentences. Creating a corpus of interpretations for all modal constructions is outside the scope of this paper. In order to alleviate the annotation effort, we focus on selected modal constructions. Specifically, we select verb-argument structures that have one ARGM-ADV or ARGM-MNR role, and that role is one of the following modal adverbs: certainly, clearly, definitely, likely, obviously, possibly, probably, surely, or unlikely. These adverbs are the most frequent that satisfy the above filter. Additionally, we discard verb-argument structures with *to be* as the main verb. These rules retrieve 324 modal constructions.

Automatic Normalization. Modal constructions often occur in long multi-clause sentences. In order to identify the eventuality from which potential interpretations should be generated, we automatically normalize the original sentence. Normalizing consists of a battery of deterministic steps implemented using syntactic dependencies and semantic roles. In contrast with previous work (Section 2), our normalization is fully automated. Hereafter, we use *verb* to refer to the main verb in the modal construction, *adverb* to the modal adverb, and *sem_roles* to all semantic roles in the modal construction.

1. Remove *adverb*.
2. Convert negated verb-argument structures into their positive counterparts. We follow 3 steps inspired by the rules to form negation proposed by (Huddleston and Pullum, 2002):
   (a) Remove the negation mark by deleting the token whose syntactic dependency is *neg*.
   (b) Remove auxiliaries, expand contractions, and fix third-person singular and past tense. For example (before: after), *doesn't go*: *goes*, *didn't go*: *went*, *won't go*: *will go*. To implement this step, we loop through tokens whose head is the negated verb with dependency *aux*, and use a list of irregular verbs[5] and grammar rules to convert to third-person singular and past tense based on orthographic patterns.
   (c) Rewrite negatively-oriented polarity-sensitive items. For example (before: after), *anyone*: *someone*, *any longer*: *still*, *yet*: *already*. *at all*: *somewhat*. We use the correspondences between negatively-oriented and positively-

[3] We use the CoNLL-2011 Shared Task distribution (Pradhan et al., 2011), http://conll.cemantix.org/2011/

[4] http://propbank.github.io/

[5] https://en.wikipedia.org/wiki/English_irregular_verbs

Sent. 1: The danger is [probably]$_{\text{ARGM-ADV}}$ [he]$_{\text{ARG}_0}$ [can]$_{\text{ARGM-MOD}}$ [not]$_{\text{ARGM-NEG}}$ [deliver]$_{verb}$ [the promises that he made during the campaign.]$_{\text{ARG}_1}$		
Normalization	Step	Output
	1	The danger is he cannot deliver the promises that he made during the campaign.
	2	The danger is he can deliver the promises that he made during the campaign.
	3	The danger is he will deliver the promises that he made during the campaign.
	4	He will deliver the promises that he made during the campaign.
	5	He will deliver the promises that he made during the campaign.
Sent. 2: [...] I wouldn't define victory as simply not raising taxes—although [I]$_{\text{ARG}_0,\, v_1,\, v_2}$ [definitely]$_{\text{ARGM-ADV},\, v_1}$ [would]$_{\text{ARGM-MOD},\, v_1}$ [like]$_{v_1}$ [to [defer]$_{v_2}$ [raising taxes]$_{\text{ARG}_1,\, v_2}$ [as long as prudently possible.]$_{\text{ARG}_2,\, v_2}$]$_{\text{ARG}_1,\, v_1}$		
Normalization	Step	Output
	1	I wouldn't define [...] although I would like to defer raising taxes as long as prudently possible.
	2, 3	I would define [...] although I will like to defer raising taxes as long as prudently possible.
	4	I will like to defer raising taxes as long as prudently possible.
	5	Normalization 1: I will like to defer raising taxes as long as prudently possible. Normalization 2: I will defer raising taxes as long as prudently possible.
Interpretations	From	Potential Interpretation
	norm. 1	$\{\text{ARG}_0\}$ will like to defer raising taxes as long as prudently possible.
		I will like $\{$to $\text{ARG}_1\}$.
	norm. 2	$\{\text{ARG}_0\}$ will defer raising taxes as long as prudently possible.
		I will defer $\{\text{ARG}_1\}$ as long as prudently possible.
		I will defer raising taxes $\{\text{ARG}_2\}$.
		I will defer $\{\text{ARG}_1\}$ $\{\text{ARG}_2\}$.

Table 1: Step-by-step execution of the procedure to automatically normalize modal constructions (Sentences 1 and 2) and generate potential interpretations (Sentence 2).

oriented polarity-sensitive items by (Huddleston and Pullum, 2002, pp. 831).

3. Fix modal verbs and tense. If a modal verb (can, could, may, would, should, must, etc.) has as syntactic head *verb*, we transform the modal construction into past or future depending on the modal and tense of *verb*. For example: *could go: went*, *can go: will go*, *should have gone: went*. We use the same grammar rules and list of irregular verbs as in Step (2b).
4. Select relevant tokens. We remove all tokens in the original sentence except *verb* and tokens belonging to the roles in *sem_roles*. Additionally, we fix phrasal verbs by adding tokens with the part-of-speech tag RP whose syntactic head is *verb* and dependency type *prt* (semantic roles in OntoNotes are annotated for verb tokens, missing the preposition when *verb* is a phrasal verb would inadvertently change meaning). We also add all tokens to the left of *verb* until we find the first token whose part-of-speech tag does not start with VB, MD, RB or EX (verbs, modals, adverbs and existential *there*).
5. Generate additional normalizations. If *verb* is followed by TO + $verb_2$ (e.g., want to go, like to play, intend to pass), we generate an additional normalization for $verb_2$ after merging the semantic roles of *verb* and $verb_2$.

Table 1 exemplifies the automatic normalization step by step with 2 modal constructions.

Generating Potential Interpretations in Plain Text. Inspired by the rules Blanco and Sarabi (2016) used to generate interpretations from negation, we generate potential interpretations from modal constructions by toggling off combinations of roles in *sem_roles*. We consider numbered roles (ARG_0–ARG_5), and argument modifiers (ARGM-) ending in LOC, TMP, MNR, PRP, CAU, EXT, PRD or DIR.

Table 1 lists some potential interpretations generated from a sample modal construction. The total number of potential interpretations for the 324 selected modal construction is 1,756 (average: 5.4).

We recognize that our procedure to generate implicit interpretations is unable to generate some useful interpretations. For example, from *This is [a person who]*$_{\text{ARG}_1}$ *[likely]*$_{\text{ARGM-ADV}}$ *[died]*$_{verb}$ *[on impact versus perhaps freezing to death]*$_{\text{ARGM-MNR}}$, we

generate *This is a person who died* {ARGM-MNR}, which is factual: the only uncertain information is the manner in which the person died. Since we toggle off semantic roles of *verb*, our procedure is unable to generate *A person died on impact* and *A person died freezing to death*; the former interpretation would receive a higher factuality score than the latter. We argue that automation is preferable, and reserve for future work generating interpretations that require splitting semantic roles.

4.2 Scoring Potential Interpretations

After automatically generating potential interpretations, we collected manual annotations to determine their factuality. The annotation interface showed the original sentence containing the modal construction, the previous and next sentences as context, and no additional information. Following previous work (Saurí and Pustejovsky, 2009; de Marneffe et al., 2012), we found it useful not to restrict answers to *yes* or *no*, but to allow for degrees of certainty. Specifically, we asked "Given the 3 sentences above, do you believe that the statement [potential interpretation] below is true?". Answers are a score ranging from -5 to 5, where -5 indicates *Certainly no*, 5 indicates *Certainly yes*, and the scores in between indicate a continuum of certainty (0 indicates *unknown*).

After pilot annotations, we examined disagreements and defined the following simple guidelines:

1. Context (previous sentence, target sentence, and next sentence) is taken into account.
2. World knowledge available at the time the original sentence was authored—not new knowledge available after—is taken into account.
3. Semantic roles toggled off are replaced with a semantically related substitute (Turney and Pantel, 2010) for the original role, e.g., give: take, customer: sales associate.

5 Corpus Analysis

The total number of modal constructions selected is 324 and the number of potential interpretations automatically generated in 1,756 (average: 5.4 interpretation per modal construction). 39.4% of interpretations are scored with a high degree of certainty. We define *high certainty* as a score below -3 (interpretation is false) or larger than 3 (interpretation is true). Importantly, on overage, modal constructions have 2.13 interpretations scored with high certainty, and 1.23 scored 3 or higher. In other words, on average, our procedure generates over 2 interpretation that are either true or false, and over 1 interpretation that is true per modal construction.

# roles toggled off	#	%≠ 0	Mean score	
			> 0	< 0
0	345	87.25	3.96	-3.94
1	800	48.50	3.67	-3.90
2	479	20.46	3.55	-4.03
3	120	5.83	3.50	-3.00

Table 2: Number of interpretations generated by toggling off 0, 1, 2 or 3 roles (#), percentage of interpretations not scored zero (% $\neq$ 0), and mean scores of interpretations with positive and negative scores.

Role	#	% ≠ 0	Mean score	
			> 0	< 0
None	345	87.25	3.96	-3.94
ARG_1	671	30.40	3.60	-3.92
ARG_0	604	25.50	3.72	-3.94
ARG_2	140	28.57	3.85	-3.84
ARGM-MNR	271	32.84	3.40	-3.85
ARGM-TMP	231	28.57	3.71	-3.84
ARGM-LOC	82	23.17	3.43	-4.60
Other	290	20.00	3.38	-3.87

Table 3: Number of interpretations generated by toggling off each semantic role (#), percentage of interpretations not scored zero (% $\neq$ 0), and mean score of interpretations with positive and negative scores.

Tables 2 and 3 present basic corpus statistics. The percentage of interpretations annotated with a score different than 0 depends greatly on the number of roles toggled off (Table 2): 0: 87.25%, 1: 48.50%, 2: 20.46%, 3: 5.83%. Note that the number of roles toggled off does not significantly affect the mean score of interpretations not scored 0 (Table 2, last 2 columns). Most interpretations have either ARG_0 or ARG_1 toggled off (Table 3), and the percentages of interpretations not scored zero range from 20% to 32.84% depending on the semantic role. Note that the average score of interpretations scored positively and negatively, however, does not depend on whether a semantic role is toggled off.

	Original sentence and sample of automatically generated potential interpretations	Score
1	Context, previous sentence: *The last thing we want to do is react to every wild statement that they make.* Original sentence: *[But]*$_{\text{ARGM-DIS}}$ *[they]*$_{\text{ARG}_0}$ *[certainly]*$_{\text{ARGM-ADV}}$ *[chose]*$_{\textit{verb}}$ *[that]*$_{\text{ARG}_1}$ *[to get our attention and that of the international community.]*$_{\text{ARGM-PRP}}$ Context, next sentence: *Uh but what they've got to realize is there is no magic bullet here.*	
	- Interpretation 1.1: But they chose that to get our attention and that of the international community.	5
	- Interpretation 1.2: But they chose $\{\text{ARG}_1\}$ to get our attention and that of the international community.	-5
2	Context, previous sentence: *Saddam Hussein (interrupting): Before you offer me your rotten goods, I ask you did you find weapons of mass destruction in Iraq or not?* Original sentence: *Rumsfeld (disconcerted): We haven't found them yet, but [we]*$_{\text{ARG}_0}$ *[will]*$_{\text{ARGM-MOD}}$ *[surely]*$_{\text{ARGM-ADV}}$ *[find]*$_{\textit{verb}}$ *[them]*$_{\text{ARG}_1}$ *[one day]*$_{\text{ARGM-TMP}}$. Context, next sentence: *Do you deny that you had intentions to manufacture a nuclear bomb?*	
	- Interpretation 2.1: We will find them one day.	4
	- Interpretation 2.2: We will find them $\{\text{ARGM-TMP}\}$.	-3
3	*"This is a rare case of [a company with a big majority holder which]*$_{\text{ARG}_0}$ *[will]*$_{\text{ARGM-MOD}}$ *[probably]*$_{\text{ARGM-ADV}}$ *[act]*$_{\textit{verb}}$ *[in the interests of the minority holders]*$_{\text{ARG}_1}$*", one investor says.*	
	- Interpretation 3.1: $\{\text{ARG}_0\}$ will act in the interests of the minority holders.	4
	- Interpretation 3.2: A company with a big majority holder will act $\{\text{ARG}_1\}$.	4
4	*I wouldn't define victory as simply not raising taxes—although [I]*$_{\text{ARG}_0,\, v_1,\, v_2}$ *[definitely]*$_{\text{ARGM-ADV},\, v_1}$ *[would]*$_{\text{ARGM-MOD},\, v_1}$ *[like]*$_{v_1}$ *[to [defer]*$_{v_2}$ *[raising taxes]*$_{\text{ARG}_1,\, v_2}$ *[as long as prudently possible.]*$_{\text{ARG}_2,\, v_2}$*]*$_{\text{ARG}_1,\, v_1}$	
	- Interpretation 4.1: I will like to defer raising taxes as long as prudently possible.	5
	- Interpretation 4.2: I will defer raising taxes as long as prudently possible.	1

Table 4: Annotation Examples. For each example, we show the original sentence containing the modal construction, context if helpful to determine scores, and 2 selected interpretations and their scores. Square brackets indicate semantic roles.

5.1 Annotation Quality

The annotation guidelines (Section 4.2) to score potential interpretations were defined after examining disagreements in pilot annotations. After defining the guidelines, inter-annotator agreement was 0.92 on 18% of randomly selected interpretations.[6] Agreement measures designed for categorical labels are unsuitable, as not all disagreements are equal, e.g., 4 vs. 5, -2 vs. 5. Because of the high agreement and following previous work (Agirre et al., 2012), the rest of interpretations were annotated once.

5.2 Annotation Examples

Table 4 presents annotation examples. For each example, we include the original sentence containing a selected modal construction, its context (previous and next sentence) if helpful for scoring, and 2 automatically generated potential interpretations with their annotated scores.

Example (1) shows that context helps in determining the factuality of potential interpretations (item (1) in the guidelines). After reading the three sentences, it is clear that *they* are making *wild statements*, and are hoping to get *attention* for it. Interpretation 1.1 removes adverb *certainly* and receives the highest score, 5. Interpretation 1.2 is obtained after toggling off ARG_1, and receives the lowest score, -5. This low score is justified by item (3) in our annotation guidelines: replacing *wild statements* with a semantically (different but) related substitute, e.g., *But they chose reasonable statements / good manners to get our attention and that of the international community*, yields an unlikely interpretation.

The interpretations in Example (2) show again the importance of context, and also exemplify item (2) in the annotation guidelines. Interpretation 2.1, *We will find them one day* receives a high score $(4/5)$, as given the context (and assuming that Rumsfeld is truthful), it is very likely that they will find the weapons of mass destruction, but it is not guaranteed. Note that annotators are not allowed to use the fact that the weapons were never found (item (2) in the guidelines). In Interpretation 2.2, *one day* could be replaced with *never / at no time* or similar constructions, and doing so yields the opposite of the intended meaning (score: -3). A possible descrip-

[6] We set an internal deadline of 3 days after agreeing on the guidelines, and we could annotate 18% of instances in that time.

Type	Feature	Description
baseline	adverb	Word form of adverb
	adverb_pos	Part-of-speech of adverb
	verb	Word form of verb
	verb_pos	Part-of-speech of verb
	distance	Number of tokens between adverb and verb
	direction	Whether adverb occurs before or after verb
adverb and verb	adverb_rel_pos	Part-of-speech tags of the parent, and left and right siblings of adverb
	adverb_subcat	Concatenation of part-of-speech tags of all siblings of adverb
	verb_rel_pos	Part-of-speech tags of the parent, and left and right siblings of verb
	adverb_subcat	Concatenation of part-of-speech tags of all siblings of verb
	path, path_l	Syntactic path between adverb and verb, and length of the path
	ancestor	POS tag of the lowest common ancestor between verb and adverb
	has_sem_role	Flags indicating whether a semantic role is in the modal construction
interpretation	num_roles_int	Number of roles toggled off in the potential interpretation
	sem_roles_int	Flags indicating which roles are toggled off in the interpretation
	roles_distance	Number of tokens between each semantic role and verb
	roles_direction	Whether each semantic role occurs before or after verb
	roles_path	Syntactic path between each role and verb
	roles_path_l	Length of syntactic path between each role and verb

Table 5: Features used to predict factuality scores to automatically generated potential interpretations. Features extracted from semantic role are extracted for ARG_0–ARG_5 and modifiers (ARGM-) ending in LOC, TMP, MNR, PRP, CAU, EXT and PRD.

tion of these scores could be "almost certainly true" (4 out of 5), and "most probably false" (-3 out of -5). We see scores as a continuum of certainty, but textual description may help understand the examples.

Example (3) demonstrates the usefulness of the normalization process—specifically, Step 4, selecting relevant tokens—and the importance of replacing roles with semantically related substitutes (item (3) in the guidelines). In interpretation 3.1, $\{\text{ARG}_0\}$ *will act in the interests of the minority holders*, ARG_0 can be replaced with *a company with several minority holders*, yielding a valid interpretation scored 4 (out of 5). Similarly, in interpretation 3.2, *A company with a big majority holder will act* $\{\text{ARG}_1\}$, ARG_1 can be replaced with *in the interests of the big majority holder*, yielding another valid interpretation also scored 4 (out of 5).

Finally, Example (4) shows Step 5 in the automatic normalization procedure (Section 4). By creating an additional verb-argument structure, we are able to differentiate between liking to do something (Interpretation 4.1, score 5/5) and actually doing that something (Interpretation 4.2, score 1/5).

6 Learning to Score Potential Interpretations

In order to automatically score potential interpretations, we follow a standard supervised machine learning approach. Each potential interpretation becomes an instance, and we split modal constructions (and their potential interpretations) into training (80%) and test (20%). When splitting, we make sure that the amount of modal constructions for each adverb in each split is proportional, i.e., 80% of modal constructions with each adverb are in the train split and the rest in the test split. Splitting instances randomly would assign interpretations generated from the same modal construction to the train and test splits, and bias the results.

We trained a Support Vector Machine (SVM) for regression with RBF kernel using scikit-learn (Pedregosa et al., 2011), which uses LIBSVM (Chang and Lin, 2011). The SVM parameters (C and γ) were tuned using 10-fold cross-validation with the training set, and we report results using the test split.

Features	Pearson
baseline	-0.029
adverb and verb	0.025
interpretation	0.494
baseline + adverb and verb	-0.013
baseline + interpretation	0.463
adverb and verb + interpretation	0.465
baseline + adverb and verb + interpretation	0.468

Table 6: Pearson correlations obtained with test instances and several feature combinations.

6.1 Feature Selection

The full set of features is detailed in Table 5. *Baseline* features are simple features characterizing *adverb* and *verb* and we do not elaborate on them. *Adverb and verb* features are extracted from the modal construction (constituent tree and semantic roles) and provide additional information about the modal construction. *Interpretation* features characterize the potential interpretation whose factuality is being scored, and are also derived from the constituent tree and semantic roles.

Most *adverb and verb* features are standard in semantic role labeling (Gildea and Jurafsky, 2002). We include the part-of-speech tags of the parent, and left and right siblings of *adverb* and *verb*, as well as their subcategorization, i.e., the concatenation of the sibling's part-of-speech tags. We also include syntactic path between *adverb* and *verb*, and its length. Additionally, we include the common ancestor, i.e., the syntactic node of the lowest common node that is an ancestor of both *adverb* and *verb*, and use binary features to indicate whether each semantic role is present in the modal construction.

Finally, *interpretation* features characterize the semantic roles toggled off to generate the potential interpretation. We include the number of roles toggled off to generate the potential interpretation, and binary flags indicating which roles. Additionally, for each role toggled off, we include the distance from the verb (number of tokens), whether it occurs before or after the verb, the syntactic path to the verb and the length of the path.

7 Experimental Results

Table 6 details results obtained with test instances using several feature combinations derived from gold linguistic information (POS tags, parse trees, semantic roles, etc.). *Baseline* and *adverb and verb* features, which characterize the modal construction from which potential interpretation are extracted, are virtually useless. They yield Pearson correlations of -0.029 and 0.025 individually, and -0.013 combined. These results suggest that the verb and adverb in the modal construction (word forms, syntactic paths, etc.) are insufficient to rank potential interpretations generated from the modal construction.

Interpretation features, which capture differences between potential interpretations being scored (number of roles toggled off, roles toggled off, etc.), obtain a modest Pearson correlation of 0.494. Combining *interpretation* features with other features proved detrimental, Pearson correlations are between 0.463 and 0.468.

8 Conclusions

Modality is a pervasive phenomenon used to talk about what is not factual. In this paper, we have presented a methodology to extract implicit interpretations from modal constructions. First, we automatically generate potential interpretations using syntactic dependencies and semantic roles, and then assign to them a factuality score.

The most important conclusion of the work presented here is that several interpretations automatically generated from a single modal construction often receive scores indicating high certainty. Indeed, on average, modal constructions have 2.13 interpretations scored lower or equal than -3, or higher or equal than 3. This contrast with previous work, which only assess factuality of one normalization per proposition.

Experimental results using supervised machine learning and relatively simple features show that the task is challenging but can be automated. We believe better results could be obtained by incorporating features capturing knowledge in the context of the modal construction, including other clauses in the same sentence, and the previous and next sentences. Another extension of the current work is to investigate a similar approach for other modality markers such as nouns (e.g., possibility, chance), adjectives (e.g.necessary, probable,) and certain verbs (e.g., claim, suggests).

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June.

Kathryn Baker, Michael Bloodgood, Bonnie J. Dorr, Chris Callison-Burch, Nathaniel W. Filardo, Christine Piatko, Lori Levin, and Scott Miller. 2012. Use of Modality and Negation in Semantically-Informed Syntactic MT. *Comput. Linguist.*, 38(2):411–438, June.

Eduardo Blanco and Zahra Sarabi. 2016. Automatic generation and scoring of positive interpretations from negated statements. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1431–1441, San Diego, California, June. Association for Computational Linguistics.

Marta Carretero and Juan Rafael Zamorano-Mansilla. 2013. An analysis of disagreement-provoking factors in the analysis of epistemic modality and evidentiality: the case of english adverbials. In *Proceedings of IWCS 2013 Workshop on Annotation of Modal Meanings in Natural Language (WAMM)*, pages 16–23, Potsdam, Germany, March.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Comput. Linguist.*, 38(2):301–333, June.

Anita de Waard and Henk Pander Maat. 2012. Epistemic modality and knowledge attribution in scientific discourse: A taxonomy of types and overview of features. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, ACL '12, pages 47–55, Stroudsburg, PA, USA.

Mona Diab, Lori Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed belief annotation and tagging. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 68–73, Suntec, Singapore, August.

Kai Von Fintel. 2006. Modality and language. In D. Borchert, editor, *Encyclopedia of Philosophy*, pages 20–27. Macmillan Reference.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September.

Valentine Hacquard. 2011. Modality. In C. Maienborn, K. von Heusinger, and P. Portner, editors, *Semantics: An International Handbook of Natural Language Meaning*, pages 1484–1515. Mouton de Gruyter.

Kees Hengeveld and J. Lachlan Mackenzie. 2008. *Functional Discourse Grammar: A Typologically-Based Theory of Language Structure*. Oxford University Press.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, April.

Otto Jespersen. 1992. *The philosophy of grammar*. University of Chicago Press, Chicago.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1648, Lisbon, Portugal, September.

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In Lynette Hirschman and James Pustejovsky, editors, *HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, pages 17–24, Boston, Massachusetts, USA, May 6.

John Lyons. 1977. *Semantics*. Cambridge University Press. Cambridge Books Online.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 41–48, Stroudsburg, PA, USA.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Roser Morante and Caroline Sporleder. 2012. Modality and negation: An introduction to the special issue. *Comput. Linguist.*, 38(2):223–260, June.

Masaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Hitoshi Isahara, and Qing Ma. 2005. Correction of errors in a verb modality corpus for machine translation with a machine-learning method. 4(1):18–37, March.

Malvina Nissim, Paola Pietrandrea, Andrea Sanso, and Caterina Mauri. 2013. Cross-linguistic annotation of

modality: a data-driven hierarchical model. In *Proceedings of the 9th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 7–14, Potsdam, Germany, March.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

F. R. Palmer. 2001. *Mood and Modality*. Cambridge University Press, second edition. Cambridge Books Online.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Vinodkumar Prabhakaran, Michael Bloodgood, Mona Diab, Bonnie Dorr, Lori Levin, Christine D. Piatko, Owen Rambow, and Benjamin Van Durme. 2012. Statistical modality tagging from rule-based annotations and crowdsourcing. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 57–64, Jeju, Republic of Korea, July.

Vinodkumar Prabhakaran, Tomas By, Julia Hirschberg, Owen Rambow, Samira Shaikh, Tomek Strzalkowski, Jennifer Tracey, Michael Arrigo, Rupayan Basu, Micah Clark, Adam Dalton, Mona Diab, Louise Guthrie, Anna Prokofieva, Stephanie Strassel, Gregory Werner, Yorick Wilks, and Janyce Wiebe. 2015. A new dataset and evaluation for belief/factuality. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 82–91, Denver, Colorado, June.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA, June.

P. Quaresma, A. Mendes, I. Hendrickx, and T. Gon?alves. 2014. Automatic tagging of modality: identifying triggers and modal value. In *Proceedings of the 10th Joint ACL SIGSEM - ISO Workshop on Interoperable Semantic Annotation*.

Victoria L. Rubin. 2006. *Identifying certainty in texts*. Ph.D. thesis, Syracuse University, Syracuse, NY.

Aynat Rubinstein, Hillary Harner, Elizabeth Krawczyk, Daniel Simonson, Graham Katz, and Paul Portner. 2013. Toward fine-grained annotation of modality in text. In *Proceedings of IWCS 2013 Workshop on Annotation of Modal Meanings in Natural Language (WAMM)*, pages 38–46, Potsdam, Germany, March.

Liliana Mamani Sánchez and Carl Vogel. 2015. A hedging annotation scheme focused on epistemic phrases for informal language. In *Proceedings of the Workshop on Models for Modality Annotation*.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Rion Snow, Lucy Vanderwende, and Arul Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 33–40, Stroudsburg, PA, USA.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling factuality judgments in social media text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 415–420, Baltimore, Maryland, June.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

Veronika Vincze, György Szarvas, Richard Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9((Suppl 11)):S9.

Veronika Vincze, György Szarvas, György Móra, Tomoko Ohta, and Richárd Farkas. 2011. Linguistic scope-based and biological event-based speculation and negation annotations in the bioscope and genia event corpora. *Journal of Biomedical Semantics*, 2(5):1–11.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210.

Understanding Negation in Positive Terms Using Syntactic Dependencies

Zahra Sarabi and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
zahrasarabi@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a two-step procedure to extract positive meaning from verbal negation. We first generate potential positive interpretations manipulating syntactic dependencies. Then, we score them according to their likelihood. Manual annotations show that positive interpretations are ubiquitous and intuitive to humans. Experimental results show that dependencies are better suited than semantic roles for this task, and automation is possible.

1 Introduction

Negation is a complex phenomenon present in all human languages, allowing for the uniquely human capacities of denial, contradiction, misrepresentation, lying, and irony (Horn and Wansing, 2015). Despite negation always being marked—in the absence of a negation cue, statements are positive—acquiring and understanding sentences that contain negation is more challenging than those that do not. Children acquire negation after learning to communicate (Nordmeyer and Frank, 2013), and adults take longer to process negated statements than positive ones (Clark and Chase, 1972).

In any given language, humans communicate in positive terms most of the time, and use negation to express something unusual or an exception (Horn, 1989). Albeit most sentences are affirmative, negation is ubiquitous (Morante and Sporleder, 2012): In scientific papers, 13.76% of statements contain a negation (Szarvas et al., 2008); in product reviews, 19% (Councill et al., 2010); and in Conan Doyle stories, 22.23% (Morante and Daelemans, 2012). In OntoNotes (Hovy et al., 2006), 10.15% of statements contain a verb negated with *not*, *n't* or *never*.

From a theoretical point of view, it is accepted that negation conveys positive meaning (Rooth, 1992; Huddleston and Pullum, 2002). For example, when reading (1) *John didn't order the right parts*, humans intuitively understand that (1a) *John ordered something*, or more specifically, (1b) *John ordered the wrong parts*. Interpretation (1a) can be obtained after determining that *n't* does not negate verb *order*, but its THEME, i.e., *the right parts*. Interpretation (1b) can be obtained after determining that *n't* is actually negating *right*, an adjective modifying *parts*.

Determining which words are intended to be negated—identifying the foci of negation, thereby revealing positive interpretations—is challenging. First, as exemplified in (1a, 1b), there is a granularity continuum yielding interpretations that entail each other, e.g., (1b) entails (1a). Second, a single negation often yields several positive interpretations, e.g., from (2) *John doesn't eat meat*, we can extract that (2a) *John eats something other than meat* and (2b) *Some people eat meat, but not John*.

This paper presents a methodology to extract positive interpretations from verbal negation. The main contributions are: (1) deterministic procedure to generate potential interpretations by manipulating syntactic dependencies; (2) analysis showing that dependencies yield finer-grained interpretations and better results than previous work using semantic roles; (3) a corpus of negations and their positive interpretations;[1] and (4) experimental results with gold-standard and predicted linguistic information.

[1] Available at http://www.cse.unt.edu/~blanco/

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1108–1118,
Austin, Texas, November 1-5, 2016.

2 Terminology, Scope and Focus

Negation is well-understood in grammars, which detail the valid ways to form a negation (Quirk et al., 2000; van der Wouden, 1997). Negation can be expressed by verbs (e.g., *avoid* running), nouns (e.g., the *absence* of evidence), adjectives (e.g., it is *pointless*), adverbs (e.g., I *never* tried Persian food before), prepositions (e.g., you can exchange it *without* a problem), determiners (e.g., the new law has *no* direct implications), pronouns (e.g., *nobody* will keep election promises), and others. In this paper, we focus on verbal negation, i.e., when the negation mark—usually an adverb such as *never* and *not*—is grammatically associated with a verb.

Positive Interpretations. In philosophy and linguistics, it is generally accepted that negation conveys positive meaning (Horn, 1989). This positive meaning ranges from implicatures, i.e., what is suggested in an utterance even though neither expressed nor strictly implied (Blackburn, 2008), to entailments. Other terms used in the literature include implied meanings (Mitkov, 2005), implied alternatives (Rooth, 1985) and semantically similars (Agirre et al., 2013). We do not strictly fit into any of this terminology, we reveal positive interpretations as intuitively done by humans when reading text.

2.1 Scope and Focus

From a theoretical perspective, it is accepted that negation has scope and focus, and that the focus—not just the scope—yields positive interpretations (Horn, 1989; Rooth, 1992; Taglicht, 1984). Scope is "the part of the meaning that is negated" and focus "the part of the scope that is most prominently or explicitly negated" (Huddleston and Pullum, 2002).

Consider the following statement in the context of the recent refugee crisis: (2) *Mr. Haile was* not *looking for heaven in Europe*. By definition, scope refers to "all elements whose individual falsity would make the negated statement strictly true", and focus is "the element of the scope that is intended to be interpreted as false to make the overall negative true" (Huddleston and Pullum, 2002). The falsity of any of the truth conditions below makes statement (2) true, thus the scope of the negation is (2a–2d):

2a. Somebody was looking for something somewhere. [verb *looking*]

2b. Mr. Haile was looking for something somewhere. [AGENT of looking, *Mr. Haile*]

2c. Somebody was looking for heaven somewhere. [THEME of looking, *heaven*]

2d. Somebody was looking for something in Europe. [LOCATION of looking, *in Europe*]

Determining the focus is almost always more challenging than the scope. The challenge relies on determining which of the truth conditions (2a–2d) is intended to be interpreted as false to make the negated statement true: all of them qualify, but some are more likely. A natural reading of statement (2) suggests that *Mr. Haile was looking for something (a regular life, a job, etc.) in Europe, but not heaven*. Determining that the focus is *heaven*, i.e., that everything in statement (2) is positive except the THEME of *looking*, is the key to reveal the intended positive interpretation. Note that scope on its own does not identify positive interpretations, and other foci yield unlikely positive interpretations, e.g., *Mr. Haile was looking for heaven somewhere, but not in Europe*.

It is worth noting that while scope is defined from a logical standpoint, in most negations there are several possible foci and corresponding positive interpretations. For example, given (3) *Most jobs now don't last for decades*, the following are valid positive interpretations: (3a) *Few jobs now last for decades*, (3b) *Most jobs in the past lasted for decades*, and (3c) *Most jobs now last for a few years*.

Granularity of Focus. The definition of focus does not provide guidelines about identifying the element of the scope that is the focus. The larger the focus, the more generic the corresponding positive interpretation; and the smaller the focus, the more specific the corresponding positive interpretation. Let us consider statement (3) again. A possible focus is *Most jobs*, yielding the positive interpretation *Something now lasts for decades, but not most jobs*. Another possible focus is *Most*, yielding the interpretation *Few (not most) jobs now last for decades*. We argue that the latter is preferable, as it yields a more specific interpretation and it entails the former: if some jobs last for decades, then something lasts for decades, but not the other way around.

We use the term *coarse-grained focus* to refer to foci that include *all* tokens belonging to an argument of a verb (e.g., *Most Jobs* above), and *fine-grained focus* to refer to foci that do not (e.g., *Most* above).

3 Previous Work

Within computational linguistics, approaches to process negation are shallow, or target scope and focus detection. Popular semantic representations such as semantic roles (Palmer et al., 2005; Baker et al., 1998) or AMR (Banarescu et al., 2013) do not reveal the positive interpretations we target in this paper. Shallow approaches are usually application-specific. In sentiment and opinion analysis, negation has been reduced to marking as negated all words between a negation cue and the first punctuation mark (Pang et al., 2002), or within a five-word window of a negation cue (Hu and Liu, 2004). The examples throughout this paper show that these techniques are insufficient to reveal implicit positive interpretations.

3.1 Scope Annotations and Detection

Scope of negation detection has received a lot of attention, mostly using two corpora: BioScope in the medical domain (Szarvas et al., 2008) and CD-SCO (Morante and Daelemans, 2012). BioScope annotates negation cues and linguistic scopes exclusively in biomedical texts. CD-SCO annotates negation cues, scopes, and negated events or properties in selected Conan Doyle stories.

There have been several supervised proposals to detect the scope of negation using BioScope and CD-SCO (Özgür and Radev, 2009; Øvrelid et al., 2010). Automatic approaches are mature (Abu-Jbara and Radev, 2012): F-scores are 0.96 for negation cue detection, and 0.89 for negation cue and scope detection (Velldal et al., 2012; Li et al., 2010). Fancellu et al. (2016) present the best results to date using CD-SCO, and analyze the main sources of errors. Outside BioScope and CD-SCO, Reitan et al. (2015) present a negation scope detector for tweets, and show that it improves sentiment analysis. As shown in Section 2, scope detection is insufficient to reveal positive interpretations from negation.

3.2 Focus Annotation and Detection

While focus of negation has been studied for decades in philosophy and linguistics (Section 2), corpora and automated tools are scarce. Blanco and Moldovan (2011) annotate focus of negation in the 3,993 negations marked with ARGM-NEG semantic role in PropBank (Palmer et al., 2005). Their annotations, PB-FOC, were used in the *SEM-2012 Shared Task (Morante and Blanco, 2012). Their guidelines require annotators to choose as focus the semantic role that "is most prominently negated" or the verb. If several roles may be the focus, they prioritize "the one that yields the most meaningful implicit [positive] information", but do not specify what *most meaningful* means. Their approach has 2 limitations. First, because they select one focus per negation, they only extract one positive interpretation per negation. Second, because they select as focus a semantic role, they only consider coarse-grained foci. Consider again statement (3) from Section 2.1. By design, their approach is limited to extract a single interpretation even though interpretations (3a–3c) are valid. Similarly, their approach is limited to select as focus *Most jobs*—all tokens belonging to a semantic role—although *Most* yields a "more meaningful" interpretation: *Something now lasts for decades* (generic, worse) vs. *Few jobs now last for decades* (specific, better).

Blanco and Sarabi (2016) present a complimentary approach to extract and score several positive interpretations from a single verbal negation. Their methodology is grounded on semantic roles and does not consider fine-grained foci. In this paper, we improve upon their work: we extract both coarse- and fine-grained interpretations, and also extract several interpretations from one negation.

Anand and Martell (2012) reannotate PB-FOC and argue that positive interpretations arising from scalar implicatures and neg-raising predicates should be separated from those arising from focus detection. They argue that 27.4% of negations with a focus annotated in PB-FOC do not have one. In this paper, we are not concerned about annotating foci per se, but about extracting positive interpretations from negation, as intuitively done by humans.

Automatic systems to detect the focus of negation yield modest results. Blanco and Moldovan (2011) obtain an accuracy of 65.5 using supervised learning and features derived from gold-standard linguistic information. With predicted linguistic information, Rosenberg and Bergler (2012) report an F-measure of 58.4 using 4 linguistically sound heuristics, and Zou et al. (2014) an F-measure of 65.62 using contextual discourse information. Blanco and Sarabi (2016) obtain Pearson correlation of 0.642

ranking coarse-grained interpretations. Unlike the work presented here, none of these systems extract fine-grained interpretations from a single negation.

4 Corpus Creation

Our goal is to create a corpus of negations and their positive interpretations. We put a strong emphasis on automation and simplicity. First, we deterministically generate potential positive interpretations from verbal negations by manipulating syntactic dependencies (Section 4.1). Second, we ask annotators to score potential positive interpretations (Section 4.2). Positive interpretations and their scores are later used to learn models to rank potential interpretations automatically (Section 6). Generating potential interpretations deterministically prior to scoring them proved very beneficial. After pilot experiments, it became clear that asking annotators to propose positive interpretations complicates the annotation effort (lower agreements) as well as learning.

We decided to work on top of OntoNotes (Hovy et al., 2006)[2] instead of plain text or other corpora for several reasons. First, OntoNotes includes gold linguistic annotations such as part-of-speech tags, parse trees and semantic roles. Second, unlike BioScope, CD-SCO and PB-FOC (Section 3.2), OntoNotes includes sentences from several genres, e.g., newswire, broadcast news and conversations, magazines, the web. We transformed the parse trees in OntoNotes into syntactic dependencies using Stanford CoreNLP (Manning et al., 2014).

4.1 Manipulating Syntactic Dependencies to Generate Potential Positive Interpretations

OntoNotes contains 63,918 sentences. Annotating all positive interpretations from all negations is outside the scope of this paper. Instead, we target selected representative negations.

Selecting Negations. We first select all verbal negations by retrieving all tokens whose syntactic head is a verb and dependency type *neg*.[3] Then, we discard negations from sentences that contain two negations, conditionals, commas or questions. Finally, we discard negations if the negated verb is *to be* or it does not have a subject (dependency *nsubj* or *nsubjpass*).

Converting Negated Statements into their positive counterparts. We apply 3 steps inspired after the grammatical rules to form negation detailed by Huddleston and Pullum (2002, Ch. 9):

1. Remove the negation mark by deleting the token with syntactic dependency *neg*.
2. Remove auxiliaries, expand contractions, and fix third-person singular and past tense. For example (before: after), *doesn't go*: *goes*, *didn't go*: *went*, *won't go*: *will go*. We loop through the tokens whose head is the negated verb with dependency *aux*, and use a list of irregular verbs and grammar rules to convert to third-person singular and past tense.
3. Rewrite negatively-oriented polarity-sensitive items. For example (before: after), *anyone*: *someone*, *any longer*: *still*, *yet*: *already*. *at all*: *somewhat*. We use the correspondences between negatively-oriented and positively-oriented polarity-sensitive items by (Huddleston and Pullum, 2002, pp. 831).

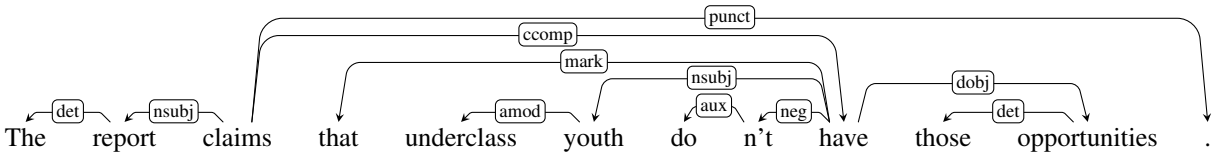
Selecting Relevant tokens. Verbal negation often occurs in multi-clause sentences. In order to identify the relevant (syntactically negated) eventuality, we simplify the original statement by including only the negated verb and all tokens that are dependents of the verb, i.e., tokens reachable from the negated verb traversing dependencies. For example, from *Individuals familiar with the Justice Department's policy said that Justice officials hadn't any knowledge of the IRS's actions in the last week*, after getting the positive counterpart and selecting relevant tokens, we obtain *Justice officials had some knowledge of the IRS's actions in the last week*.

Generating Interpretations. Given the simplified positive counterpart, generating all combinations of tokens as potential foci would result in 2^t potential positive interpretations for t tokens. To avoid a brute-force approach that generates many nonsensical potential interpretations, we define a procedure grounded on syntactic dependencies.

The main idea is to run a modified breadth-first traversal of the dependency tree to select subtrees that are potential foci. We start the traversal from the negated verb and stop it at depth 3, selecting as potential foci the subtrees rooted at all tokens except

[2] We use the CoNLL-2011 Shared Task distribution (Pradhan et al., 2011), `http://conll.cemantix.org/2011/`

[3] The Stanford manual describes and exemplifies all syntactic dependencies (de Marneffe and Manning, 2008).



Negated statement: The report claims that underclass youth don't have those opportunities.			
Positive counterpart	Step 1		The report claims that underclass youth do have those opportunities.
	Step 2		The report claims that underclass youth have those opportunities.
	Step 3		The report claims that underclass youth have those opportunities. (idem)
Relevant tokens	Underclass youth have those opportunities.		
Potential positive interpretations	*none*	coarse	Underclass youth [some verb] those opportunities, *but not have*.
	nsubj	coarse	[Some people] have those opportunities, but not *Underclass youth*.
	amod	fine	[Some adjective] youth have those opportunities, but not *Underclass youth*.
	nsubj	fine	Underclass [some people] have those opportunities, but not *Underclass youth*.
	dobj	coarse	Underclass youth have [something], but not *those opportunities*.
	det	fine	Underclass youth have [some] opportunities, but not *those opportunities*.
	dobj	fine	Underclass youth have those [something], but not *those opportunities*.

Table 1: Negated statement and syntactic dependencies (top), and automatically generated positive counterpart and potential positive interpretations (bottom). For potential interpretations, we include the dependency from the focus to the rest of the interpretation.

those whose syntactic dependency is *aux*, *auxpass* or *punct* (auxiliary, passive auxiliary and punctuation). Additionally, we discard potential foci that consist only of (1) the determiners *the*, *a* and an, or (2) a single token with part-of-speech tag TO, CC, UH, POS, XX, IN, WP or dependency relation *prt*. These rules were defined after manually observing several examples and concluding that the corresponding positive interpretation was useless. For example, from the negated statement *And our credit standards haven't changed one iota*, we avoid generating the useless potential interpretation *Our credit standards X changed one iota, but not have changed.* (focus would be *have*, with dependency *aux*). Similarly, from *It is not supported by the text or history of the Constitution*, we avoid generating potential interpretation *It is supported by X text or history of the Constitution, but not by the text or history of the Constitution* (focus would be *the*); and from *You don't want to get yourself too upset about these things*, potential interpretation *You want X get yourself too upset about these things, but not to get* (focus would be *to*, with part-of-speech tag TO).

Once potential foci are selected, we generate positive interpretations by rewriting each focus with "someone/some people/something/etc." and appending "but not text_of_focus" at the end. Additionally, if the first token of the focus is a preposition, we include it to improve readability, e.g., *didn't leave [by noon]*: *left by sometime, but not by noon*.

Note that potential interpretations obtained from foci that are direct syntactic dependents of the negated verb are coarse-grained interpretations, and the rest are fine-grained interpretations. Table 1 exemplifies the procedure step by step.

4.2 Scoring Potential Positive Interpretations

After generating potential positive interpretations automatically, we asked annotators to score them. Annotators had access to the original negated sentence, the previous and next sentence as context, and one potential positive interpretation at a time. The interface asked *Given the three sentences [previous sentence, negated sentence and next sentence] above, do you think the statement [positive interpretation] below is true?* Annotators were forced to answer with a score from 0 to 5, where 0 means *absolutely disagree* and 5 means *absolutely agree*. We did not provide descriptions for intermediate scores or use categorical labels. This simple guidelines were sufficient to reliably score plausible positive interpretations automatically generated (Section 5).

5 Corpus Analysis

The procedure described in Section 4.1 generates 9729 potential positive interpretations (5865 coarse-grained and 3864 fine-grained) from 1671 verbal negations. Out of all these potential positive interpretations, we annotate 1700 (1008 coarse- and 692

	Negated statement, context if relevant to determining scores, and all positive interpretations	Score
1	Context, previous statement:You're not giving me enough benefits. Negated Statement: You're not paying me for my overtime work. Context, next statement: Well I think the Walton family does take it personally.	
	- Int. 1.1 [coarse, root]: You're [some verb] me for my overtime work, but not *paying*.	4
	- Int. 1.2 [coarse, nsubj]: [Some people]'re paying me for my overtime work, but not *you*.	0
	- Int. 1.3 [coarse, dobj]: You're paying [somebody] for my overtime work, but not *me*.	1
	- Int. 1.4 [coarse, prep]: You're paying me for [something], but not *for my overtime work*.	5
	- Int. 1.5 [fine, poss]: You're paying me for [somebody's] overtime work, but not *for my overtime work*.	0
	- Int. 1.6 [fine, nn]: You're paying me for my [some adjective] work, but not *for my overtime work*.	5
	- Int. 1.7 [fine, pobj]: You're paying me for my overtime [something] but not *for my overtime work*.	0
2	Negated Statement: Those concerns aren't expressed in public.	
	- Int. 2.1 [coarse, root]: Those concerns are [some verb] in public, but not *expressed*.	5
	- Int. 2.2 [coarse, nsubjpass]: [Some things] are expressed in public, but not *Those concerns*.	5
	- Int. 2.3 [fine, nsubjpass]: Those [some noun] are expressed in public, but not *Those concerns*.	2
	- Int. 2.4 [fine, det]: [Some] concerns are expressed in public but, not *Those concerns*.	4
	- Int. 2.5 [coarse, prep]: Those concerns are expressed in [somewhere], but not *in public*.	5

Table 3: Negated statements, all potential positive interpretations automatically generated and their manually assigned scores.

Dependency	#	%	Mean	SD
nsubj	358	21.13%	3.36	1.47
dobj	237	14.05%	3.73	1.57
pobj	178	10.51%	3.48	1.59
ccomp	125	7.29%	3.29	1.77
advmod	108	6.39%	3.33	1.59
xcomp	90	5.28%	3.92	1.50
amod	67	3.96%	4.08	1.29
conj	40	2.38%	2.80	1.60
advcl	40	2.32%	2.84	1.80
nsubjpass	35	2.17%	3.63	1.51
other	209	13.34%	2.9	1.7
verb	213	12.5%	2.01	1.46
All	1,700	100.00%	3.20	1.66

Table 2: Basic corpus analysis. For each dependency, we show the number of potential interpretations generated (#) and percentage (%), mean score and standard deviation.

fine-grained). Overall, the mean score is 3.20, and the standard deviation is 1.66. Table 2 shows basic statistics for potential foci, where *dependency* indicates the dependency from the potential focus to a token outside the potential focus. Most foci are *nsubj*, *dobj* and *pobj*, and the mean scores and standard deviation are similar for most dependencies.

Annotation Quality. In order to ensure annotation quality, we calculated Pearson correlation. Kappa and other measures designed for categorical labels are ill-suited for our annotations, since not all disagreements between numeric scores are the same, e.g., 4 vs. 5 should be counted as higher agreement, than 1 vs. 5. Overall Pearson correlation was 0.75.

5.1 Annotation Examples

Table 3 presents 2 statements that contain verbal negation, the list of positive interpretations automatically generated and the annotated scores.

Example (1) is a simple negated clause, yet we generate 7 potential positive interpretations and 3 of them receive high scores (4 or 5). Given *You're not paying me for my overtime work* and the previous statement, it is reasonable to believe that the author is in an employee-employer relationship, and the employer is not fair to the employee. Interpretations 1.1, 1.4 and 1.6 are implicit positive interpretations intuitively understood by humans when reading the original negated statement. Namely, Interpretation 1.1: *You (the employer) are nickel-and-diming me for my overtime work* (focus is *paying*), Interpretation 1.4: *You (the employer) are paying me for something* (focus is *my overtime work*), and Interpretation 1.6: *You (the employer) are paying me for my regular work* (focus is *overtime*). These interpretations show the benefits of fine-grained interpretations: Interpretation 1.6 is a refinement of Interpretation 1.4, and the former is more desirable than the latter as it reveals more specific positive knowledge. The remaining interpretations are legible, but do not make sense given the negated statement, e.g., interpretation 1.2: *Somebody (but not the employer) pays me for my overtime* (focus is *You*).

Example (2) is also a simple negated clause, and 4 out of 5 interpretations receive high scores, capturing valid positive meaning. Specifically, Interpreta-

Type	Name	Description
Basic	neg_mark	word form of negation mark
	verb	word form and part-of-speech tag of verb
	coarse_or_fine	flag indicating whether interpretation is coarse- or fine-grained
Path	syn_path_dep	syntactic path from focus to verb (concatenation of dependencies)
	syn_path_pos	syntactic path from focus to verb (concatenation of part-of-speech tags)
	syn_path_last_dep	last syntactic dependency in syn_path_dep (direct dependent of verb)
	syn_path_last_pos	last part-of-speech tag in syn_path_pos (direct dependent of verb)
Focus	focus_length	number of words in subgraph chosen as focus
	focus_first_word	word form and part-of-speech tag of first word in focus
	focus_last_word	word form and part-of-speech tag of last word in focus
	focus_direction	flag indicating whether focus occurs before or after verb
	focus_head_word	word form of head of focus
	focus_head_pos	part-of-speech tag of head of focus
	focus_head_rel	syntactic dependency of head of focus

Table 4: Features used to score potential positive interpretations automatically generated.

tion 2.1: *Those concerns are avoided in public* (focus is *expressed*), Interpretation 2.2: *Something is expressed in public* (focus is *Those concerns*), Interpretation 2.4: *Some concerns (but not problematic or secret concerns) are expressed in public* (focus is *Those*), and Interpretation 2.5: *Those concerns are expressed in private* (focus is *in public*).

5.2 Syntactic Dependencies vs. Semantic Roles

The procedure presented in Section 4.1 is not the first to generate potential positive interpretations from negation (Section 3.2). Our approach has 2 advantages with respect to those grounded on semantic roles (Blanco and Sarabi, 2016): (1) it generates both coarse- and fine-grained interpretations, and (2) learning to score interpretations is easier because state-of-the-art tools extract dependencies more reliably than semantic roles.

To support claim (1), we compare the interpretations generated with our procedure and previous work using semantic roles. 96.12% of interpretations generated using roles are also generated using syntactic dependencies. Also, using dependencies allow us to generate 67.9% of additional (fine-grained) interpretations not obtainable with roles.

To support claim (2), we compare interpretations generated with gold and predicted linguistic information (roles or dependencies). The overlap with semantic roles is 70.1%, and with syntactic dependencies, 92.8%. Syntactic dependencies are thus better in a realistic scenario because they allow us to automatically generate (and score) most interpretations.

6 Supervised Learning to Score Potential Positive Interpretations

We follow a standard supervised machine learning approach. The 1,700 potential positive interpretations along with their scores become instances, and we divide them into training (80%) and test splits (20%) making sure that all interpretations generated from a sentence are assigned to either the training or test splits. Note that splitting instances randomly would not be sound: training with some interpretations generated from a negation, and testing with the rest of interpretations generated from the same negation would be an unfair evaluation.

We train a Support Vector Machine for regression with RBF kernel using scikit-learn (Pedregosa et al., 2011), which in turn uses LIBSVM (Chang and Lin, 2011). SVM parameters (C and γ) were tuned using 10-fold cross-validation with the training set, and results are calculated using the test set.

6.1 Feature Selection

Table 4 presents the full feature set. Features are relatively simple and characterize the verbal negation from which a potential interpretation was generated, as well as the interpretation per se, i.e., the dependency subgraph chosen as potential focus.

Basic features account for the negation mark, the negation verb (word form and part-of-speech tag) and a binary flag indicating whether we are scoring a coarse- or fine-grained interpretation.

Path features are derived from the syntactic path

Features	Gold	Predicted
neg_mark	-0.109	-0.077
basic	0.033	0.026
basic + path	0.474	0.482
basic + path + focus	0.530	0.560

Table 5: Pearson correlations obtained with the test split. Results are provided using gold-standard and predicted linguistic information (part-of-speech tags and syntactic dependencies).

between the subgraph selected as focus and the verb. We include the actual path (concatenation of dependencies and up/down symbols), and the modified path using part-of-speech tags. Additionally, we also include the last dependency and part-of-speech tag, i.e., the ones closest to the verb in the path.

Focus features characterize the dependency subgraph chosen as focus to generate the potential interpretation. Specifically, we include the number of tokens, word form and part-of-speech tags of the first and last tokens, and whether the focus occurs before or after the verb. We also include features derived form the head of the focus, which we define as the token whose syntactic head is outside the focus. We include the word form and part-of-speech of the focus head, as well as its the dependency.

7 Experiments and Results

We report results obtained with several combinations of features in Table 5. We detail results obtained with features extracted from gold-standard and predicted linguistic annotations (part-of-speech tags and syntactic dependencies) as annotated in the *gold* and *auto* files from the CoNLL-2011 Shared Task release of OntoNotes (Pradhan et al., 2011). All models are trained with gold-standard linguistic annotations, and tested with either gold-standard or predicted linguistic annotations.

Testing with gold-standard POS tags and syntactic dependencies. Training with the word form of the negation mark is virtually useless, it yields a Pearson correlation of -0.109. *Basic* features (negation mark, verb and flag indicating coarse- or fine-grained interpretation) are also ineffective to score potential interpretations (Pearson: 0.033). Including features derived from the syntactic *path* yields higher correlation, 0.474, even though these features only capture the syntactic relationship between the focus from which the interpretation was generated and the verb. Finally, adding *focus* features yields the best results (Pearson: 0.53, $+11.8\%$).

Testing with predicted POS tags and syntactic dependencies. We selected 20% of positive interpretations in our corpus as test instances, totalling 379 interpretations (Section 6). When executing the procedure to generate potential interpretations (Section 4.1) with predicted linguistic information, however, we are unable to generate all of them due to incorrect and missing syntactic dependencies. Specifically, 352 of the 379 interpretations are generated (92.8%). While we do not generate 7.2% of instances, this percentage is substantially lower than previous work grounded on semantic roles (Section 5.2).

Pearson correlations with predicted linguistic information are calculated using the 352 instances that were also generated with gold dependencies (and thus assigned a score during the manual annotations). Correlations are slightly higher and follow a similar trend than the correlations obtained with gold-standard linguistic information. These results should be taken with a grain of salt: the test instances are not exactly the same, and the 352 test instances in this scenario are presumably easier to score than the remainder 27, as dependencies were predicted correctly.

8 Conclusions

Humans intuitively extract positive meaning from negation when reading text. This paper presents an automated procedure to generate potential positive interpretations from verbal negation, and score them according to their likelihood. Our procedure is grounded on syntactic dependencies, allowing us to extract fine-grained interpretations beyond semantic roles (67.9% additional interpretations). Additionally, because dependencies are extracted automatically more reliably than semantic roles, we generate 92.8% of all potential interpretations when using predicted linguistic information, as opposed to 70.1% with semantic roles.

On average, we generate 6.4 potential interpretations per verbal negation (coarse-grained: 3.8, fine-grained: 2.6). Manual annotations show that potential interpretations are deemed likely. The mean

score is 3.20 (out of 5.0), thus we extract a substantial amount of positive meaning.

The work presented in this paper is not tied to any existing semantic representation. While we rely heavily on syntactic dependencies, positive interpretations are generated in plain text, and they could be processed, along with the original negated statement, with any NLP pipeline.

References

Amjad Abu-Jbara and Dragomir Radev. 2012. Umichigan: A conditional random field model for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 328–334. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Pranav Anand and Craig Martell. 2012. Annotating the focus of negation in terms of questions under discussion. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 65–69, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, Montreal, Canada.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Simon Blackburn. 2008. *The Oxford Dictionary of Philosophy*. Oxford University Press.

Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 581–589, Portland, Oregon, USA, June. Association for Computational Linguistics.

Eduardo Blanco and Zahra Sarabi. 2016. Automatic generation and scoring of positive interpretations from negated statements. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1431–1441, San Diego, California, June. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May.

H. H. Clark and W. G. Chase. 1972. On the process of comparing sentences against pictures. *Cognitive Psychology*, 3(3):472–517, July.

Isaac Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59, Uppsala, Sweden, July. University of Antwerp.

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

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 495–504, Berlin, Germany, August. Association for Computational Linguistics.

Laurence R. Horn and Heinrich Wansing. 2015. Negation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2015 edition.

Laurence R. Horn. 1989. *A natural history of negation*. Chicago University Press, Chicago.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA. ACM.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, April.

Junhui Li, Guodong Zhou, Hongling Wang, and Qiaoming Zhu. 2010. Learning the Scope of Negation via

Shallow Semantic Parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 671–679, Beijing, China, August. Coling 2010 Organizing Committee.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ruslan Mitkov. 2005. *The Oxford handbook of computational linguistics*. Oxford University Press.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 265–274, Montréal, Canada, June.

Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul*.

Roser Morante and Caroline Sporleder. 2012. Modality and negation: An introduction to the special issue. *Comput. Linguist.*, 38(2):223–260, June.

Ann E Nordmeyer and Michael C Frank. 2013. Measuring the comprehension of negation in 2-to 4-year-old children. *Proceedings of the 35th Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society*.

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010. Syntactic Scope Resolution in Uncertainty Analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1379–1387, Beijing, China, August. Coling 2010 Organizing Committee.

Arzucan Özgür and Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1398–1407, Singapore, August. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics, July.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA, June. Association for Computational Linguistics.

Randolph Quirk, Sidney Greenbaum, and Geoffrey Leech. 2000. *A comprehensive grammar of the English language*. Longman, London.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108, Lisboa, Portugal, September. Association for Computational Linguistics.

Mats Rooth. 1985. Association with focus.

Mats Rooth. 1992. A theory of focus interpretation. *Natural language semantics*, 1(1):75–116.

Sabine Rosenberg and Sabine Bergler. 2012. Uconcordia: Clac negation focus detection at *sem 2012. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 294–300, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scopein biomedical texts. In *Proceedings of BioNLP 2008*, pages 38–45, Columbus, Ohio, USA. ACL.

Josef Taglicht. 1984. *Message and emphasis: On focus and scope in English*, volume 15. Addison-Wesley Longman Limited.

Ton van der Wouden. 1997. *Negative contexts: collocation, polarity, and multiple negation*. Routledge, London.

Erik Velldal, Lilja Ovrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Comput. Linguist.*, 38(2):369–410, June.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In *Proceedings of the 52nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 522–530, Baltimore, Maryland, June. Association for Computational Linguistics.

Demographic Dialectal Variation in Social Media: A Case Study of African-American English

Su Lin Blodgett[†] **Lisa Green**[*] **Brendan O'Connor**[†]
[†]College of Information and Computer Sciences [*]Department of Linguistics
University of Massachusetts Amherst

Abstract

Though dialectal language is increasingly abundant on social media, few resources exist for developing NLP tools to handle such language. We conduct a case study of dialectal language in online conversational text by investigating African-American English (AAE) on Twitter. We propose a distantly supervised model to identify AAE-like language from demographics associated with geo-located messages, and we verify that this language follows well-known AAE linguistic phenomena. In addition, we analyze the quality of existing language identification and dependency parsing tools on AAE-like text, demonstrating that they perform poorly on such text compared to text associated with white speakers. We also provide an ensemble classifier for language identification which eliminates this disparity and release a new corpus of tweets containing AAE-like language.

Data and software resources are available at: http://slanglab.cs.umass.edu/TwitterAAE

1 Introduction

Owing to variation within a standard language, regional and social dialects exist within languages across the world. These varieties or dialects differ from the standard variety in syntax (sentence structure), phonology (sound structure), and the inventory of words and phrases (lexicon). Dialect communities often align with geographic and sociological factors, as language variation emerges within distinct social networks, or is affirmed as a marker of social identity.

As many of these dialects have traditionally existed primarily in oral contexts, they have historically been underrepresented in written sources. Consequently, NLP tools have been developed from text which aligns with mainstream languages. With the rise of social media, however, dialectal language is playing an increasingly prominent role in online conversational text, for which traditional NLP tools may be insufficient. This impacts many applications: for example, dialect speakers' opinions may be mischaracterized under social media sentiment analysis or omitted altogether (Hovy and Spruit, 2016). Since this data is now available, we seek to analyze current NLP challenges and extract dialectal language from online data.

Specifically, we investigate dialectal language in publicly available Twitter data, focusing on African-American English (AAE), a dialect of Standard American English (SAE) spoken by millions of people across the United States. AAE is a linguistic variety with defined syntactic-semantic, phonological, and lexical features, which have been the subject of a rich body of sociolinguistic literature. In addition to the linguistic characterization, reference to its speakers and their geographical location or speech communities is important, especially in light of the historical development of the dialect. Not all African-Americans speak AAE, and not all speakers of AAE are African-American; nevertheless, speakers of this variety have close ties with specific communities of African-Americans (Green, 2002). Due to its widespread use, established history in the sociolinguistic literature, and demographic associations, AAE provides an ideal starting point for the development of a statistical model that uncovers dialectal

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1119–1130,
Austin, Texas, November 1-5, 2016.

language. In fact, its presence in social media is attracting increasing interest for natural language processing (Jørgensen et al., 2016) and sociolinguistic (Stewart, 2014; Eisenstein, 2015; Jones, 2015) research.[1] In this work we:

- Develop a method to identify *demographically-aligned* text and language from geo-located messages (§2), based on distant supervision of geographic census demographics through a statistical model that assumes a soft correlation between demographics and language.
- Validate our approach by verifying that text aligned with African-American demographics follows well-known phonological and syntactic properties of AAE, and document the previously unattested ways in which such text diverges from SAE (§3).
- Demonstrate *racial disparity* in the efficacy of NLP tools for language identification and dependency parsing—they perform poorly on this text, compared to text associated with white speakers (§4, §5).
- Improve language identification for U.S. online conversational text with a simple ensemble classifier using our demographically-based distant supervision method, aiming to eliminate racial disparity in accuracy rates (§4.2).
- Provide a corpus of 830,000 tweets aligned with African-American demographics.

2 Identifying AAE from Demographics

The presence of AAE in social media and the generation of resources of AAE-like text for NLP tasks has attracted recent interest in sociolinguistic and natural language processing research; Jones (2015) shows that nonstandard AAE orthography on Twitter aligns with historical patterns of African-American migration in the U.S., while Jørgensen et al. (2015) investigate to what extent it supports well-known sociolinguistics hypotheses about AAE. Both, however, find AAE-like language on Twitter through keyword searches, which may not yield broad corpora reflective of general AAE use. More recently, Jørgensen et al. (2016) generated a large unlabeled corpus of text from hip-hop lyrics, subtitles from *The Wire* and *The Boondocks*, and tweets from a region of the southeast U.S. While this corpus does indeed capture a wide variety of language, we aim to discover AAE-like language by utilizing finer-grained, neighborhood-level demographics from across the country.

Our approach to identifying AAE-like text is to first harvest a set of messages from Twitter, cross-referenced against U.S. Census demographics (§2.1), then to analyze words against demographics with two alternative methods, a seedlist approach (§2.2) and a mixed-membership probabilistic model (§2.3).

2.1 Twitter and Census data

In order to create a corpus of demographically-associated dialectal language, we turn to Twitter, whose public messages contain large amounts of casual conversation and dialectal speech (Eisenstein, 2015). It is well-established that Twitter can be used to study both geographic dialectal varieties[2] and minority languages.[3]

Some methods exist to associate messages with authors' races; one possibility is to use birth record statistics to identify African-American-associated names, which has been used in (non-social media) social science studies (Sweeney, 2013; Bertrand and Mullainathan, 2003). However, metadata about authors is fairly limited on Twitter and most other social media services, and many supplied names are obviously not real.

Instead, we turn to geo-location and induce a distantly supervised mapping between authors and the demographics of the neighborhoods they live in (O'Connor et al., 2010; Eisenstein et al., 2011b; Stewart, 2014). We draw on a set of geo-located Twitter messages, most of which are sent on mobile phones, by authors in the U.S. in 2013. (These are selected from a general archive of the "Gardenhose/Decahose" sample stream of public Twit-

[1] Including a recent linguistics workshop: http://linguistlaura.blogspot.co.uk/2016/06/using-twitter-for-linguistic-research.html

[2] For example, of American English (Huang et al., 2015; Doyle, 2014).

[3] For example, Lynn et al. (2015) develop POS corpora and taggers for Irish tweets; see also related work in §4.1.

ter messages (Morstatter et al., 2013)). Geolocated users are a particular sample of the userbase (Pavalanathan and Eisenstein, 2015), but we expect it is reasonable to compare users of different races within this group.

We look up the U.S. Census blockgroup geographic area that the message was sent in; blockgroups are one of the smallest geographic areas defined by the Census, typically containing a population of 600–3000 people. We use race and ethnicity information for each blockgroup from the Census' 2013 American Community Survey, defining four covariates: percentages of the population that are non-Hispanic whites, non-Hispanic blacks, Hispanics (of any race), and Asian.[4] Finally, for each user u, we average the demographic values of all their messages in our dataset into a length-four vector $\pi_u^{(census)}$. Under strong assumptions, this could be interpreted as the probability of which race the user is; we prefer to think of it as a rough proxy for likely demographics of the author and the neighborhood they live in.

Messages were filtered in order to focus on casual conversational text; we exclude tweets whose authors had 1000 or more followers, or that (a) contained 3 or more hashtags, (b) contained the strings "http", "follow", or "mention" (messages designed to generate followers), or (c) were retweeted (either containing the string "rt" or marked by Twitter's metadata as re-tweeted).

Our initial Gardenhose/Decahose stream archive had 16 billion messages in 2013; 90 million were geo-located with coordinates that matched a U.S. Census blockgroup. 59.2 million tweets from 2.8 million users remained after pre-processing; each user is associated with a set of messages and averaged demographics $\pi_u^{(census)}$.

2.2 Direct Word-Demographic Analysis

Given a set of messages and demographics associated with their authors, a number of methods could be used to infer statistical associations between language and demographics.

Direct word-demographic analysis methods use the $\pi_u^{(census)}$ quantities to calculate statistics at the word level in a single pass. An intuitive approach is to calculate the *average demographics per word*. For a token in the corpus indexed by t (across the whole corpus), let $u(t)$ be the author of the message containing that token, and w_t be the word token. The average demographics of word type w is:[5]

$$\pi_w^{(softcount)} \equiv \frac{\sum_t 1\{w_t = w\}\pi_{u(t)}^{(census)}}{\sum_t 1\{w_t = w\}}$$

We find that terms with the highest $\pi_{w,\text{AA}}$ values (denoting high average African-American demographics of their authors' locations) are very non-standard, while Stewart (2014) and Eisenstein (2013) find large $\pi_{w,\text{AA}}$ associated with certain AAE linguistic features.

One way to use the $\pi_{w,k}$ values to construct a corpus is through a seedlist approach. In early experiments, we constructed a corpus of 41,774 users (2.3 million messages) by first selecting the $n = 100$ highest-$\pi_{w,\text{AA}}$ terms occurring at least $m = 3000$ times across the data set, then collecting all tweets from frequent authors who have at least 10 tweets and frequently use these terms, defined as the case when at least $p = 20\%$ of their messages contain at least one of the seedlist terms. Unfortunately, the n, m, p thresholds are ad-hoc.

2.3 Mixed-Membership Demographic-Language Model

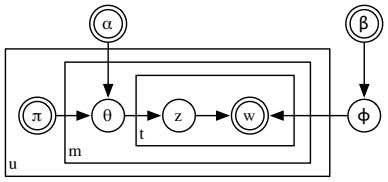
The direct word-demographics analysis gives useful validation that the demographic information may yield dialectal corpora, and the seedlist approach can assemble a set of users with heavy dialectal usage. However, the approach requires a number of ad-hoc thresholds, cannot capture authors who only occasionally use demographically-aligned language, and cannot differentiate language use at the message-level. To address these concerns, we develop a mixed-membership model for demographics and language use in social media.

The model directly associates each of the four demographic variables with a topic; i.e. a unigram language model over the vocabulary.[6] The model assumes an author's mixture over the topics tends to

[4] See appendix for additional details.

[5] $\pi_{w,k}$ has the flavor of "soft counts" in multinomial EM. By changing the denominator to $\sum_t \pi_{u(t)}^{(census)}$, it calculates a unigram language model that sums to one across the vocabulary. This hints at a more complete modeling approach (§2.3).

[6] To build the vocabulary, we select all words used by at least 20 different users, resulting in 191,873 unique words; other words are mapped to an out-of-vocabulary symbol.



$$\theta_m \sim Dir(\alpha\pi_u), \;\; \phi \sim Dir(\beta/V)$$
$$z_t \sim \theta_m, \;\; w_z \sim \phi_{z_t}$$

Figure 1: Mixed-membership model for users (u), messages (m) and tokens (t). Observed variables have a double lined border.

be similar to their Census-associated demographic weights, and that every message has its own topic distribution. This allows for a single author to use different types of language in different messages, accommodating multidialectal authors. The message-level topic probabilities θ_m are drawn from an asymmetric Dirichlet centered on $\pi_u^{(census)}$, whose scalar concentration parameter α controls whether authors' language is very similar to the demographic prior, or can have some deviation. A token t's latent topic z_t is drawn from θ_m, and the word itself is drawn from ϕ_{z_t}, the language model for the topic (Figure 1).

Thus the model learns *demographically-aligned* language models for each demographic category. The model is much more tightly constrained than a topic model—for example, if $\alpha \rightarrow \infty$, θ becomes fixed and the likelihood is concave as a function of ϕ—but it still has more joint learning than a direct calculation approach, since the inference of a messages' topic memberships θ_m is affected not just by the Census priors, but also by the language used. A tweet written by an author in a highly AA neighborhood may be inferred to be non-AAE-aligned if it uses non-AAE-associated terms; as inference proceeeds, this information is used to learn sharper language models.

We fit the model with collapsed Gibbs sampling (Griffiths and Steyvers, 2004) with repeated sample updates for each token t in the corpus,

$$p(z_t = k \mid w, z_{-t}) \propto \frac{N_{wk} + \beta/V}{N_k + \beta} \frac{N_{mk} + \alpha\pi_{uk}}{N_m + \alpha}$$

where N_{wk} is the number of tokens where word w occurs under topic $z = k$, N_{mk} is the number of tokens in the current message with topic k, etc.; all counts exclude the current t position. We observed convergence of the log-likelihood within 100 to 200 iterations, and ran for 300 total.[7] We average together count tables from the last 50 Gibbs samples for analysis of posterior topic memberships at the word, message, and user level; for example, the posterior probability a particular user u uses topic k, $P(z = k \mid u)$, can be calculated as the fraction of tokens with topic k within messages authored by u.

We considered α to be a fixed control parameter; setting it higher increases the correlations between $P(z = k \mid u)$ and $\pi_{u,k}^{(census)}$. We view the selection of α as an inherently difficult problem, since the correlation between race and AAE usage is already complicated and imperfect at the author-level, and census demographics allow only for rough associations. We set $\alpha = 10$ which yields posterior user-level correlations of $P(z = AA \mid u)$ against $\pi_{u,AA}$ to be approximately 0.8.

This model has broadly similar goals as non-latent, log-linear generative models of text that condition on document-level covariates (Monroe et al., 2008; Eisenstein et al., 2011a; Taddy, 2013). The formulation here has the advantage of fast inference with large vocabularies (since the partition function never has to be computed), and gives probabilistic admixture semantics at arbitrary levels of the data. This model is also related to topic models where the selection of θ conditions on covariates (Mimno and McCallum, 2008; Ramage et al., 2011; Roberts et al., 2013), though it is much simpler without full latent topic learning.

In early experiments, we used only two classes (AA and not AA), and found Spanish terms being included in the AA topic. Thus we turned to four race categories in order to better draw out non-AAE language. This removed Spanish terms from the AA topic; interestingly, they did not go to the Hispanic topic, but instead to Asian, along with other foreign languages. In fact, the correlation between users' Census-derived proportions of Asian populations, versus this posterior topic's proportions, is only 0.29, while the other three topics correlate to their respective Census priors in the range 0.83 to 0.87. This indicates the "Asian" topic actually functions as a background topic (at least in part). Better modeling of demographics and non-English

[7] Our straightforward single core implementation (in Julia) spends 80 seconds for each iteration over 586 million tokens.

language interactions is interesting potential future work.

By fitting the model to data, we can directly analyze unigram probabilities within the model parameters ϕ, but for other analyses, such as analyzing larger syntactic constructions and testing NLP tools, we require an explicit corpus of messages.

To generate a user-based AA-aligned corpus, we collected all tweets from users whose posterior probability of using AA-associated terms under the model was at least 80%, and generated a corresponding white-aligned corpus as well. In order to remove the effects of non-English languages, and given uncertainty about what the model learned in the Hispanic and Asian-aligned demographic topics, we focused only on AA- and white-aligned language by imposing the additional constraint that each user's combined posterior proportion of Hispanic or Asian language was less than 5%. Our two resulting user corpora contain 830,000 and 7.3 million tweets, for which we are making their message IDs available for further research (in conformance with the Twitter API's Terms of Service). In the rest of the work, we refer to these as the AA- and white-aligned corpora, respectively.

3 Linguistic Validation

Because validation by manual inspection of our AA-aligned text is impractical, we turn to the well-studied phonological and syntactic phenomena that traditionally distinguish AAE from SAE. We validate our model by reproducing these phenomena, and document a variety of other ways in which our AA-aligned text diverges from SAE.

3.1 Lexical-Level Variation

We begin by examining how much AA- and white-aligned lexical items diverge from a standard dictionary. We used SCOWL's largest wordlist with level 1 variants as our dictionary, totaling 627,685 words.[8]

We calculated, for each word w in the model's vocabulary, the ratio

$$r_k(w) = \frac{p(w|z = k)}{p(w|z \neq k)}$$

where the $p(.|.)$ probabilities are posterior inferences, derived from averaged Gibbs samples of the sufficient statistic count tables N_{wk}.

We selected heavily AA- and white-aligned words as those where $r_{AA}(w) \geq 2$ and $r_{white}(w) \geq 2$, respectively. We find that while 58.2% of heavily white-aligned words were not in our dictionary, fully 79.1% of heavily AA-aligned words were not. While a high number of out-of-dictionary lexical items is expected for Twitter data, this disparity suggests that the AA-aligned lexicon diverges from SAE more strongly than the white-aligned lexicon.

3.2 Internet-Specific Orthographic Variation

We performed an "open vocabulary" unigram analysis by ranking all words in the vocabulary by $r_{AA}(w)$ and browsed them and samples of their usage. Among the words with high r_{AA}, we observe a number of Internet-specific orthographic variations, which we separate into three types: abbreviations (e.g. *llh*, *kmsl*), shortenings (e.g. *dwn*, *dnt*), and spelling variations which do not correlate to the word's pronunciation (e.g. *axx*, *bxtch*). These variations do not reflect features attested in the literature; rather, they appear to be purely orthographic variations highly specific to AAE-speaking communities online. They may highlight previously unknown linguistic phenomena; for example, we observe that *thoe* (SAE *though*) frequently appears in the role of a discourse marker instead of its standard SAE usage (e.g. *Girl Madison outfit THOE*). This new use of *though* as a discourse marker, which is difficult to observe using the SAE spelling amidst many instances of the SAE usage, is readily identifiable in examples containing the *thoe* variant. Thus, non-standard spellings provide valuable windows into a variety of linguistic phenomena.

In the next section, we turn to variations which do appear to arise from known phonological processes.

3.3 Phonological Variation

Many phonological features are closely associated with AAE (Green, 2002). While there is not a perfect correlation between orthographic variations and people's pronunciations, Eisenstein (2013) shows that some genuine phonological phenomena, including a number of AAE features, are accurately reflected in orthographic variation on social media. We therefore validate our model by verifying that spellings reflecting known AAE phonological features align closely with the AA topic.

[8] http://wordlist.aspell.net/

AAE	Ratio	SAE
sholl	1802.49	sure
iont	930.98	I don't
wea	870.45	where
talmbout	809.79	talking about
sumn	520.96	something

Table 1: Of 31 phonological variant words, top five by ratio $r_{AA}(w)$. SAE translations are shown for reference.

We selected 31 variants of SAE words from previous studies of AAE phonology on Twitter (Jørgensen et al., 2015; Jones, 2015). These variations display a range of attested AAE phonological features, such as derhotacization (e.g. *brotha*), deletion of initial *g* and *d* (e.g. *iont*), and realization of voiced *th* as *d* (e.g. *dey*) (Rickford, 1999).

Table 1 shows the top five of these words by their $r_{AA}(w)$ ratio. For 30 of the 31 words, $r \geq 1$, and for 13 words, $r \geq 100$, suggesting that our model strongly identifies words displaying AAE phonological features with the AA topic. The sole exception is the word *brotha*, which appears to have been adopted into general usage as its own lexical item.

3.4 Syntactic Variation

We further validate our model by verifying that it reproduces well-known AAE syntactic constructions, investigating three well-attested AAE aspectual or preverbal markers: habitual *be*, future *gone*, and completive *done* (Green, 2002). Table 2 shows examples of each construction.

To search for the constructions, we tagged the corpora using the ARK Twitter POS tagger (Gimpel et al., 2011; Owoputi et al., 2013),[9] which Jørgensen et al. (2015) show has similar accuracy rates on both AAE and non-AAE tweets, unlike other POS taggers. We searched for each construction by searching for sequences of unigrams and POS tags characterizing the construction; e.g. for habitual *be* we searched for the sequences O-*be*-V and O-*b*-V. Nonstandard spellings for the unigrams in the patterns were identified from the ranked analysis of §3.2.

We examined how a message's likelihood of using each construction varies with the message's posterior probability of AA. We split all messages into deciles based on the messages' posterior probability of AA. From each decile, we sampled 200,000 messages and calculated the proportion of messages containing the three syntactic constructions.

[9] Version 0.3.2: http://www.cs.cmu.edu/~ark/TweetNLP/

Construction	Example	Ratio
O-*be/b*-V	*I be tripping bruh*	11.94
gone/gne/gon-V	*Then she gon be single Af*	14.26
done/dne-V	*I done laughed so hard that I'm weak*	8.68

Table 2: AAE syntactic constructions and the ratios of their occurrences in the AA- vs. white-aligned corpora (§2.3).

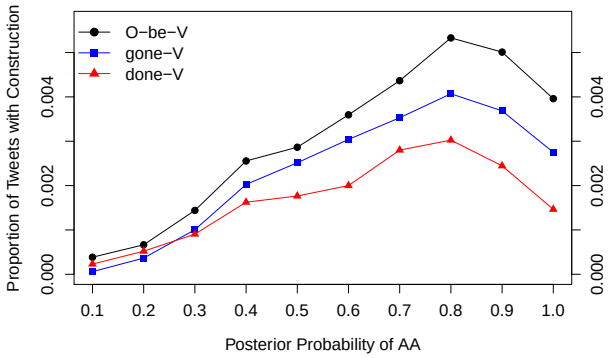


Figure 2: Proportion of tweets containing AAE syntactic constructions by messages' posterior probability of AA. On the x-axis, 0.1 refers to the decile [0, 0.1).

For all three constructions, we observed the clear pattern that as messages' posterior probabilities of AA increase, so does their likelihood of containing the construction. Interestingly, for all three constructions, frequency of usage peaks at approximately the [0.7, 0.8) decile. One possible reason for the decline in higher deciles might be tendency of high-AA messages to be shorter; while the mean number of tokens per message across all deciles in our samples is 9.4, the means for the last two deciles are 8.6 and 7.1, respectively.

Given the important linguistic differences between our demographically-aligned subcorpora, we hypothesize that current NLP tools may behave differently. We investigate this hypothesis in §4 and §5.

4 Lang ID Tools on AAE

4.1 Evaluation of Existing Classifiers

Language identification, the task of classifying the major world language in which a message is written, is a crucial first step in almost any web or social

	AAE	White-Aligned
langid.py	13.2%	7.6%
Twitter-1	8.4%	5.9%
Twitter-2	24.4%	17.6%

Table 3: Proportion of tweets in AA- and white-aligned corpora classified as non-English by different classifiers. (§4.1)

media text processing pipeline. For example, in order to analyze the opinions of U.S. Twitter users, one might throw away all non-English messages before running an English sentiment analyzer.

Hughes et al. (2006) review language identification methods; social media language identification is challenging since messages are short, and also use non-standard and multiple (often related) languages (Baldwin et al., 2013). Researchers have sought to model code-switching in social media language (Rosner and Farrugia, 2007; Solorio and Liu, 2008; Maharjan et al., 2015; Zampieri et al., 2013; King and Abney, 2013), and recent workshops have focused on code-switching (Solorio et al., 2014) and general language identification (Zubiaga et al., 2014). For Arabic dialect classification, work has developed corpora in both traditional and Romanized script (Cotterell et al., 2014; Malmasi et al., 2015) and tools that use n-gram and morphological analysis to identify code-switching between dialects and with English (Elfardy et al., 2014).

We take the perspective that since AAE is a dialect of American English, it ought to be classified as English for the task of major world language identification. Lui and Baldwin (2012) develop *langid.py*, one of the most popular open source language identification tools, training it on over 97 languages from texts including Wikipedia, and evaluating on both traditional corpora and Twitter messages. We hypothesize that if a language identification tool is trained on standard English data, it may exhibit disparate performance on AA- versus white-aligned tweets. Since language identifiers are typically based on character n-gram features, they may get confused by the types of lexical/orthographic divergences seen in §3. To evaluate this hypothesis, we compare the behavior of existing language identifiers on our subcorpora.

We test *langid.py* as well as the output of Twitter's in-house identifier, whose predictions are included in a tweet's metadata (from 2013, the time of data collection); the latter may give a language code or a missing value (*unk* or an empty/null value). We record the proportion of non-English predictions by these systems; *Twitter-1* does not consider missing values to be a non-English prediction, and *Twitter-2* does.

We noticed emojis had seemingly unintended consequences on *langid.py*'s classifications, so removed all emojis by characters from the relevant Unicode ranges. We also removed @-mentions.

User-level analysis We begin by comparing the classifiers' behavior on the AA- and white-aligned corpora. Of the AA-aligned tweets, 13.2% were classified by *langid.py* as non-English; in contrast, 7.6% of white-aligned tweets were classified as such. We observed similar disparities for *Twitter-1* and *Twitter-2*, illustrated in Table 3.

It turns out these "non-English" tweets are, for the most part, actually English. We sampled and annotated 50 tweets from the tweets classified as non-English by each run. Of these 300 tweets, only 3 could be unambiguously identified as written in a language other than English.

Message-level analysis We examine how a message's likelihood of being classified as non-English varies with its posterior probability of AA. As in §3.4, we split all messages into deciles based on the messages' posterior probability of AA, and predicted language identifications on 200,000 sampled messages from each decile.

For all three systems, the proportion of messages classified as non-English increases steadily as the messages' posterior probabilities of AA increase. As before, we sampled and annotated from the tweets classified as non-English, sampling 50 tweets from each decile for each of the three systems. Of the 1500 sampled tweets, only 13 ($\sim$0.87%) could be unambiguously identified as being in a language other than English.

4.2 Adapting Language Identification for AAE

Natural language processing tools can be improved to better support dialects; for example, Jørgensen et al. (2016) use domain adaptation methods to improve POS tagging on AAE corpora. In this section, we contribute a fix to language identification to correctly identify AAE and other social media messages as English.

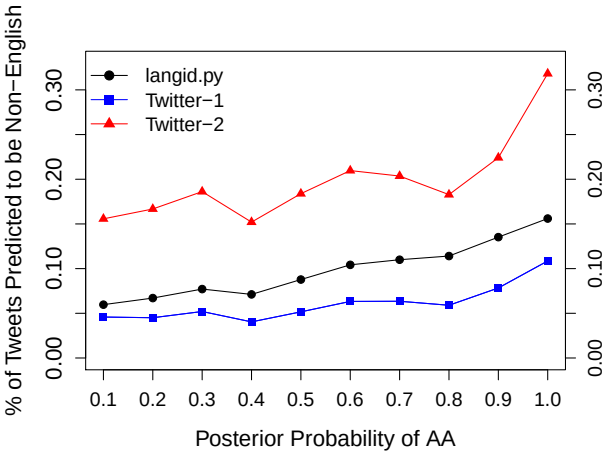


Figure 3: Proportion of tweets classified as non-English by messages' posterior probability of AA. On the x-axis, 0.1 refers to the decile [0, 0.1).

4.2.1 Ensemble Classifier

We observed that messages where our model infers a high probability of AAE, white-aligned, or "Hispanic"-aligned language almost always are written in English; therefore we construct a simple ensemble classifier by combining it with *langid.py*.

For a new message $\vec{w}$, we predict its demographic-language proportions $\hat{\theta}$ via posterior inference with our trained model, given a symmetric α prior over demographic-topic proportions (see appendix for details). The ensemble classifier, given a message, is as follows:

- Calculate langid.py's prediction $\hat{y}$.
- If $\hat{y}$ is English, accept it as English.
- If $\hat{y}$ is non-English, and at least one of the message's tokens are in demographic model's vocabulary: Infer $\hat{\theta}$ and return English only if the combined AA, Hispanic, and white posterior probabilities are at least 0.9. Otherwise return the non-English $\hat{y}$ decision.

Another way to view this method is that we are effectively training a system on an extended Twitter-specific English language corpus softly labeled by our system's posterior inference; in this respect, it is related to efforts to collect new language-specific Twitter corpora (Bergsma et al., 2012) or minority language data from the web (Ghani et al., 2001).

4.2.2 Evaluation

Our analysis from §4.1 indicates that this method would correct erroneous false negatives for AAE messages in the training set for the model. We further confirm this by testing the classifier on a sample of 2.2 million geolocated tweets sent in the U.S. in 2014, which are not in the training set.

Message set	*langid.py*	Ensemble
High AA	80.1%	99.5%
High White	96.8%	99.9%
General	*88.0%*	*93.4%*

Table 4: Imputed recall of English messages in 2014 messages. For the *General* set these are an approximation; see text.

In addition to performance on the entire sample, we examine our classifier's performance on messages whose posterior probability of using AA- or white-associated terms was greater than 0.8 within the sample, which in this section we will call high AA and high white messages, respectively. Our classifier's precision is high across the board, at 100% across manually annotated samples of 200 messages from each sample.[10] Since we are concerned about the system's overall recall, we impute recall (Table 4) by assuming that all high AA and high white messages are indeed English. Recall for *langid.py* alone is calculated by $\frac{n}{N}$, where n is the number of messages predicted to be English by *langid.py*, and N is the total number of messages in the set. (This is the complement of Table 3, except evaluated on the test set.) We estimate the ensemble's recall as $\frac{n+m}{N}$, where $m = (n_{flip})P(\text{English} \mid flip)$ is the expected number of correctly changed classifications (from non-English to English) by the ensemble and the second term is the precision (estimated as 1.0). We observe the baseline system has considerable difference in recall between the groups which is solved by the ensemble.

We also apply the same calculation to the general set of all 2.2 million messages; the baseline classifies 88% as English. This is a less accurate approximation of recall since we have observed a substantial presence of non-English messages. The ensemble classifies an additional 5.4% of the messages as English; since these are all (or nearly all) correct, this

[10] We annotated 600 messages as English, not English, or not applicable, from 200 sampled each from general, high AA, and high white messages. Ambiguous tweets which were too short (e.g. "Gm") or contained only named entities (e.g. "Tennessee") were excluded from the final calculations. The resulting samples have 197/197, 198/198, and 200/200 correct English classifications, respectively.

reflects at least a 5.4% gain to recall.

5 Dependency Parser Evaluation

Given the lexical and syntactic variation of AAE compared to SAE, we hypothesize that syntactic analysis tools also have differential accuracy. Jørgensen et al. (2015) demonstrate this for part-of-speech tagging, finding that SAE-trained taggers had disparate accuracy on AAE versus non-AAE tweets.

We assess a publicly available syntactic dependency parser on our AAE and white-aligned corpora. Syntactic parsing for tweets has received some research attention; Foster et al. (2011) create a corpus of constituent trees for English tweets, and Kong et al. (2014)'s *Tweeboparser* is trained on a Twitter corpus annotated with a customized unlabeled dependency formalism; since its data was uniformly sampled from tweets, we expect it may have low disparity between demographic groups.

We focus on widely used syntactic representations, testing the *SyntaxNet* neural network-based dependency parser (Andor et al., 2016),[11] which reports state-of-the-art results, including for web corpora. We evaluate it against a new manual annotation of 200 messages, 100 randomly sampled from each of the AA- and white-aligned corpora described in §2.3.

SyntaxNet outputs grammatical relations conforming to the Stanford Dependencies (SD) system (de Marneffe and Manning, 2008), which we used to annotate messages using *Brat*,[12] comparing to predicted parses for reference. Message order was randomized and demographic inferences were hidden from the annotator. To increase statistical power relative to annotation effort, we developed a partial annotation approach to only annotate edges for the root word of the first major sentence in a message. Generally, we found that that SD worked well as a descriptive formalism for tweets' syntax; we describe handling of AAE and Internet-specific non-standard issues in the appendix. We evaluate labeled recall of the annotated edges for each message set:

Parser	AA	Wh.	Difference
SyntaxNet	64.0 (2.5)	80.4 (2.2)	16.3 (3.4)
CoreNLP	50.0 (2.7)	71.0 (2.5)	21.0 (3.7)

Bootstrapped standard errors (from 10,000 message resamplings) are in parentheses; differences are statistically significant ($p < 10^{-6}$ in both cases).

The white-aligned accuracy rate of 80.4% is broadly in line with previous work (compare to the parser's unlabeled accuracy of 89% on English Web Treebank full annotations), but parse quality is much worse on AAE tweets at 64.0%. We test the Stanford CoreNLP neural network dependency parser (Chen and Manning, 2014) using the *english_SD* model that outputs this formalism;[13] its disparity is worse. Soni et al. (2014) used a similar parser[14] on Twitter text; our analysis suggests this approach may suffer from errors caused by the parser.

6 Discussion and Conclusion

We have presented a distantly supervised probabilistic model that employs demographic correlations of a dialect and its speaker communities to uncover dialectal language on Twitter. Our model can also close the gap between NLP tools' performance on dialectal and standard text.

This represents a case study in dialect *identification*, *characterization*, and ultimately language technology *adaptation* for the dialect. In the case of AAE, dialect identification is greatly assisted since AAE speakers are strongly associated with a demographic group for which highly accurate governmental records (the U.S. Census) exist, which we leverage to help identify speaker communities. The notion of non-standard dialectal language implies that the dialect is underrepresented or underrecognized in some way, and thus should be inherently difficult to collect data on; and of course, *many* other language communities and groups are not necessarily officially recognized. An interesting direction for future research would be to combine distant supervision with unsupervised linguistic models to automatically uncover such underrecognized dialectal language.

Acknowledgments: We thank Jacob Eisenstein, Taylor Jones, Anna Jørgensen, Dirk Hovy, and the anonymous reviewers for discussion and feedback.

[11] Using the publicly available *mcparseface* model: https://github.com/tensorflow/models/tree/master/syntaxnet

[12] http://brat.nlplab.org/

[13] *pos,depparse* options in version 2015-04-20, using tokenizations output by SyntaxNet.

[14] The older Stanford *englishPCFG* model with dependency transform (via pers. comm.).

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. How noisy social media text, how diffrnt social media sources? In *International Joint Conference on Natural Language Processing*, pages 356–364, 2013.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. Language identification for creating language-specific Twitter collections. In *Proceedings of the Second Workshop on Language in Social Media*, pages 65–74. Association for Computational Linguistics, 2012.

Marianne Bertrand and Sendhil Mullainathan. Are Emily and Greg more employable than Lakisha and Jamal? A field experiment on labor market discrimination. Technical report, National Bureau of Economic Research, 2003.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1082.

Ryan Cotterell, Adithya Renduchintala, Naomi Saphra, and Chris Callison-Burch. An Algerian Arabic-French code-switched corpus. In *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, page 34, 2014.

M. C. de Marneffe and C. D. Manning. *Stanford typed dependencies manual*. Technical report, last revised April 2015 edition, 2008.

Gabriel Doyle. Mapping dialectal variation by querying social media. In *Proceedings of EACL*, pages 98–106, 2014.

Jacob Eisenstein. Phonological factors in social media writing. In *Proc. of the Workshop on Language Analysis in Social Media*, pages 11–19, 2013.

Jacob Eisenstein. Identifying regional dialects in online social media. In C. Boberg, J. Nerbonne, and D. Watt, editors, *Handbook of Dialectology*. Wiley, 2015.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. Sparse additive generative models of text. In *Proceedings of ICML*, pages 1041–1048, 2011a.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1365–1374. Association for Computational Linguistics, 2011b.

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. Aida: Identifying code switching in informal Arabic text. *Proceedings of EMNLP 2014*, page 94, 2014.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. #hardtoparse: POS tagging and parsing the Twitterverse. In *Proc. of AAAI-11 Workshop on Analysing Microtext*, 2011.

Rayid Ghani, Rosie Jones, and Dunja Mladenić. Mining the web to create minority language corpora. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 279–286. ACM, 2001.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeff Flanigan, and Noah A. Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics, 2011.

Lisa J. Green. *African American English: A Linguistic Introduction*. Cambridge University Press, 2002.

T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of*

Sciences of the United States of America, 101 (Suppl 1):5228, 2004.

Dirk Hovy and L. Shannon Spruit. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-2096. URL http://aclweb.org/anthology/P16-2096.

Yuan Huang, Diansheng Guo, Alice Kasakoff, and Jack Grieve. Understanding US regional linguistic variation with Twitter data analysis. *Computers, Environment and Urban Systems*, 2015.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. Reconsidering language identification for written language resources. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA), 2006. URL http://aclweb.org/anthology/L06-1274.

Taylor Jones. Toward a description of African American Vernacular English dialect regions using "Black Twitter". *American Speech*, 90(4): 403–440, 2015.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. Learning a POS tagger for AAVE-like language. In *Proceedings of NAACL*. Association for Computational Linguistics, 2016.

Anna Katrine Jørgensen, Dirk Hovy, and Anders Søgaard. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18, 2015.

Ben King and Steven P Abney. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of HLT-NAACL*, pages 1110–1119, 2013.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1108.

M. Lui and T. Baldwin. langid. py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012), Demo Session, Jeju, Republic of Korea*, 2012. URL http://www.aclweb.org/anthology-new/P/P12/P12-3005.pdf.

Teresa Lynn, Kevin Scannell, and Eimear Maguire. Minority language Twitter: Part-of-speech tagging and analysis of Irish tweets. *Proceedings of ACL-IJCNLP 2015*, page 1, 2015.

Suraj Maharjan, Elizabeth Blair, Steven Bethard, and Thamar Solorio. Developing language-tagged corpora for code-switching tweets. In *The 9th Linguistic Annotation Workshop held in conjuncion with NAACL 2015*, page 72, 2015.

Shervin Malmasi, Eshrag Refaee, and Mark Dras. Arabic dialect identification using a parallel multidialectal corpus. In *International Conference of the Pacific Association for Computational Linguistics*, pages 35–53. Springer, 2015.

David Mimno and Andrew McCallum. Topic models conditioned on arbitrary features with Dirichlet-Multinomial regression. In *Uncertainty in Artificial Intelligence*, pages 411–418, 2008.

B. L. Monroe, M. P. Colaresi, and K. M. Quinn. Fightin' Words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372, 2008.

Fred Morstatter, Jrgen Pfeffer, Huan Liu, and Kathleen Carley. Is the sample good enough? Comparing data from twitter's streaming api with Twitter's Firehose. In *International AAAI Conference on Weblogs and Social Media*, 2013. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6071.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. A mixture model of demographic lexical variation. In *NIPS Workshop on Machine Learning for Social Computing*, 2010.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for on-

line conversational text with word clusters. In *Proceedings of NAACL*, 2013.

Umashanthi Pavalanathan and Jacob Eisenstein. Confounds and consequences in geotagged Twitter data. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, September 2015. URL http://www.aclweb.org/anthology/D/D15/D15-1256.pdf.

Daniel Ramage, Christopher D. Manning, and Susan Dumais. Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 457–465, 2011.

John Russell Rickford. *African American Vernacular English: Features, Evolution, Educational Implications*. Wiley-Blackwell, 1999.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, and Edoardo M Airoldi. The structural topic model and applied social science. In *Advances in Neural Information Processing Systems Workshop on Topic Models: Computation, Application, and Evaluation*, 2013.

Mike Rosner and Paulseph-John Farrugia. A tagging algorithm for mixed language identification in a noisy domain. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

Thamar Solorio and Yang Liu. Learning to predict code-switching points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 973–981. Association for Computational Linguistics, 2008.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-3907.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. Modeling factuality judgments in social media text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 415–420, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-2068.

Ian Stewart. Now we stronger than ever: African-American syntax in Twitter. *Proceedings of EACL*, page 31, 2014.

Latanya Sweeney. Discrimination in online ad delivery. *ACM Queue*, 11(3):10, 2013.

Matt Taddy. Multinomial inverse regression for text analysis. *Journal of the American Statistical Association*, 108(503):755–770, 2013.

Marcos Zampieri, Binyam Gebrekidan Gebre, and Sascha Diwersy. N-gram language models and pos distribution for the identification of Spanish varieties. *Proceedings of TALN2013*, pages 580–587, 2013.

Arkaitz Zubiaga, Inaki San Vincente, Pablo Gamallo, Jose Ramom Pichel, Inaki Algeria, Nora Aranberri, Aitzol Ezeiza, and Victor Fresno. Overview of TweetLID: Tweet language identification at SEPLN 2014. In *Proceedings of the Tweet Language Identification Workshop*, Girona, Spain, September 2014. Spanish Society for Natural Language Processing. URL http://ceur-ws.org/Vol-1228/.

Understanding Language Preference for Expression of Opinion and Sentiment: What do Hindi-English Speakers do on Twitter?

Koustav Rudra
IIT Kharagpur, India
koustav.rudra@cse.iitkgp.ernet.in

Shruti Rijhwani*
Carnegie Mellon University,
Pittsburgh, Pennsylvania
srijhwan@cs.cmu.edu

Rafiya Begum
Microsoft Research Labs,
Bangalore, India
t-rafbeg@microsoft.com

Kalika Bali
Microsoft Research Labs,
Bangalore, India
kalikab@microsoft.com

Monojit Choudhury
Microsoft Research Labs,
Bangalore, India
monojitc@microsoft.com

Niloy Ganguly
IIT Kharagpur, India
niloy@cse.iitkgp.ernet.in

Abstract

Linguistic research on multilingual societies has indicated that there is usually a preferred language for expression of emotion and sentiment (Dewaele, 2010). Paucity of data has limited such studies to participant interviews and speech transcriptions from small groups of speakers. In this paper, we report a study on 430,000 unique tweets from Indian users, specifically Hindi-English bilinguals, to understand the language of preference, if any, for expressing opinion and sentiment. To this end, we develop classifiers for opinion detection in these languages, and further classifying opinionated tweets into positive, negative and neutral sentiments. Our study indicates that Hindi (i.e., the native language) is preferred over English for expression of negative opinion and swearing. As an aside, we explore some common pragmatic functions of code-switching through sentiment detection.

1 Introduction

The pattern of language use in a multilingual society is a complex interplay of socio-linguistic, discursive and pragmatic factors. Sometimes speakers have a preference for a particular language for certain conversational and discourse settings; on other occasions, there is fluid alteration between two or more languages in a single conversation, also known as *Code-switching* (CS) or *Code-mixing*[1]. Understanding and characterizing language preference in multilingual societies has been the subject matter of linguistic inquiry for over half a century (see Milroy and Muysken (1995) for an overview).

Conversational phenomena such as CS were observed only in speech and therefore, all previous studies are based on data collected from a small set of speakers or from interviews. With the growing popularity of social media, we now have an abundance of conversation-like data that exhibit CS and other speech phenomena, hitherto unseen in text (Bali et al., 2014). Leveraging such data from Twitter, we conduct a large-scale study on language preference, if any, for the expression of opinion and sentiment by Hindi-English (Hi-En) bilinguals.

We first build a corpus of 430,000 unique India-specific tweets across four domains (sports, entertainment, politics and current events) and automatically classify the tweets by their language: English, Hindi and Hi-En CS. We then develop an opinion detector for each language class to further categorize them into opinionated and non-opinionated tweets. Sentiment detectors further classify the opinionated tweets as positive, negative or neutral. Our study shows that there is a strong preference towards Hindi (i.e. the native language or L1) over English (L2) for expression of negative opinion. The effect is clearly visible in CS tweets, where a switch from English to Hindi is often correlated with a switch from a positive to negative sentiment. This is referred to as the *polarity–switch* function of CS (Sanchez, 1983). Using the same experimental technique, we also explore other pragmatic functions of CS, such as *reinforcement* and *narrative–evaluative*.

* This work was done when the author was a Research Fellow at Microsoft Research Lab India.

[1] Although some linguists differentiate between Code-switching and Code-mixing, this paper will use the two terms interchangeably.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1131–1141,
Austin, Texas, November 1-5, 2016.

Apart from being the first large-scale quantitative study of language preference in multilingual societies, this work also has several other contributions: (a) We develop one of the first opinion and sentiment classifiers for Romanized Hindi and CS Hi-En tweets with higher accuracy than the only known previous attempt (Sharma et al., 2015b). (b) We present a novel methodology for automatically detecting pragmatic functions of code-switching through opinion and sentiment detection.

The rest of the paper is organized as follows: Sec. 2 introduces language preference, functions of CS and Hindi-English bilingualism on the web. Sec. 3 formulates the problem and presents the fundamental questions that this paper seeks to answer. Sec. 4 and 5 discuss dataset creation and opinion and sentiment detection techniques respectively. Sec. 6 evaluates the hypotheses in light of the observations on the tweet corpus. We conclude in Sec. 7, and raise some interesting sociolinguistic questions for future studies.

2 Background and Related Work

In order to situate the questions addressed in our work in existing literature, we present a brief overview of the past research in pragmatic and discursive analysis of code-switching, and specifically, on language preference for emotional expression. A primer to Hi-En bilingualism and its presence in social media shall follow.

2.1 CS Functions and Language Preference

In multilingual communities, where there are more than one linguistic channels for information exchange, the choice of the channel depends on a variety of factors, and is usually unpredictable (Auer, 1995). Nevertheless, linguistic studies point out certain frequently-observed patterns. For instance, certain speech activities might be exclusively or more commonly related to a certain language choice (e.g. Fishman (1971) reports use of English for professional purposes and Spanish for informal chat for English-Spanish bilinguals from Puerto Rico). Apart from association between such conversational contexts and language preference, language alteration is often found to be used as a signaling device to imply certain pragmatic functions (Barredo, 1997; Sanchez, 1983; Nishimura, 1995; Maschler, 1991; Maschler, 1994) such as: (a) reported speech (b) narrative to evaluative switch (c) reiterations or emphasis (d) topic shift (e) puns and language play (f) topic/comment structuring etc. Attempts of predicting the preferred language, or even exhaustively listing such functions, have failed. However, linguists agree that language alteration in multilingual communities is not a random process.

Of specific interest to us are the studies on language preference for expression of emotions. Through large-scale interviews and two decades of research, Dewaele (2004; 2010) argued that for most multilinguals, L1 (the dominant language, which is often, but not always, the native or mother tongue) is the language preference for emotions, which include emotional inner speech, swearing and even emotional conversations. Dewaele argues that emotionally charged words in L1 elicit stronger emotions than those in other languages, and hence L1 is preferred for emotion expression.

2.2 Hindi-English Bilingualism

Around 125 million people in India speak English, half of whom have Hindi as their mother-tongue. The large proportion of the remaining half, especially those residing in the metropolitan cities, also know at least some Hindi. This makes Hi-En code-switching, commonly called *Hinglish*, extremely widespread in India. There is historical attestation, as well as recent studies on the growing use of Hinglish in general conversation, and in entertainment and media (see Parshad et al. (2016) and references therein). Several recent studies (Bali et al., 2014; Barman et al., 2014; Solorio et al., 2014; Sequiera et al., 2015) also provide evidence of Hinglish and other instances of CS on online social media such as Twitter and Facebook. In a Facebook dataset analyzed by Bali et al. (2014), almost all sufficiently long conversation threads were found to be multilingual, and as much as 17% of the comments had CS. This study also indicates that on online social media, Hindi is seldom written in the Devanagari script. Instead, loose Roman transliteration, or Romanized Hindi, is common, especially when users code-switch between Hindi and English.

While there has been some effort towards computational processing of CS text (Solorio and Liu, 2008; Solorio and Liu, 2010; Vyas et al., 2014; Peng

et al., 2014), to the best of our knowledge, there has been no study on automatic identification of functional aspects of CS or any large-scale, data-driven study of language preference. The current study adds to the growing repertoire of work on quantitative analysis of social media data for understanding socio-linguistic and pragmatic issues, such as detection of depression (De Choudhury et al., 2013), politeness (Danescu-Niculescu-Mizil et al., 2013), speech acts (Vosoughi and Roy, 2016), and social status (Tchokni et al., 2014).

3 Problem Formulation

Along the lines of (Dewaele, 2010), we ask the following question: *Is there a preferred language for expression of opinion and sentiment by the Hi-En bilinguals on Twitter?*

3.1 Definitions

More formally, let $\Lambda = \{h, e, m\}$ be the set of languages: Hindi (h), English (e) and Mixed (m), i.e., code-switched. Let $\Sigma = \{d, r\}$, be the set of scripts:[2] Devanagari (d) and Roman (r). Let us further introduce a set of sentiments, $\diamondsuit = \{+, -, 0, \otimes\}$, where $+$, $-$ and 0 respectively denote utterances with positive, negative and neutral opinions. $\otimes$ denote non-opinionated (like factual) texts.

Let $T = \{t_1, t_2, \ldots t_{|T|}\}$ be a set of tweets (or any text) generated by Hi-En bilinguals. We define:

- $\lambda(T)$, $\sigma(T)$ and $\diamond(T)$ as the subsets of T that respectively contain all tweets in language λ, script σ and sentiment $\diamond$.
- $\lambda\sigma\diamond(T) = \lambda(T) \cap \sigma(T) \cap \diamond(T)$. Likewise, we also define $\lambda\diamond(T) = \lambda(T) \cap \diamond(T)$, $\lambda\sigma(T) = \lambda(T) \cap \sigma(T)$ and $\sigma\diamond(T) = \sigma(T) \cap \diamond(T)$.

The preference towards a language-script pair $\lambda\sigma$ for expressing a type of sentiment $\diamond$ is given by the probability

$$pr(\lambda\sigma|\diamond; T) = \frac{pr(\diamond|\lambda\sigma; T)pr(\lambda\sigma|T)}{pr(\diamond|T)} \quad (1)$$

However, pr($\lambda\sigma$), which defines the prior probability of choosing $\lambda\sigma$ for a tweet is dependent on a large number of socio-linguistic parameters beyond sentiment. For instance, on social media, English is overwhelmingly more common than any Indic language (Bali et al., 2014). This is because (a) English tweets come from a large number of users apart from Hi-En bilinguals and (b) English is the preferred language for tweeting even for Hi-En bilinguals because it expands the target audience of the tweet by manifolds. The preference of $\lambda\sigma$ for expressing $\diamond$, therefore, can be quantified as:

$$pr(\diamond|\lambda\sigma; T) = \frac{|\lambda\sigma\diamond(T)|}{|\lambda\sigma(T)|} \quad (2)$$

We say $\lambda\sigma$ is the preferred language-script choice over $\lambda'\sigma'$ for expressing sentiment $\diamond$ if and only if

$$pr(\diamond|\lambda\sigma; T) > pr(\diamond|\lambda'\sigma'; T) \quad (3)$$

The strength of the preference is directly proportionate the ratio of the probabilities: $pr(\diamond|\lambda\sigma; T)/pr(\diamond|\lambda'\sigma'; T)$. An alternative but related way of characterizing the preference is through comparing the odds of choosing a sentiment type $\diamond$ to its polar opposite - $\diamond'$. We say, $\lambda\sigma$ is the preferred language-script pair for expressing $\diamond$, if

$$\frac{pr(\diamond|\lambda\sigma; T)}{pr(\diamond'|\lambda\sigma; T)} > \frac{pr(\diamond|\lambda'\sigma'; T)}{pr(\diamond'|\lambda'\sigma'; T)} \quad (4)$$

3.2 Hypotheses

Now we can formally define the two hypotheses, we intend to test here.

Hypothesis I: For Hi-En bilinguals, Hindi is the preferred language for expression of opinion on Twitter. Therefore, we expect

$$pr(\{+, -, 0\}|hd; T) > pr(\{+, -, 0\}|er; T) \quad (5)$$

$$\text{i.e.,} \quad pr(\otimes|hd; T) < pr(\otimes|er; T) \quad (6)$$

And similarly,

$$pr(\otimes|hr; T) < pr(\otimes|er; T) \quad (7)$$

Hypothesis II: For Hi-En bilinguals, Hindi is the preferred language for expression of negative sentiment. Therefore,

$$pr(-|hd; T) \approx pr(-|hr; T) > pr(-|er; T) \quad (8)$$

[2] Tweets in mixed script are rare and hence we do not include a symbol for it, though the framework does not preclude such possibilities.

In particular, we would like to hypothesize that the odds of choosing Hindi for negative over positive is really high compared to the odds for English. I.e.,

$$\frac{pr(-|hd;T)}{pr(+|hd;T)} \approx \frac{pr(-|hr;T)}{pr(+|hr;T)} > \frac{pr(-|er;T)}{pr(+|er;T)} \quad (9)$$

A special case of the above hypotheses arise in the context of code-mixing, i.e., for the set $mr(T)$. Since the mixed tweets certainly come from proficient bilinguals and have both Hi and En fragments, we can reformulate our hypotheses at a tweet level. Let $m^hr(T)$ and $m^er(T)$ respectively denote the set of Hi and En fragments in $mr(T)$.

Hypothesis Ia: Hindi is the preferred language for expression of opinion in Hi-En code-mixed tweets. Therefore, we expect

$$\text{i.e.,} \quad pr(\otimes|m^hr;T) < pr(\otimes|m^er;T) \quad (10)$$

Hypothesis IIa: Hindi is the preferred language for expression of negative sentiment in Hi-En code-switched tweets. Therefore,

$$pr(-|m^hr;T) > pr(-|m^er;T) \quad (11)$$

$$\frac{pr(-|m^hr;T)}{pr(+|m^hr;T)} > \frac{pr(-|m^er;T)}{pr(+|m^er;T)} \quad (12)$$

Likewise, the above hypotheses also apply for the Devanagari script, though for technical reasons, we do not test them here.

Besides comparing aggregate statistics on $mr(T)$, it is also interesting to look at the sentiment of $m^hr(t_i)$ and $m^er(t_i)$ for each tweet t_i. In particular, for every pair of $\diamond \neq \diamond'$, we want to study the fraction of tweets in $mr(T)$ where $m^hr(t_i)$ has sentiment $\diamond$ and $m^er(t_i)$ has $\diamond'$. Let this fraction be $pr(h\diamond \leftrightarrow e\diamond'; mr(T))$. Under "no-preference for language" (i.e., the null) hypothesis, we would expect $pr(h\diamond \leftrightarrow e\diamond'; mr(T)) \approx pr(h\diamond' \leftrightarrow e\diamond; mr(T))$. However, if $pr(h\diamond \leftrightarrow \diamond'; mr(T))$ is significantly higher than $pr(h\diamond' \leftrightarrow e\diamond; mr(T))$, it means that speakers prefer to switch from English to Hindi when they want to express a sentiment $\diamond$ and vice versa.

Pragmatic Functions of Code-Switching: When native speakers tend to switch from Hindi to English when they switch from an expression with sentiment $\diamond$ to one with $\diamond'$, or in other words $\diamond \leftrightarrow \diamond'$, we say this is an observed pragmatic function of code-switching between Hindi and English (note that the order of the languages is important), if and only if

$$\frac{pr(h\diamond \leftrightarrow e\diamond'; mr(T))}{pr(h\diamond' \leftrightarrow e\diamond; mr(T))} > 1 \quad (13)$$

Topic (# tweets): Hashtags
Sports (188K): #IndvsPak, #IndvsUae, #IndvsSa
Movies (82K): #MSG3successfulweeks, #MSGincinemas, #BlockbusterMSG, #Shamitabh, #PK
Politics (92K): #DelhiDecides, #RahulonlLeave, #AAPStorm, #AAPSweep
Current Events (68K): #RailBudget2015, #Beefban, #LandAcquisitionBill, #UnionBudget2015

Table 1: Hashtags used and number of tweets collected

3.3 A Note on Statistical Significance

All the statistics defined here are likelihoods; Equations 9, 12 and 13, in particular, state our hypothesis in the form of the *Likelihood Ratio Test*. However, the true classes λ and $\diamond$ are unknown; we predict the class labels using automatic language and sentiment detection techniques that have non-negligible errors. Under such a situation, the likelihoods cannot be considered as true *test statistics*, and consequently, hypothesis testing cannot be done per se. Nevertheless, we can use these as descriptive statistics and investigate the status of the aforementioned hypotheses.

4 Datasets

We collected tweets with certain India-specific hashtags (Table 1) using the Twitter Search API (Twi, 2015b) over three months (December 2014 – February 2015). In this paper, we use tweets in Devanagari script Hindi (hd), and Roman script English (er), Hindi (hr) and Hi-En Mixed (mr). English and mixed tweets written in Devanagari are extremely rare (Bali et al., 2014) and we do not study them here. We filter out tweets labeled by the Twitter API (Twi, 2015a) as German, Spanish, French, Portuguese, Turkish, and all non-Roman script languages (except Hindi).

We experiment on the following different corpora:

$\boldsymbol{T_{All}}$: All tweets after filtering. This corpus contains 430,000 unique tweets posted by 1,25,396 unique users.

$\boldsymbol{T_{BL}}$: Tweets from users who are certainly Hi-En bilinguals, which are approximately 55% (240,000) of the tweets in $\boldsymbol{T_{All}}$. We define a user to be a Hi-En bilingual if there is at least one mr tweet from the user, or if the user has tweeted at least once in Hindi (hd or hr) and once in English (er).

$\boldsymbol{T_{spo}, T_{mov}, T_{pol}, T_{eve}}$: Topic-wise corpora for sports, movies, politics and events (Table 1).

$\boldsymbol{T_{CS}}$: Tweets with inter-sentential CS. We define these as tweets containing at least one sequence of 5 contiguous Hindi words and one sequence of 5 contiguous English words. The corpus has 3,357 tweets.

SAC: 1000 monolingual tweets (er, hr, hd) and 260 mixed (mr) tweets manually annotated with sentiment and opinion labels. These were annotated by two linguists, both fluent Hi-En speakers. The annotators first checked whether the tweet is opinionated or $\otimes$ and then identified polarity of the opinionated tweets ($+$, $-$ or 0). Thus, the tweets are classified into the four classes in the set $\diamond$. If a tweet contains both opinion and $\otimes$, each fragment was individually annotated. The inter-annotator agreement is 77.5% ($\kappa = 0.59$) for opinion annotation and 68.4% ($\kappa = 0.64$) over all four classes. A third linguist independently corrected the disagreements.

$\mathbf{LLC_{Test}}$: 141 er, 137 hr, and 241 mr tweets annotated by a Hi-En bilingual form the test set for the Language Labeling system (Sec. 5.1).

SAC and $\mathbf{LLC_{Test}}$ can be downloaded and used for research purposes[3].

Note that apart from **SAC** and $\mathbf{LLC_{Test}}$, all corpora are subsets of $\boldsymbol{T_{All}}$. For generalizability of our observations, it is important to ensure that the tweets in $\boldsymbol{T_{All}}$ come from a large number of users and the datasets do not over-represent a small set of users. In Figure 1, we plot the minimum fraction of users required (x-axis) to cover a certain percentage of the tweets in $\boldsymbol{T_{All}}$ (y-axis). Tweets from at least 10%, i.e., 12.5K users are needed to cover 50% of the corpus. As expected, we do observe a power-law-like distribution, where a few users contribute a large number of tweets, and a large number of users contribute a few tweets each. We believe that 12.5K users is sufficient to ensure an unbiased study.

Further, we classify the users into three specific groups (i) news channels, (ii) general users (having $\leq$ 10,000 followers), (iii) popular users or celebrities (having $>$ 10,000 followers). Interestingly, for both $\mathbf{T_{All}}$, and $\mathbf{T_{BL}}$ corpora, we observe that around 98% of all users are general, and 96% of all tweets come from such users. Hence, most observations from these corpora are expected to be representative of the average online linguistic behavior of a Hi-En bilingual.

[3] http://www.cnergres.iitkgp.ac.in/codemixing

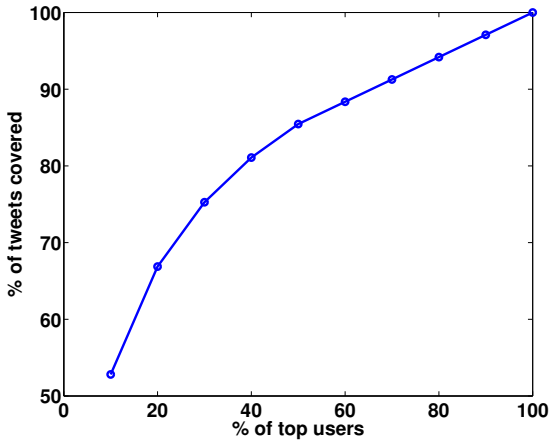


Figure 1: Distribution of cumulative % of tweets and # of users (sorted in descending order by number of tweets).

5 Method

Fig. 2 diagrammatically summarizes our experimental method. We identify the language used in each tweet before detecting opinion and sentiment.

5.1 Language Labeling

Tweets in Devanagari script are accurately detected by the Twitter API as Hindi tweets – we label these as hd, though a small fraction of them could also be md. To classify Roman script tweets as er, hr or mr, we use the system that performed best in the FIRE 2013 shared task for word-level language detection of Hi-En text (Gella et al., 2013). This system uses character n-gram features with a Maximum Entropy model for labeling each input word with a language label (either English or Hindi). We design minor modifications to the system to improve its performance on Twitter data, which are omitted here due to paucity of space.

5.2 Opinion and Sentiment Detection

Most of the existing research in opinion detection (Qadir, 2009; Brun, 2012; Rajkumar et al.,

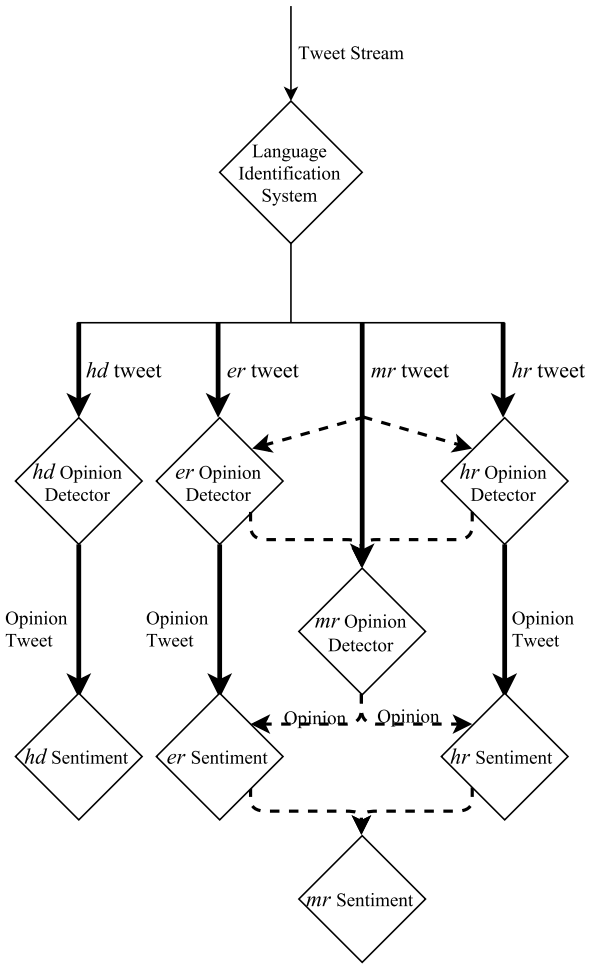


Figure 2: Overview of the experimental method.

2014) and sentiment analysis (Mohammad, 2012; Mohammad et al., 2013; Mittal et al., 2013; Rosenthal et al., 2015) focus on monolingual tweets and sentences. Recently, there has been a couple of studies on sentiment detection of code-switched tweets (Vilares et al., 2015; Sharma et al., 2015b). Sharma et al. (2015b) use Hindi SentiWordNet and normalization techniques to detect sentiment in Hi-En CS tweets.

We propose a two-step classification model. We first identify whether a tweet is opinionated or non-opinionated ($\otimes$). If the tweet is opinionated, we further classify it according to its sentiment ($+$, $-$ or 0). Fig. 2 shows the architecture of the proposed model. Two-step classification was empirically found to be better than a single four-class classifier.

We develop individual classifiers for each language class (er, hr, hd, mr) using an SVM with RBF kernel from Scikit-learn (Pedregosa et al., 2011). We use the **SAC** dataset (Sec. 4) as training data and features as described in Sec. 5.3.

5.3 Classifier Features

For opinion classification (opinion or $\otimes$), we propose a set of event-independent lexical features and Twitter-specific features. (i) **Subjective words:** Expected to be present in opinion tweets. We use lexicons from Volkova et al. (2013) for er and Bakliwal et al. (2012) for hd. We Romanize the hd lexicon for the hr classifiers (ii) **Elongated words:** Words with one character repeated more than two times, e.g. *sooo, naaahhhhi* (iii) **Exclamations:** Presence of contiguous exclamation marks (iv) **Emoticons**[4] (v) **Question marks:** Queries are generally non-opinionated. (vi) **Wh-words:** These are used to form questions (vii) **Modal verbs:** e.g. *should, could, would, cud, shud* (viii) **Excess hashtags:** Presence of more than two hashtags (ix) **Intensifiers:** Generally used to emphasize sentiment, e.g., *we shouldn't get* **too** *comfortable* (x) **Swear words**[5]**:** Prevalent in opinionated tweets, e.g. *that was a* f__ing *no ball!!!! #indvssa* (xi) **Hashtags:** Hashtags might convey user sentiment (Barbosa et al., 2012). We manually identify hashtags in our corpus that represent explicit opinion. (xii) **Domain lexicon:** For hr, & hd category tweets, we construct sentiment lexicons from 1000 manually annotated tweets. Each word or phrase in this lexicon represents $+$, or $-$, or 0 sentiment. (xiii) **Twitter user mentions** (xiv) **Pronouns:** Opinion is often in first person using pronouns like *I* and *we*.

For sentiment classification, we use emoticons, swear words, exclamation marks and elongated words as described above. We also use subjective words from various lexicons (Mohammad and Turney, 2013; Volkova et al., 2013; Bakliwal et al., 2012; Sharma et al., 2015a). Additionally, we use – (i) **Sentiment words:** From Hashtag Sentiment and Sentiment140 lexicons (Mohammad et al., 2013). We also manually annotate hashtags from our dataset that represent sentiment. (ii) **Negation:** A negated context is tweet segment that begins with a negation word and ends with a punctuation mark (Pang et al., 2002). The list of negation words are

[4]The list of emoticons was extracted from Wikipedia

[5]Swear word lexicons from *noswearing.com*, *youswear.com*

Classifier	er	hd	hr	mr
Opinion	72.6	72.0	79.9	73.5
Sentiment	64.4	61.5	62.7	63.4

Table 2: Accuracy of the opinion and sentiment classifiers. All values are in %.

taken from Christopher Potts' sentiment tutorial[6].

The mr opinion classifier uses the output from the er and hr classifiers as features (Fig. 2), along with an additional feature that represents whether the majority of the words in the tweet are Hindi or not. A similar strategy is used for mr sentiment detection.

5.4 Evaluation

We evaluated the language labeling system on the $\boldsymbol{LLC_{Test}}$ corpus, on which the precision (recall) values were $0.93(0.91)$, $0.90(0.85)$ and $0.88(0.92)$ for er, hr and mr classes respectively. The tweet-level classification accuracy was 89.8%.

The opinion and sentiment classifiers were evaluated using 10-fold cross validation on the **SAC** dataset. Table 2 details the class-wise accuracy. For comparison, we also reimplemented the dictionary and dependency-based method by Qadir (2009). The accuracy of the opinion classifier on the er tweets was found to be 65.7%, 7% lower than our system. We also compared our mr sentiment classifier with that of Sharma et al. (2015b). As their method performs two class sentiment detection ($+$ and $-$), we select such tweets from **SAC**. Their system achieves an accuracy of 68.2%, which is 4% lower than the accuracy of our system.

An analysis of the errors showed more false negatives (i.e., opinions labeled $\otimes$) than false positives in opinion classification. Sentiment misclassification is uniformly distributed.

Table 3 reports the accuracy of the opinion classifier for feature ablation experiments. For all three language-script pairs, lexicon and non-word (emoticons, elongated words, hashtags, exclamation) features are the most effective, though all features have some positive contribution towards the final accuracy of opinion detection. For hr and hd tweets, domain knowledge is significant, as shown by the 4% accuracy drop with removing the domain lexicon.

[6]http://sentiment.christopherpotts.net/lingstruc.html

Ablated Feature(s)	er	hr	hd
NONE	72.6	79.9	72.0
mention	70.1	79.3	70.8
lexicon	**68.1**	**75.9**	**66.6**
subjective	69.7	79.8	70.3
wh-words	71.0	79.3	70.1
modal verb	71.1	79.3	71.3
intensifier	71.3	76.6	69.6
slang	70.0	79.2	70.6
pronoun	71.6	79.7	70.3
domain lex.	N.A.	77.0	**67.7**
non-word	**67.7**	**75.6**	68.9

Table 3: Feature ablation experiments for the opinion classifiers. NONE represents the case when all features were used. The two smallest values (pertaining to the two most effective features) are shown in bold.

Corpus	T_{BL}	T_{All}	T_{pol}	T_{mov}
$\|er(T)\|/\|T\|$	0.65	0.79	0.76	0.70
$\|hd(T)\|/\|T\|$	0.12	0.08	0.13	0.04
$\|hr(T)\|/\|T\|$	0.08	0.05	0.05	0.09
$\|mr(T)\|/\|T\|$	0.15	0.08	0.06	0.17

Table 4: Distribution across classes in Λ

6 Experiments and Observations

In this section, we report our experiments on 430,000 unique tweets (T_{All}), and its various subsets as defined in Sec 4. First, we run the language detection system on the corpora. Table 4 shows the language-wise distribution. We see that language preference varies by topic, which is not surprising. Due to paucity of space, the correlation between language usage and topic will not be discussed at length here, but we will highlight cases where the differences are striking.

We apply the language-specific opinion and sentiment classifiers to tweets detected as the corresponding language class. In the following subsections, we empirically investigate the hypotheses.

6.1 Status of Hypotheses I and II

Table 5 shows $pr(\otimes|\lambda\sigma;T)$, $pr(-|\lambda\sigma;T)$ and $pr(-|\lambda\sigma;T)/pr(+|\lambda\sigma;T)$ for T_{All}, T_{BL} and two randomly selected topics – Movie and Politics. The statistics are fairly consistent over the corpora, with slight differences but similar trends in T_{mov}.

Statistic	$\lambda\sigma$	T_{BL}	T_{All}	T_{pol}	T_{mov}
$pr(\otimes\|\lambda\sigma;T)$	er	0.34	0.35	0.37	0.29
	hd	0.45	0.47	0.48	0.49
	hr	0.38	0.39	0.37	0.49
$pr(-\|\lambda\sigma;T)$	er	0.16	0.17	0.22	0.07
	hd	0.18	0.17	0.19	0.16
	hr	0.24	0.25	0.27	0.13
$\frac{pr(-\|\lambda\sigma;T)}{pr(+\|\lambda\sigma;T)}$	er	0.35	0.38	0.59	0.11
	hd	3.00	3.27	5.67	1.90
	hr	1.46	1.60	1.96	0.55

Table 5: Sentiment across languages: Statistics concerning hypotheses I and II.

We need the first statistic in order to investigate **Hypothesis I** (Eqs. 6 and 7), and the two latter ones for verifying **Hypothesis II** (Eqs. 8 and 9).

Contrary to Eqs. 6 and 7, for all corpora except T_{mov}, we observe the following trend:

$$pr(\otimes|hd;T) > pr(\otimes|hr;T) \geq pr(\otimes|er;T)$$

In other words, hd is more commonly preferred for expressing non-opinions than hr and er. Hypothesis I is clearly untrue for these corpora, though due to the small differences between hr and er, we cannot claim that English is the preferred language for expressing opinions. A closer scrutiny of the corpora revealed that hd tweets mostly come from official sources (news channels, political parties, production houses) and celebrities, which are mostly factual. hr tweets are from general users and show similar trends as English. Thus, in general, there seems to be no preferred language for expressing opinion by the Hi-En bilinguals on Twitter.

In the context of **Hypothesis II**, we see the general pattern (with some topic specific variations):

$$pr(-|hr;T) > pr(-|hd;T) \geq pr(-|er;T)$$

The pattern emerges even more strongly, when we look at $pr(-|\lambda\sigma;T)/pr(+|\lambda\sigma;T)$. The odds of expressing a negative opinion over positive opinion in Hindi is between 1.5 and 6 (T_{mov} exhibits a slightly different pattern but similar preference, T_{pol} shows a stronger preference towards Hindi for negative sentiment), whereas the same for English is between 0.1 and 0.6. In other words, English is more preferred for expressing positive opinion, and Hindi for negative opinion. These observations provide very strong evidence in favor of Hypothesis II.

Statistic	m^hr	m^er
$pr(\otimes\|\lambda\sigma;T_{CS})$	0.39	0.45
$pr(-\|\lambda\sigma;T_{CS})$	0.22	0.14
$pr(-\|\lambda\sigma;T_{CS})/pr(+\|\lambda\sigma;T_{CS})$	2.2	0.34

Table 6: T_{CS} statistics for testing hypotheses Ia and IIa

6.2 Status of Hypotheses Ia and IIa

Recall that **Hypothesis Ia** and **Hypothesis IIa** are essentially same as Hypotheses I and II, but applied on m^hr and m^er fragments from the T_{CS} corpus.

Table 6 reports the three statistics necessary for testing these hypotheses. $pr(\otimes|m^er;T_{CS})$ is slightly greater than $pr(\otimes|m^hr;T_{CS})$, which is what we would expect if Hypothesis Ia was true. However, since the difference is small, we view it as a trend rather than a proof of Hypothesis Ia.

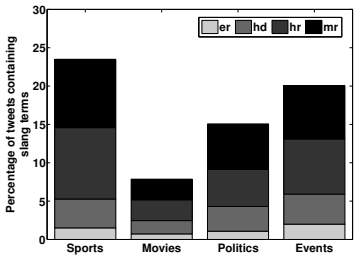
The statistics clearly show that Hypothesis IIa holds true for T_{CS}. The fraction of negative sentiment in m^hr is over 1.5 times higher than that of m^er. Further, the odds of expressing a negative sentiment in Hindi over positive sentiment in Hindi in a code-switched tweet is 6.5 times higher than the same odds for English.

6.3 Switching Functions

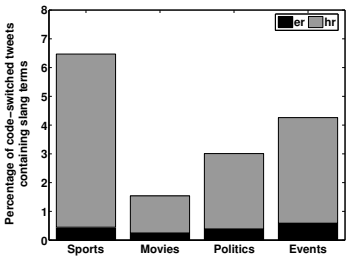
Recall that using Eq. 13 (Sec. 3), we can estimate the preference, if any, for switching to a particular language while changing the sentiment. In particular, research in socio-linguistics has shown that users often switch between languages when they switch from non-opinion ($\otimes$) to opinion ($\{+,-,0\}$). This is called the *Narrative-Evaluative* function of CS (Sanchez, 1983). This function appears in 46.1% of the tweets in $\boldsymbol{T_{CS}}$. We find that

$$\frac{pr(h\{+,-,0\} \leftrightarrow e\otimes;T_{CS})}{pr(h\otimes \leftrightarrow e\{+,-,0\};T_{CS})} = 0.86$$

which indicates that there is no preference for switching to Hindi (or English) while switching between opinion and non-opinion. This is also confirmed above in the context of hypotheses I and Ia. While switching between opinion and non-opinion in a tweet, users do switch language. However, we



(a) Abusive tweets



(b) Swearing pref. in T_{CS}

Figure 3: Distribution of swear words by language

observe no particular preference for the languages chosen for each part.

We also report two other pragmatic functions:

$$\frac{pr(h- \leftrightarrow e\{+,0,\otimes\}; T_{CS})}{pr(h\{+,0,\otimes\} \leftrightarrow e-; T_{CS})} = 1.98$$

$$\frac{pr(h- \leftrightarrow e+; T_{CS})}{pr(h+ \leftrightarrow e-; T_{CS})} = 10.27$$

The latter function is called *polarity switch*. The extremely high value for these ratios is an evidence for a strong preference towards switching language from English to Hindi while switching to negative sentiment (and switching to English when sentiment changes from negative to positive).

We also observe cases where there is a language switch, but no sentiment switch and hence, we cannot evaluate language preference using Eq. 13 (because $\diamond = \diamond'$). In $\boldsymbol{T_{CS}}$, 15.3% of the tweets show *Positive Reinforcement*, where both fragments are of positive sentiment. *Negative Reinforcement* is defined similarly and is seen in 8.7% of the tweets. Other tweets in $\boldsymbol{T_{CS}}$ likely have pragmatic functions that cannot be identified based on sentiment.

6.4 Language Preference for Swearing

Since there is evidence that the native language (Hindi, in this case) is preferred for swearing (Dewaele, 2004), we computed the fraction of tweets that contain swear words in each language class. Fig. 3a shows the distribution across topics. The languages hr and mr have a much higher fraction of abusive tweets than er and hd. Fig. 3b shows the distribution of abusive $m^h r$ and $m^e r$ fragments for tweets in T_{CS}. Interestingly, over 90% of the swear words occur in $m^h r$. Both distributions strongly suggest a preference for swearing in Hindi.

7 Conclusion

In this paper, through a large scale empirical study of nearly half a million tweets, we tried to answer a fundamental question regarding multilingualism, namely, is there a preferred language for expression of sentiment. We also looked at some of the pragmatic functions of code-switching. Our results indicate a strong preference for using Hindi, L1 for the users from whom these tweets come, for expressing negative sentiment, including swearing. However, we do not observe any particular preference towards Hindi for expressing opinions.

Previous linguistic studies (Dewaele, 2004; Dewaele, 2010) have already shown a preference for L1 for expressing emotion and swearing. However, we observe that for expressing positive emotion, English (which would be L2) is the language of preference. This raises some intriguing socio-linguistic questions. Is it the case that English being the language of aspiration in India, it is preferred for positive expression? Or is it because Hindi is specifically preferred for swearing and therefore, is the language of preference for negative emotion? How do such preferences vary across topics, users and other multilingual communities? How representative of the society is this kind of social media study? We plan to explore some of these questions in the future.

Our study also indicates that inferences drawn on multilingual societies by analyzing data in just one language (usually English), which has been the norm so far, are likely to be incorrect.

Acknowledgement

Koustav Rudra was supported by a fellowship from Tata Consultancy Services.

References

Peter Auer. 1995. The pragmatics of code-switching: a sequential approach. In Lesley Milroy and Pieter Muysken, editors, *One speaker, two languages*, pages 115–135. Cambridge University Press.

Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. 2012. Hindi subjective lexicon : A lexical resource for hindi polarity classification. In *Proc. LREC*, Austin, Texas, USA, May.

Kalika Bali, Yogarshi Vyas, Jatin Sharma, and Monojit Choudhury. 2014. "i am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook. In *Proc. First Workshop on Computational Approaches to Code Switching, EMNLP*.

Glivia A. R. Barbosa, Wagner Meira Jr, Ismael S. Silva, Raquel O. Prates, Mohammed J. Zaki, and Adriano Veloso. 2012. Characterizing the effectiveness of twitter hashtags to detect and track online population sentiment. In *Proc. ACM CHI*, Austin, Texas, USA, May.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *The 1st Workshop on Computational Approaches to Code Switching, EMNLP 2014*.

Inma Muñoa Barredo. 1997. Pragmatic functions of code-switching among Basque-Spanish bilinguals. *Retrieved on October*, 26:528–541.

Caroline Brun. 2012. Learning opinionated patterns for contextual opinion detection. In *COLING (Posters)*, pages 165–174. Citeseer.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. *Proceedings of ACL*.

Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting depression via social media. In *ICWSM*.

Jean-Marc Dewaele. 2004. Blistering barnacles! What language do multilinguals swear in?! *Estudios de Sociolinguistica*, 5:83–105.

Jean-Marc Dewaele. 2010. *Emotions in multiple languages*. Palgrave Macmillan, Basingstoke, UK.

J. A. Fishman. 1971. *Sociolinguistics*. Rowley, Newbury, MA.

Spandana Gella, Jatin Sharma, and Kalika Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description.

Yael Maschler. 1991. The language games bilinguals play: language alternation at language boundaries. *Language and communication*, 11(2):263–289.

Yael Maschler. 1994. Appreciation ha'araxa 'o ha'arasta? [valuing or admiration]. *Negotiating contrast in bilingual disagreement talk*, 14(2):207–238.

Lesley Milroy and Pieter Muysken, editors. 1995. *One speaker, two languages: Cross-disciplinary perspectives on code-switching*. Cambridge University Press.

Namita Mittal, Basant Agarwal, Garvit Chouhan, Nitin Bania, and Prateek Pareek. 2013. Sentiment analysis of hindi review based on negation and discourse relation. In *proceedings of International Joint Conference on Natural Language Processing*, pages 45–50.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. 29(3):436–465.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA, June.

Saif M Mohammad. 2012. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics.

Miwa Nishimura. 1995. A functional analysis of Japanese/English code-switching. *Journal of Pragmatics*, 23(2):157–181.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proc. EMNLP*, pages 79–86.

Rana D. Parshad, Suman Bhowmick, Vineeta Chand, Nitu Kumari, and Neha Sinha. 2016. What is India speaking? Exploring the "Hinglish" invasion. *Physica A*, 449:375–389.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nanyun Peng, Yiming Wang, and Mark Dredze. 2014. Learning polylingual topic models from code-switched social media documents. In *ACL (2)*, pages 674–679.

Ashequl Qadir. 2009. Detecting opinion sentences specific to product features in customer reviews using typed dependency relations. In *Proceedings of the Workshop on Events in Emerging Text Types*, pages 38–43. Association for Computational Linguistics.

Pujari Rajkumar, Swara Desai, Niloy Ganguly, and Pawan Goyal. 2014. A novel two-stage framework for extracting opinionated sentences from news articles. *TextGraphs-9*, page 25.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. *Proceedings of SemEval-2015*.

Rosaura Sanchez. 1983. *Chicano discourse*. Rowley, Newbury House.

Royal Sequiera, Monojit Choudhury, Parth Gupta, Paolo Rosso, Shubham Kumar, Somnath Banerjee, Sudip Kumar Naskar, Sivaji Bandyopadhyay, Gokul Chittaranjan, Amitava Das, and Kunal Chakma. 2015. Overview of fire-2015 shared task on mixed script information retrieval. In *Working Notes of FIRE*, pages 21–27.

Raksha Sharma, Pushpak Bhattacharyya, Ultimate Goal, and Hindi Senti Lexicon Statistics. 2015a. A sentiment analyzer for hindi using hindi senti lexicon.

Shashank Sharma, Pykl Srinivas, and Rakesh Chandra Balabantaray. 2015b. Text normalization of code mix and sentiment analysis. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 1468–1473. IEEE.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060. Association for Computational Linguistics.

Thamar Solorio and Yang Liu. 2010. Learning to Predict Code-Switching Points. In *Proc. EMNLP*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. *Proceedings of The First Workshop on Computational Approaches to Code Switching, EMNLP*, pages 62–72.

Simo Tchokni, D.O. Séaghdha, and Daniele Quercia. 2014. Emoticons and phrases: Status symbols in social media. In *Eighth International AAAI Conference on Weblogs and Social Media*.

2015a. GET help/languages — Twitter Developers, 8.

2015b. GET search/tweets — Twitter Developers, 8.

David Vilares, Miguel A Alonso, and Carlos Gómez-Rodrıguez. 2015. Sentiment analysis on monolingual, multilingual and code-switching twitter corpora. In *6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring Sentiment in Social Media: Bootstrapping Subjectivity Clues from Multilingual Twitter Streams. In *Proc. ACL (Vol2: Short Papers)*.

Soroush Vosoughi and Deb Roy. 2016. Tweet acts: A speech act classifier for twitter. In *Tenth International AAAI Conference on Web and Social Media*.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS Tagging of English-Hindi Code-Mixed Social Media Content. In *Proc. EMNLP*, pages 974–979.

Detecting and Characterizing Events

Allison J. B. Chaney
Princeton University
achaney@cs.princeton.edu

Hanna Wallach
Microsoft Research
wallach@microsoft.com

Matthew Connelly
Columbia University
mjc96@columbia.edu

David M. Blei
Columbia University
david.blei@columbia.edu

Abstract

Significant events are characterized by interactions between entities (such as countries, organizations, or individuals) that deviate from typical interaction patterns. Analysts, including historians, political scientists, and journalists, commonly read large quantities of text to construct an accurate picture of when and where an event happened, who was involved, and in what ways. In this paper, we present the *Capsule* model for analyzing documents to detect and characterize events of potential significance. Specifically, we develop a model based on topic modeling that distinguishes between topics that describe "business as usual" and topics that deviate from these patterns. To demonstrate this model, we analyze a corpus of over two million U.S. State Department cables from the 1970s. We provide an open-source implementation of an inference algorithm for the model and a pipeline for exploring its results.

1 Introduction

Foreign embassies of the United States government communicate with one another and with the U.S. State Department through diplomatic cables. The National Archive collects these cables in a corpus, which traces the (declassified) diplomatic history of the United States.[1] The corpus contains, for example, over two million cables sent between 1973 and 1978.

Most of these cables describe diplomatic "business as usual," such as arrangements for visiting officials, recovery of lost or stolen passports, or obtaining lists of names for meetings and conferences. For example, the embassies sent 8,635 cables during the week of April 21, 1975. Here is one, selected at random:

> Hoffman, UNESCO Secretariat, requested info from PermDel concerning an official invitation from the USG RE subject meeting scheduled 10–13 JUNE 1975, Madison, Wisconsin. Would appreciate info RE status of action to be taken in order to inform Secretariat. Hoffman communicating with Dr. John P. Klus RE list of persons to be invited.

But, hidden in the corpus are also cables about important diplomatic events—the cables and events that are most interesting to historians, political scientists, and journalists. For example, during that same week, the U.S. was in the last moments of the Vietnam War and, on April 30, 1975, lost its hold on Saigon. This marked the end of the war and induced a mass exodus of refugees. Here is one cable about this event:

> GOA program to move Vietnamese Refugees to Australia is making little progress and probably will not cover more than 100-200 persons. Press comment on smallness of program has recognized difficulty of getting Vietnamese out of Saigon, but "Canberra Times" Apr 25 sharply critical of government's performance. [...] Labor government clearly hopes whole matter will somehow disappear.

Our goal in this paper is to develop a tool to help historians, political scientists, and journalists wade

[1] The National Archives' corpus also includes messages sent by diplomatic pouch; however, for brevity, and at the risk of being imprecise, we also refer to these messages as "cables."

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1142–1152,
Austin, Texas, November 1-5, 2016.

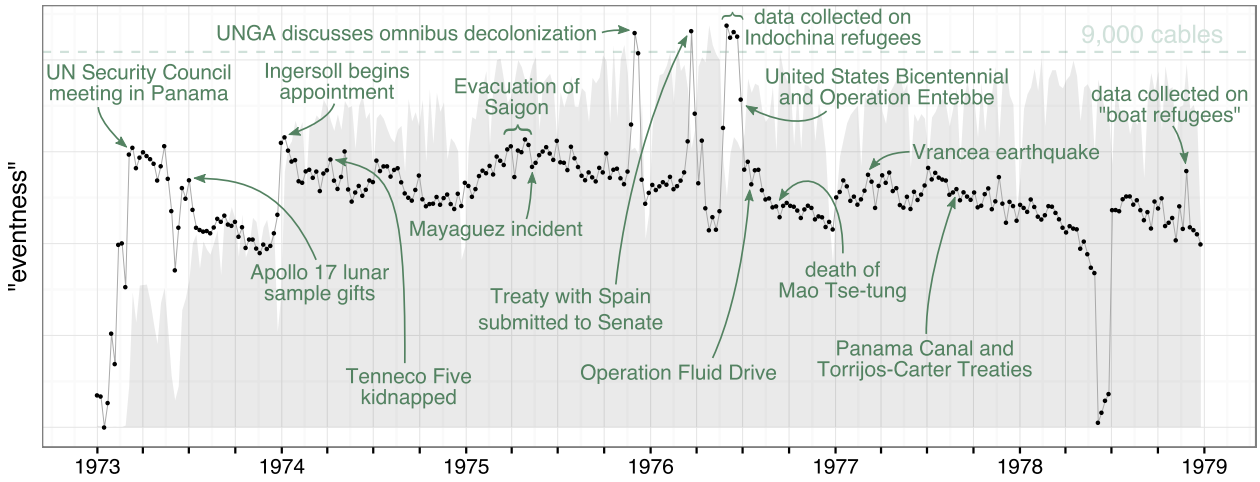


Figure 1: Capsule's analysis (described in detail in section 5) of two million cables from the National Archives' corpus. The y-axis represents a loose measure of "eventness" (equation (5)). The gray background depicts the number of cables sent over time.

through corpora of documents to find potentially significant events and the primary sources around them. We present *Capsule*, a probabilistic model for detecting and characterizing important events, such as the fall of Saigon, in large corpora of historical communication, such as diplomatic cables from the 1970s.

Figure 1 illustrates Capsule's analysis of two million cables from the National Archives' corpus. The y-axis represents "eventness," a loose measure of how strongly a week's cables deviate from typical diplomatic "business as usual" to discuss some matter that is common to many embassies. (We describe this measure of "eventness" in detail in section 3.)

This figure shows that Capsule detects many well-known events between 1973 and 1978, including the fall of Saigon (April 30, 1975) and the death of Mao Tse-tung (September 9, 1976). Capsule also uncovers obscure, but significant, events that have largely escaped the attention of scholars, such as when the U.S. defended its control of the Panama Canal before the United Nations Security Council (March 19, 1973). Capsule therefore provides a new way to detect and characterize historical moments that may be of interest to historians, political scientists, and journalists.

The intuition behind Capsule is this: Embassies write cables throughout the year, usually describing typical diplomatic business, such as visits from government officials. Sometimes, however, important events occur, such as the fall of Saigon, that pull embassies away from their typical activities and lead them to write cables that discuss these events and their consequences. Capsule therefore operationalizes an "event" as a moment in history when multiple embassies deviate from their usual topics of discussion and each embassy deviates in a similar way.

Capsule embeds this intuition into a Bayesian model that uses latent variables to encode what "business as usual" means for each embassy, to characterize the events of each week, and to identify the cables that discuss those events. Given a corpus of cables, the corresponding posterior distribution of the latent variables provides a filter for the cables that isolates important moments in diplomatic history. Figure 1 depicts the mean of this posterior distribution.

We present the Capsule model in section 3, providing both a formal model specification and guidance on how to use the model to detect and characterize real-world events. In section 4, we validate Capsule using simulated data, and in section 5, we use it to analyze over two million U.S. State Department cables. Although we describe Capsule in the context of diplomatic cables, it is suitable for exploring any corpus with the same underlying structure: text (or other discrete multivariate data) generated over time by known entities. This includes email, consumer behavior, social media posts, and opinion articles.

2 Related Work

We first review previous work on automatic event detection and other related concepts, to contextualize our approach in general and Capsule in particular.

In both univariate and multivariate settings, ana-

lysts often want to predict whether or not rare events will occur (Weiss and Hirsh, 1998; Das et al., 2008). In contrast, Capsule is intended to help analysts explore and understand their data; our goal is human interpretability rather than prediction or forecasting.

Events can be construed as either anomalies—temporary deviations from usual behavior—or "changepoints" that mark persistent shifts in usual behavior (Guralnik and Srivastava, 1999; Adams and MacKay, 2007). We focus on events as anomalies.

Event detection in the context of news articles (Zhao et al., 2012; Zhao et al., 2007; Zhang et al., 2002; Li et al., 2005; Wang et al., 2007; Allan et al., 1998) and social media posts (Atefeh and Khreich, 2015; VanDam, 2012; Lau et al., 2012; Jackoway et al., 2011; Sakaki et al., 2010; Reuter and Cimiano, 2012; Becker et al., 2010; Sayyadi et al., 2009) usually means identifying clusters of documents. For news, the goal is to create new clusters as novel stories appear; each article is assumed to be associated with one event, which does not allow for distinctions between typical content and rare events. For social media, the goal is to identify rare events, but the resultant methods are intended for short documents, and are not appropriate for longer documents that may contain information about a variety of subjects.

Many existing methods for detecting events from text focus on individual vocabulary terms, often weighted by tf-idf values (Fung et al., 2005; Kumaran and Allan, 2004; Brants et al., 2003; Das Sarma et al., 2011; Zhao et al., 2007; Zhao et al., 2012). We characterize events by bursts in groups of terms.

Although groups of terms can be summarized directly (Peng et al., 2007; Chakrabarti and Punera, 2011; Gao et al., 2012), topic models (Blei, 2012) provide a way to automatically identify groups of related terms and reduce the dimensionality of text data. Researchers have previously used topic models to detect events mentioned in social media posts (Lau et al., 2012; Dou et al., 2012) and to find posts relevant to particular, monitored events (VanDam, 2012). Capsule uses topics to characterize both typical diplomatic content and potentially significant events.

In addition to modeling text over time, researchers have also used spatial information (Neill et al., 2005; Mathioudakis et al., 2010; Liu et al., 2011) and information about authors (Zhao et al., 2007) and news outlets (Wang et al., 2007) to enhance event detection. We rely on author information to characterize diplomatic "business as usual" for each embassy.

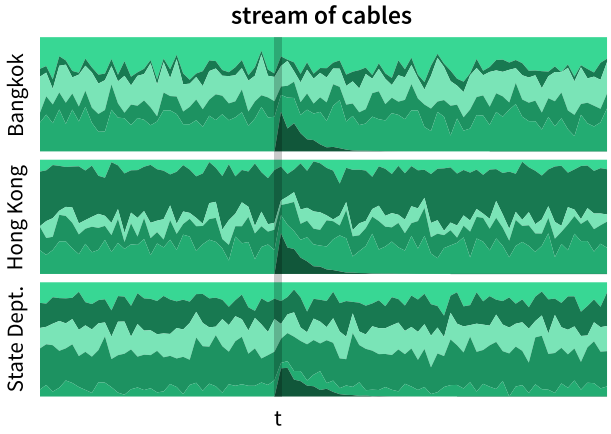


Figure 2: Cartoon intuition. The y-axis represents the stacked proportions of cables about various topics, while the x-axis represents time. The Bangkok embassy, Hong Kong embassy, and U.S. State Department all have typical diplomatic business, about which they usually send cables. When an event occurs during time interval t, the cables alter to cover the event before returning to "business as usual." Capsule discovers the entities' typical concerns, as well as the timing and content of events.

Event detection is closely related to detecting and characterizing relationships between entities (Schein et al., 2015; Linderman and Adams, 2014; Das Sarma et al., 2011). Capsule can trivially use sender–receiver pairs instead of authors, and the model specification can be tailored to reflect network structure.

Finally, there are connections between Capsule and recent work on Poisson processes. In particular, we can interpret Capsule as a collection of related discrete-time Poisson processes with random intensity measures. Further, marginalizing out the event strengths (described in section 3.1) reveals that the use of a vocabulary term by one embassy can "excite" the use of that term by another. This suggests a close relationship to Hawkes processes (Hawkes, 1971).

3 The Capsule Model

In this section, we present the Capsule model for detecting and characterizing significant diplomatic events. We first provide the intuition behind Capsule, and then formally specify the model. We also explain how to use Capsule to explore a corpus and how to learn the posterior distribution of the latent variables.

Consider an entity like the Bangkok embassy, as

illustrated in figure 2. We can imagine that this entity sends a stream of diplomatic cables over time—some to the U.S. State Department, others to other American embassies, such as the one in Hong Kong. Embassies usually write cables that describe typical diplomatic business. For example, the Bangkok embassy might write about topics regarding southeast Asia more generally. We can think of a topic as being a probability distribution over vocabulary terms.

Now imagine that an event, such as the capture of Saigon during the Vietnam War, occurs during a particular time interval t. We cannot directly observe the occurrence of this event, but we can observe the stream of cables and the event's impact on it. When the event occurs, multiple entities deviate from their usual topics of discussion simultaneously, before returning to their usual behavior, as depicted in figure 2. For example, the day after the capture of Saigon, the majority of the diplomatic cables written by the Bangkok embassy and several other entities were about Vietnam War refugees. If we think of the event as another probability distribution over vocabulary terms, then each entity's stream of cables reflects its typical concerns, as well as any significant events.

3.1 Model Specification

We now define the Capsule model. Our data come from *entities* (e.g., embassies) who send *messages* (e.g., diplomatic cables) over *time*; specifically, we observe the number of times n_{dv} that each vocabulary term v occurs in each message d. Each message is associated with an author entity a_d and a time interval t_d within which that message was sent.

We model each message with a bank of Poisson distributions[2]—one for each vocabulary term:

$$n_{dv} \sim \text{Poisson}\left(\lambda_{dv}\right). \tag{1}$$

The rate λ_{dv} blends the different influences on message content. Specifically, it blends three types of *topics*, intended to capture "business-as-usual" discussion and content related to significant events.

We operationalize each topic as a specialized probability distribution over vocabulary terms (the set of unique words in the corpus of messages), as is common in topic models (Blei et al., 2003; Canny, 2004; Gopalan et al., 2014)—i.e., each term is associated with each topic, but with a different probability.

Topic Type	Top Terms
General	visit, hotel, schedule, arrival
Entity	soviet, moscow, ussr, agreement
Event	saigon, evacuation, vietnam, help

Table 1: The highest-probability vocabulary terms for examples of the three types of topics (general, entity, and event). These examples come from the analysis that we describe section 5.

Each message blends 1) general topics $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ about diplomacy (e.g., terms about diplomats, terms about communication), 2) an entity topic $\boldsymbol{\eta}_{a_d}$ specific to the author of that message (e.g., terms about Hong Kong),[3] and 3) event topics $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_T$ that are specific to the events in recent time intervals (e.g., terms about a coup, terms about the death of a dignitary).

Examples of these three types of topics are in table 1. The general topic relates to planning travel, the entity topic captures words related to the U.S.S.R., and the event topic captures words related to the evacuation of Saigon toward the end of the Vietnam War.

The messages share the three types of topics in different ways: all messages share the general topics, messages written by a single entity share an entity topic, and messages in the same time interval use the event topics in similar ways. Each message blends its corresponding topics with a set of message-specific strengths. As a result, each message captures a different mix of general diplomacy discussion, entity-specific terms, and recent events. Specifically, the Poisson rate for vocabulary term v in message d is

$$\lambda_{dv} = \sum_{k=1}^{K} \theta_{dk}\beta_{kv} + \zeta_d \eta_{a_d v} + \sum_{t=1}^{T} f(t_d, t)\, \epsilon_{dt}\gamma_{tv}, \tag{2}$$

where θ_{dk} is message d's strength for general topic k, ζ_d is message d's strength for a_d's entity topic, and ϵ_{dt} is message d's strength for event topic t. The function $f(\cdot)$ ensures that the events influences decay over time. As we describe in appendix B, we

[2] Readers familiar with topic modeling may expect a multinomial model of term occurrences, but Poisson models of counts better capture messages with different lengths (Canny, 2004).

[3] The entity-specific topics play a similar role to the background topics introduced by Paul and Dredze (2012).

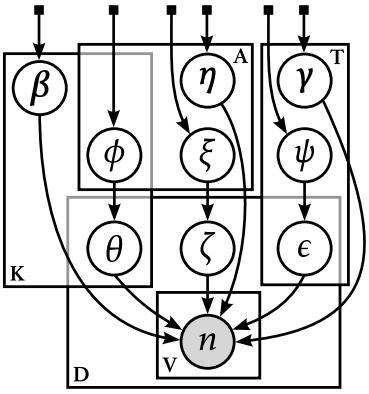


Figure 3: Graphical model for Capsule. Observed term counts depend on general topics $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$, entity topics $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_A$, and event topics $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_T$, as well as message-specific strengths $\boldsymbol{\theta}_d$, ζ_d, and $\boldsymbol{\epsilon}_d$. Variables $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_A$ and $\xi_1, \ldots, \xi_A$ represent entity-specific strengths, while $\psi_1, \ldots, \psi_T$ allow time intervals to be more or less "eventful." Black squares denote hyperparameters (unlabeled for visual simplicity).

compared several different decay functions (exponential, linear, and step) and found that the following exponential decay function works well in practice:

$$f(t_d, t) = \begin{cases} 0 & t \leq t_d < t + \tau \\ \exp\left(\frac{-(t_d - t)}{\tau / 5}\right) & \text{otherwise.} \end{cases} \quad (3)$$

Dividing τ by five means that we can interpret it as the number of time intervals after which an event will have little impact on the content of the messages.

We place hierarchical gamma priors over the message-specific strengths, introducing entity-specific strengths $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_A$ and $\xi_1, \ldots, \xi_A$ that allow different entities to focus on different topics and event strengths $\psi_1, \ldots, \psi_T$ that allow different time intervals to be more or less "eventful." We place Dirichlet priors over the topics. The graphical model is in figure 3 and the generative process is in figure 4.

Given a corpus of messages, learning the posterior distribution of the latent variables uncovers the three types of topics, the message- and entity-specific strengths, and the event strengths. In section 3.3, we explain how an analyst can use the event strengths as a filter that isolates potentially significant messages.

3.2 Learning the Posterior Distribution

In order to use Capsule to to explore a corpus of messages, we must first learn the posterior distribution of the latent variables—the general topics, the entity topics, the event topics, the message- and entity-specific strengths, and the event strengths—conditioned on the observed term counts. As for many Bayesian models, this posterior distribution is not tractable to compute; approximating it is therefore our central statistical and computational problem. We introduce an approximate inference algorithm for Capsule, based on variational methods (Jordan et al., 1999),[4], which

- for $k = 1, \ldots, K$,
  - draw general topic $\boldsymbol{\beta}_k \sim \text{Dirichlet}_V(\alpha, \ldots, \alpha)$
  - for each entity $a = 1, \ldots, A$,
    - ► draw entity-specific strength $\phi_{ak} \sim \text{Gamma}(s, r)$
- for each entity $a = 1, \ldots, A$,
  - draw entity topic $\boldsymbol{\eta}_a \sim \text{Dirichlet}_V(\alpha, \ldots, \alpha)$
  - draw entity-specific strength $\xi_a \sim \text{Gamma}(s, r)$
- for each time interval $t = 1, \ldots, T$,
  - draw event topic $\boldsymbol{\gamma}_t \sim \text{Dirichlet}_V(\alpha, \ldots, \alpha)$
  - draw event strength $\psi_t \sim \text{Gamma}(s, r)$
- for each message $d = 1, \ldots, D$, sent during time interval t_d by author entity a_d,
  - for each general topic $k = 1, \ldots, K$,
    - ► draw message-specific strength $\theta_{dk} \sim \text{Gamma}(s, \phi_{a_d k})$
  - draw message-specific strength $\zeta_d \sim \text{Gamma}(s, \xi_{a_d})$
  - for each time interval $t = 1, \ldots, T$,
    - ► draw message-specific strength $\epsilon_{dt} \sim \text{Gamma}(s, \psi_t)$
  - for each vocabulary term $v = 1, \ldots, V$,
    - ► set $\lambda_{dv} = \sum_{k=1}^{K} \theta_{dk} \beta_{kv} + \zeta_d \eta_{a_d v} + \sum_{t=1}^{T} f(t_d, t)\, \epsilon_{dt} \gamma_{tv}$
    - ► draw term counts $n_{d,v} \sim \text{Poisson}(\lambda_{dv})$

Figure 4: Generative process for Capsule. We use s and r to denote top-level (i.e., fixed) shape and rate hyperparameters; they can be set to different values for different variables.

[4] Source code: `https://github.com/ajbc/capsule`.

we outline in appendix A.[5] This algorithm produces a fitted variational distribution which be can then be used as a proxy for the true posterior distribution.

3.3 Detecting and Characterizing Events

We can use the mean of the fitted variational distribution to explore the data. Specifically, we can explore "business-as-usual" content using the posterior expected values of the general topics $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ and the entity topics $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_A$, and we can detect and characterize events using the posterior expected values of the event strengths and the event topics.

To detect events, we define an measure that quantifies the "eventness" of time interval t. Specifically, we first compute how relevant each message d is to that time interval: $m_{dt} = f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$. Using these relevancy values, we then compute the proportion of each message's term counts that are associated with the event topic specific to time interval t:

$$p_{dt} = \frac{m_{dt}}{\sum_k \mathbb{E}[\theta_{dk}] + \mathbb{E}[\zeta_d] + \sum_{t'} m_{dt'}}. \quad (4)$$

Finally, we aggregate these values over messages:

$$\frac{1}{\sum_d f(t_d, t)} \sum_{d=1}^{D} p_{dt}, \quad (5)$$

where the multiplicative fraction ensures that messages that were sent during time intervals that are further from t contribute less than than messages that were sent during time intervals that are closer to t.

We can characterize an event t by selecting the highest-probability vocabulary terms from $\mathbb{E}[\boldsymbol{\gamma}_t]$. By ordering the messages according to $m_{dt} = f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$, we can also identify the messages that are most strongly associated with event t.

In section 5, we explore the cables associated with significant events in the National Archives' corpus of diplomatic cables. To make Capsule more accessible for historians, political scientists, and journalists, we have released an open-source tool for visualizing its results.[6] This tool allows analysts to browse a corpus of messages and the mean of the corresponding posterior distribution, including general topics, entity topics, and event topics. Figure 5 contains several screenshots of the tool's browsing interface.

[5] Appendices are in the supplemental material.

[6] Source code: `https://github.com/ajbc/capsule-viz`; demo: `http://www.princeton.edu/~achaney/capsule/`.

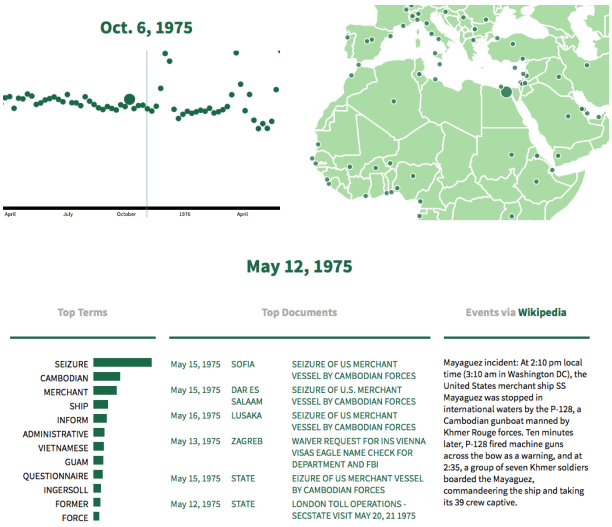


Figure 5: Screenshots of the Capsule visualization tool used to explore U.S. State Department cables. Top left: events over time (similar to figure 1). Top right: entities located on a map. Bottom: summary of the week of May 12, 1975, including top vocabulary terms, relevant cables, and text from Wikipedia.

4 Model Validation with Simulated Data

Before using Capsule to explore a corpus of real messages (described in section 5), we provide a quantitative validation of the model using simulated data.

We used the generative process in figure 4 to create ten data sets, each with 100 time intervals, ten general topics, ten entities, and roughly 20,000 messages. We then used these data sets to compare Capsule's event detection performance to that of four baseline methods. We also compared the methods' abilities to identify the most relevant messages for each event.

4.1 Detecting Events

For each data set, we ordered the time intervals from most to least eventful, using the "eventness" measure described in section 3.3 and the simulated values of the latent variables. We then treated these ranked lists of time intervals as "ground truth" and assessed how well each method was able to recover them.

For Capsule itself, we used our approximate inference algorithm to obtain a fitted variational distribution for each simulated data set. We then ordered the time intervals using our "eventness" measure and the posterior expected values of the latent variables.

For our first baseline, we constructed an "event-only" version of Capsule by dropping the first and

second terms in equation (2). We used this baseline to test whether modeling "business as usual" discussion makes it easier to detect significant events. We obtained a fitted variational distribution for this model using a variant of our approximate inference algorithm, and then ordered the time intervals using our "eventness" measure, modified appropriately, and the posterior expected values of the latent variables.

For our second baseline, we drew inspiration from previous work on event detection in the context of news articles, and focused on each time interval's deviation in term counts from the average. Specifically, we ordered the time intervals $1, \ldots, T$ for each simulated data set according to this measure:

$$\sum_{v=1}^{V} \sum_{\substack{d=1 \\ t_d = t}}^{D} \left| n_{dv} - \frac{1}{D} \sum_{d=1}^{D} n_{dv} \right|. \quad (6)$$

We added tf-idf term weights for our third baseline:

$$\sum_{v=1}^{V} \text{tf-idf}\,(v) \sum_{\substack{d=1 \\ t_d = t}}^{D} \left| n_{dv} - \frac{1}{D} \sum_{d=1}^{D} n_{dv} \right|. \quad (7)$$

Finally, we randomly ordered the time intervals for each data set to serve as a straw-man baseline.

We also experimented with baselines that involved term-count deviations on the entity level and topic-usage deviations on the message level (Dou et al., 2012), but found that they were not competitive.

For each data set, we compared each method's ranked list of time intervals to the corresponding "ground-truth" list of time intervals, by dividing the sum of the lists' actual set overlap at each rank by the sum of their maximum set overlap at each rank:

$$\frac{\sum_{r=1}^{T} |S_r^{\text{truth}} \cap S_r^{\text{method}}|}{\sum_{r=1}^{T} r}, \quad (8)$$

where S_r^{truth} is a set of the top r time intervals according to the "ground-truth" list and S_r^{method} is a set of the top r time intervals according to the method.

Figure 6 shows that Capsule outperforms all four baseline methods. These results serve as a sanity check for both the model and its implementation.

4.2 Identifying Relevant Messages

For each data set, we created a list of the most relevant messages for each time interval t by computing $f(t_d, t)\,\epsilon_{dt}$ for each message d (using the simulated values of ϵ_{dt}) and ordering the messages accordingly. We then treated these ranked lists of messages as "ground truth" and assessed how well Capsule and the baseline methods were able to recover them.

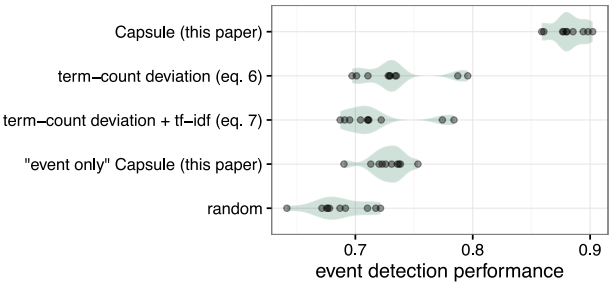


Figure 6: Event detection performance using ten simulated data sets. Each dot represents the performance (equation (8); higher is better) of a single method on a single data set; each shaded green area summarizes the distribution of performance for a single method. Capsule outperforms all four baseline methods.

For Capsule, we used our approximate inference algorithm to obtain a fitted variational distribution for each data set, and then, for each time interval, ordered the messages according to $m_{dt} = f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$. For our second and third baselines, we ordered the messages sent during each time interval according message-specific versions of equations (6) and (7).

For each data set, we compared each method's ranked list of messages for each time interval to the corresponding "ground-truth" list, by computing precision at ten messages. The average precision for Capsule was was 0.44, while the average precision for the "event-only" version of the model was 0.09. The other baselines recovered zero relevant messages.

5 Exploratory Analysis

Capsule is intended to help analysts explore and understand their data. In this section, we demonstrate its capabilities by analyzing a corpus of over two million U.S. State Department cables from the 1970s.

5.1 Data

The National Archive collects diplomatic cables sent between the U.S. State Department and its foreign embassies. We obtained a subset of this corpus from the Central Foreign Policy Files at the National Archives, via the History Lab at Columbia Univer-

sity;[7] the subset contains cables sent between 1973 and 1978. In addition to the text of the cables, each message is labeled with its author (e.g., the U.S. State Department, a particular embassy, or an individual), the date the cable was sent, and other metadata. We used a vocabulary of 6,293 terms and omitted cables with fewer than three terms, resulting in 2,021,852 cables sent by 22,961 entities. We used weekly time intervals, as few cables were sent on weekends.

5.2 Model Settings

We ran our approximate inference algorithm for Capsule to obtain a fitted variational distribution. We used $K = 100$ general topics, the exponential decay function in equation (3) with $\tau = 4$, and top-level hyperparameters $s = r = 0.3$. With these settings, a single iteration of the algorithm took about an hour.[8]

5.3 Detecting Well-Known Events

To evaluate Capsule's ability to detect well-known events, we used a list, provided to us by the History Lab, of thirty-nine well-known events that took place between 1973 and 1978. Each event is present in at least one of six reputable collections of historic events, such as the Office of the Historian's Milestones in the History of U.S. Foreign Relations.[9] We treated this list of events as "ground truth" and assessed how well Capsule and each of the baselines described in section 4.1 were able to recover them—or, in other words, how well the methods identify these eventful weeks, compared to more typical weeks.

Specifically, we used each method to construct a ranked list of time intervals. Then, for each method, we computed the discounted cumulative gain (DCG), which, in this context, is equivalent to computing

$$\sum_{e=1}^{39} \frac{1}{\log\left(\text{rank}\left(e, L_T^{\text{method}}\right)\right)}, \tag{9}$$

where L_T^{method} is the method's ranked list of time intervals and $\text{rank}\left(e, L_T^{\text{method}}\right)$ is the rank of the e^{th} well-known event in L_T^{method}. Finally, we divided the DCG by the ideal DCG—i.e., $\sum_{e=1}^{39} \frac{1}{\log(e)}$—to obtain the normalized DCG (nDCG). Table 2 shows that Capsule outperforms all four baseline methods.

Method	nDCG
Capsule (this paper)	0.693
term-count deviation + tf-idf (equation (7))	0.652
term-count deviation (equation (6))	0.642
random	0.557
"event-only" Capsule (this paper)	0.426

Table 2: Event detection performance (nDCG; higher is better) using thirty-nine well-known events that took place between 1973 and 1978. Capsule outperforms all four baseline methods.

5.4 Exploration

We now turn to our primary goal—using Capsule to explore and understand a corpus of messages. Figure 1 shows our "eventness" measure (equation (5)) over time. One of the tallest peaks occurs during the week of December 1, 1975, when the United Nations General Assembly discussed omnibus decolonization. As described in section 3.3, we can characterize this event by computing $m_{dt} = f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$ for each message d and then ordering the messages accordingly. Table 3 lists the highest-ranked messages.

Another notable event was the seizure of the S.S. Mayaguez, an American merchant vessel, during May, 1975, at the end of the Vietnam War. Table 4 lists the highest-ranked messages for this event. We can examine these messages to confirm their relevancy and learn more about the event. For example, here is the content of the most relevant message:

> In absence of MFA Chief of Eighth Department Avramov, I informed American desk officer Yankov of circumstances surrounding seizure and recovery of merchant ship Mayaguez and its crew. Yankov promised to inform the Foreign Minister of US statement today (May 15). Batjer

A third week of interest occurs in early July, 1976. On July 4, the U.S. celebrated its Bicentennial, but on the same day, Israeli forces completed a hostage rescue mission because an Air France flight from Tel Aviv had been hijacked and taken to Entebbe, Uganda.[10] This event was mostly discussed the week

[7] http://history-lab.org

[8] Each iteration of our algorithm considers all messages. Modifying it to stochastically sample the data would reduce the time required to obtain an equivalent fitted variational distribution.

[9] https://history.state.gov/milestones/1969-1976

[10] Capsule assumes that only one event occurs during each

$f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$	Date	Author Entity	Subject
4.60	1975-12-05	Canberra	30th UNGA: Item 23, Guam, Obmibus Decolonization and ...
4.26	1975-12-05	Mexico	30th UNGA-Item 23: Guam, Omnibus Decolonization and ...
4.21	1975-12-06	State	30th UNGA-Item 23: Guam, Omnibus Decolonization and ...
4.11	1975-12-03	Dakar	30th UNGA: Resolutions on American Samoa, Guam and ...
4.08	1975-12-04	Monrovia	30th UNGA: Item 23: Resolutions on decolonization and A...

Table 3: Highest-ranked messages for the week of December 1, 1975, when the United Nations General Assembly discussed decolonization. Capsule accurately recovers messages related to this real-world event. Typos are intentionally copied from the data.

$f(t_d, t)\,\mathbb{E}[\epsilon_{dt}]$	Date	Author Entity	Subject
5.06	1975-05-15	Sofia	Seizure of US merchant vessel by Cambodian forces
5.05	1975-05-15	Dar es Salaam	Seizure of U.S. merchant vessel by Cambodian forces
4.92	1975-05-16	Lusaka	Seizure of US merchant vessel by Cambodian forces
4.61	1975-05-13	Zagreb	Waiver request for INS Vienna visas Eagle name check...
4.59	1975-05-15	State	eizure of US merchant Vessel by Cambodian forces

Table 4: Highest-ranked messages for the week of May 12, 1975, when the S.S. Mayaguez, an American merchant vessel, was captured. Capsule accurately recovers messages related to this real-world event. Typos are intentionally copied from the data.

after the event took place; the most relevant messages are listed in appendix B (table 5). The cable from Stockholm describing the "Ugandan role in Air France hijacking" begins with the following content, which reveals further information about this event:

> 1. We provided MFA Director of Political Affairs Leifland with Evidence of Ugandan assistance to hijackers contained in Ref A. After reading material, Leifland described it a "quite good", and said it would be helpful for meeting MFA has scheduled for early this morning to determine position GOS will take at July 8 UNSC consideration of Israeli Rescue Operation. ...

In addition to detecting and characterizing well-known events, such the S.S. Mayaguez incident and Operation Entebbe, Capsule can detect and characterize obscure, but significant, events, such as when Eritrean rebels kidnapped Tenneco oil employees (April 8, 1974) and when the U.S. Navy evacuated citizens from Lebanon ("Operation Fluid Drive," June 20, 1976). Both events appear in figure 1. Capsule uncovers events where analysts might not otherwise look.

Capsule also provides a way to explore "business-as-usual" discussion using the posterior expected values of the general topics $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ and the entity topics $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_A$. Examples of each of these types of topics are in appendix B (tables 6 and 7, respectively); these examples illustrate that, as desired, the entity topics absorb location-specific terms, preventing them from overwhelming the general topics.

6 Conclusion

We presented Capsule, a Bayesian model for detecting and characterizing potentially significant events. We evaluated Capsule's ability to detect events and identify relevant messages; it outperformed four baseline methods. We used Capsule to analyze a large corpus of U.S. State Department cables from the 1970s, demonstrating that it can discover both well-known and obscure (but significant) events, as well as relevant documents. We anticipate that Capsule, and our visualization tool, will be useful for historians, political scientists, and journalists who wish to explore and understand large corpora of documents. This is increasingly important—the U.S. State Department alone produces around two billion e-mails annually.

Acknowledgments

This work was supported by NSF IIS-1247664; ONR N00014-11-1-0651; DARPA FA8750-14-2-0009 and N66001-15-C-4032; Adobe; the Alfred P. Sloan Foundation; the Columbia Global Policy Initiative.

time interval. This example is a clear violation of this assumption, but also serves to demonstrate that Capsule can successfully detect and characterize multiple events, even when they overlap.

References

Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45.

Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164.

Hila Becker, Mor Naaman, and Luis Gravano. 2010. Learning similarity metrics for event identification in social media. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 291–300.

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, January.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Thorsten Brants, Francine Chen, and Ayman Farahat. 2003. A system for new event detection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 330–337.

John Canny. 2004. Gap: a factor model for discrete data. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 122–129.

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*, 11:66–73.

Kaustav Das, Jeff Schneider, and Daniel B Neill. 2008. Anomaly pattern detection in categorical datasets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 169–176.

Anish Das Sarma, Alpa Jain, and Cong Yu. 2011. Dynamic relationship and event discovery. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 207–216.

Wenwen Dou, Xiaoyu Wang, Drew Skau, William Ribarsky, and Michelle X Zhou. 2012. Leadline: Interactive visual analysis of text data through event identification and exploration. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 93–102. IEEE.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S Yu, and Hongjun Lu. 2005. Parameter free bursty events detection in text streams. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 181–192. VLDB Endowment.

Wei Gao, Peng Li, and Kareem Darwish. 2012. Joint topic modeling for event summarization across news and social media streams. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1173–1182.

Prem K Gopalan, Laurent Charlin, and David Blei. 2014. Content-based recommendations with Poisson factorization. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 3176–3184. Curran Associates, Inc.

Valery Guralnik and Jaideep Srivastava. 1999. Event detection from time series data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 33–42.

Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

Alan Jackoway, Hanan Samet, and Jagan Sankaranarayanan. 2011. Identification of live news events using twitter. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 25–32. ACM.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November.

Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 297–304.

Jey Han Lau, Nigel Collier, and Timothy Baldwin. 2012. On-line trend analysis with topic models:\# twitter trends detection topic model online. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1519–1534.

Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. 2005. A probabilistic model for retrospective news event detection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 106–113.

Scott W Linderman and Ryan P Adams. 2014. Discovering latent network structure in point process data. *arXiv preprint arXiv:1402.0914*.

Xueliang Liu, Raphaël Troncy, and Benoit Huet. 2011. Using social media to identify events. In *Proceedings of the ACM SIGMM International Workshop on Social Media (WSM)*, pages 3–8.

Michael Mathioudakis, Nilesh Bansal, and Nick Koudas. 2010. Identifying, attributing and describing spatial bursts. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 3(1-2):1091–1102.

Daniel B Neill, Andrew W Moore, Maheshkumar Sabhnani, and Kenny Daniel. 2005. Detection of emerging space-time clusters. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 218–227.

Michael J Paul and Mark Dredze. 2012. A model for mining public health topics from twitter. *Health*, 11:16–6.

Wei Peng, Charles Perng, Tao Li, and Haixun Wang. 2007. Event summarization for system management. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1028–1032.

Timo Reuter and Philipp Cimiano. 2012. Event-based classification of social media streams. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 22. ACM.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 851–860.

Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event detection and tracking in social streams. In *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*.

Aaron Schein, John Paisley, David M Blei, and Hanna Wallach. 2015. Bayesian Poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1045–1054.

Courtland VanDam. 2012. A probabilistic topic modeling approach for event detection in social media. Master's thesis, Michigan State University.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 784–793. ACM.

Gary M Weiss and Haym Hirsh. 1998. Learning to predict rare events in event sequences. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 359–363.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 81–88.

Qiankun Zhao, Prasenjit Mitra, and Bi Chen. 2007. Temporal and information flow based event detection from social text streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 7, pages 1501–1506.

Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. 2012. A novel burst-based text representation model for scalable event detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 43–47. Association for Computational Linguistics.

Convolutional Neural Network Language Models

Ngoc-Quan Pham and **German Kruszewski** and **Gemma Boleda**
Center for Mind/Brain Sciences
University of Trento
{firstname.lastname}@unitn.it

Abstract

Convolutional Neural Networks (CNNs) have shown to yield very strong results in several Computer Vision tasks. Their application to language has received much less attention, and it has mainly focused on static classification tasks, such as sentence classification for Sentiment Analysis or relation extraction. In this work, we study the application of CNNs to language modeling, a dynamic, sequential prediction task that needs models to capture local as well as long-range dependency information. Our contribution is twofold. First, we show that CNNs achieve 11-26% better absolute performance than feed-forward neural language models, demonstrating their potential for language representation even in sequential tasks. As for recurrent models, our model outperforms RNNs but is below state of the art LSTM models. Second, we gain some understanding of the behavior of the model, showing that CNNs in language act as feature detectors at a high level of abstraction, like in Computer Vision, and that the model can profitably use information from as far as 16 words before the target.

1 Introduction

Convolutional Neural Networks (CNNs) are the family of neural network models that feature a type of layer known as the convolutional layer. This layer can extract features by convolving a learnable filter (or kernel) along different positions of a vectorial input.

CNNs have been successfully applied in Computer Vision in many different tasks, including object recognition, scene parsing, and action recognition (Gu et al., 2015), but they have received less attention in NLP. They have been somewhat explored in *static classification tasks* where the model is provided with a full linguistic unit as input (e.g. a sentence) and classes are treated as independent of each other. Examples of this are sentence or document classification for tasks such as Sentiment Analysis or Topic Categorization (Kalchbrenner et al., 2014; Kim, 2014), sentence matching (Hu et al., 2014), and relation extraction (Nguyen and Grishman, 2015). However, their application to *sequential prediction tasks*, where the input is construed to be part of a sequence (for example, language modeling or POS tagging), has been rather limited (with exceptions, such as Collobert et al. (2011)). The main contribution of this paper is a systematic evaluation of CNNs in the context of a prominent sequential prediction task, namely, language modeling.

Statistical language models are a crucial component in many NLP applications, such as Automatic Speech Recognition, Machine Translation, and Information Retrieval. Here, we study the problem under the standard formulation of learning to predict the upcoming token given its previous context. One successful approach to this problem relies on counting the number of occurrences of n-grams while using smoothing and back-off techniques to estimate the probability of an upcoming word (Kneser and Ney, 1995). However, since each individual word is treated independently of the others, n-gram models fail to capture semantic relations between words. In contrast, neural network language models (Bengio et al., 2006) learn to predict the up-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1153–1162,
Austin, Texas, November 1-5, 2016.

coming word given the previous context while embedding the vocabulary in a continuous space that can represent the similarity structure between words. Both feed-forward (Schwenk, 2007) and recurrent neural networks (Mikolov et al., 2010) have been shown to outperform n-gram models in various setups (Mikolov et al., 2010; Hai Son et al., 2011). These two types of neural networks make different architectural decisions. Recurrent networks take one token at a time together with a hidden "memory" vector as input and produce a prediction and an updated hidden vector for the next time step. In contrast, feed-forward language models take as input the last n tokens, where n is a fixed window size, and use them jointly to predict the upcoming word.

In this paper we define and explore CNN-based language models and compare them with both feed-forward and recurrent neural networks. Our results show a 11-26% perplexity reduction of the CNN with respect to the feed-forward language model, comparable or higher performance compared to similarly-sized recurrent models, and lower performance with respect to larger, state-of-the-art recurrent language models (LSTMs as trained in Zaremba et al. (2014)).

Our second contribution is an analysis of the kind of information learned by the CNN, showing that the network learns to extract a combination of grammatical, semantic, and topical information from tokens of all across the input window, even those that are the farthest from the target.

2 Related Work

Convolutional Neural Networks (CNNs) were originally designed to deal with hierarchical representation in Computer Vision (LeCun and Bengio, 1995). Deep convolutional networks have been successfully applied in image classification and understanding (Simonyan and Zisserman, 2014; He et al., 2015). In such systems the convolutional kernels learn to detect visual features at both local and more abstract levels.

In NLP, CNNs have been mainly applied to static classification tasks for discovering latent structures in text. Kim (2014) uses a CNN to tackle sentence classification, with competitive results. The same work also introduces kernels with varying window sizes to learn complementary features at different aggregation levels. Kalchbrenner et al. (2014) propose a convolutional architecture for sentence representation that vertically stacks multiple convolution layers, each of which can learn independent convolution kernels. CNNs with similar structures have also been applied to other classification tasks, such as semantic matching (Hu et al., 2014), relation extraction (Nguyen and Grishman, 2015), and information retrieval (Shen et al., 2014). In contrast, Collobert et al. (2011) explore a CNN architecture to solve various sequential and non-sequential NLP tasks such as part-of-speech tagging, named entity recognition and also language modeling. This is perhaps the work that is closest to ours in the existing literature. However, their model differs from ours in that it uses a max-pooling layer that picks the most activated feature across time, thus ignoring temporal information, whereas we explicitly avoid doing so. More importantly, the language models trained in that work are only evaluated through downstream tasks and through the quality of the learned word embeddings, but not on the sequence prediction task itself, as we do here.

Besides being applied to word-based sequences, the convolutional layers have also been used to model sequences at the character level. Kim et al. (2015) propose a recurrent language model that replaces the word-indexed projection matrix with a convolution layer fed with the character sequence that constitutes each word to find morphological patterns. The main difference between that work and ours is that we consider words as the smallest linguistic unit, and thus apply the convolutional layer at the word level.

Statistical language modeling, the task we tackle, differs from most of the tasks where CNNs have been applied before in multiple ways. First, the input typically consists of incomplete sequences of words rather than complete sentences. Second, as a classification problem, it features an extremely large number of classes (the words in a large vocabulary). Finally, temporal information, which can be safely discarded in many settings with little impact in performance, is critical here: An n-gram appearing close to the predicted word may be more informative, or yield different information, than the same n-gram appearing several tokens earlier.

3 Models

Our model is constructed by extending a feed-forward language model (FFLM) with convolutional layers. In what follows, we first explain the implementation of the base FFLM and then describe the CNN model that we study.

3.1 Baseline FFLM

Our baseline feed-forward language model (FFLM) is almost identical to the original model proposed by Bengio et al. (2006), with only slight changes to push its performance as high as we can, producing a very strong baseline. In particular, we extend it with highway layers and use Dropout as regularization. The model is illustrated in Figure 1 and works as follows. First, each word in the input n-gram is mapped to a low-dimensional vector (viz. embedding) though a shared lookup table. Next, these word vectors are concatenated and fed to a highway layer (Srivastava et al., 2015). Highway layers improve the gradient flow of the network by computing as output a convex combination between its input (called the *carry*) and a traditional non-linear transformation of it (called the *transform*). As a result, if there is a neuron whose gradient cannot flow through the transform component (e.g., because the activation is zero), it can still receive the back-propagation update signal through the carry gate. We empirically observed the usage of a single highway layer to significantly improve the performance of the model. Even though a systematic evaluation of this aspect is beyond the scope of the current paper, our empirical results demonstrate that the resulting model is a very competitive one (see Section 4).

Finally, a softmax layer computes the model prediction for the upcoming word. We use ReLU for all non-linear activations, and Dropout (Hinton et al., 2012) is applied between each hidden layer.

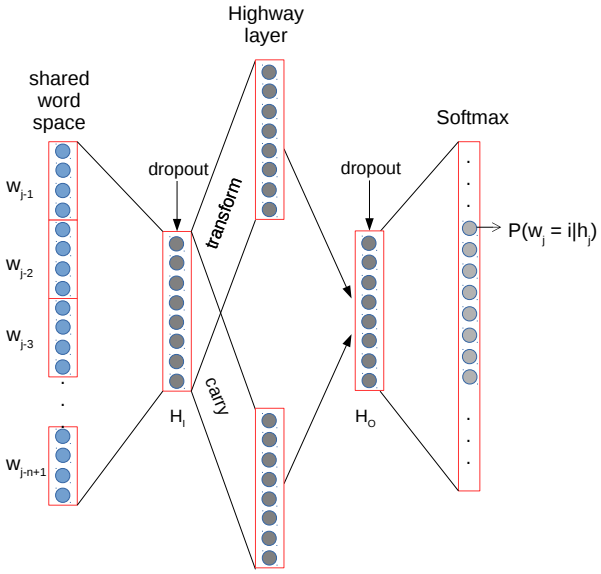


Figure 1: Overview of baseline FFLM.

3.2 CNN and variants

The proposed CNN network is produced by injecting a convolutional layer right after the words in the input are projected to their embeddings (Figure 2). Rather than being concatenated into a long vector, the embeddings $x_i \in \mathbb{R}^k$ are concatenated transversally producing a matrix $x_{1:n} \in \mathbb{R}^{n \times k}$, where n is the size of the input and k is the embedding size. This matrix is fed to a time-delayed layer, which convolves a sliding window of w input vectors centered on each word vector using a parameter matrix $W \in \mathbb{R}^{w \times k}$. Convolution is performed by taking the dot-product between the kernel matrix W and each sub-matrix $x_{i-w/2:i+w/2}$ resulting in a scalar value for each position i in input context. This value represents how much the words encompassed by the window match the feature represented by the filter W. A ReLU activation function is applied subsequently so negative activations are discarded. This operation is repeated multiple times using various kernel matrices W, learning different features independently. We tie the number of learned kernels to be the same as the embedding dimensionality k, such that the output of this stage will be another matrix of dimensions $n \times k$ containing the activations for each kernel at each time step. The number of kernels was tied to the embedding size for two reasons, one practical, namely, to limit the hyper parameter search, one methodological, namely, to keep the network structure identical to that of the baseline feed-forward model.

Next, we add a batch normalization stage immediately after the convolutional output, which facilitates learning by addressing the internal covariate

shift problem and regularizing the learned representations (Ioffe and Szegedy, 2015).

Finally, this feature matrix is directly fed into a fully connected layer that can project the extracted features into a lower-dimensional representation. This is different from previous work, where a max-over-time pooling operation was used to find the most activated feature in the time series. Our choice is motivated by the fact that the max pooling operator loses the specific position where the feature was detected, which is important for word prediction.

After this initial convolutional layer, the network proceeds identically to the FFNN by feeding the produced features into a highway layer, and then, to a softmax output.

This is our basic CNN architecture. We also experiment with three expansions to the basic model, as follows. First, we generalize the CNN by extending the shallow linear kernels with deeper multilayer perceptrons, in what is called a MLP Convolution (**MLPConv**) structure (Lin et al., 2013). This allows the network to produce non-linear filters, and it has achieved state-of-the-art performance in object recognition while reducing the number of total layers compared to other mainstream networks. Concretely, we implement MLPConv networks by using another convolutional layer with a 1×1 kernel on top of the convolutional layer output. This results in an architecture that is exactly equivalent to sliding a one-hidden-layer MLP over the input. Notably, we do not include the global pooling layer in the original Network-in-Network structure (Lin et al., 2013).

Second, we explore stacking convolutional layers on top of each other (Multi-layer CNN or **ML-CNN**) to connect the local features into broader regional representations, as commonly done in computer vision. While this proved to be useful for sentence representation (Kalchbrenner et al., 2014), here we have found it to be rather harmful for language modeling, as shown in Section 4. It is important to note that, in ML-CNN experiments, we stack convolutions with the same kernel size and number of kernels on top of each other, which is to be distinguished from the MLPConv that refers to the deeper structure in each CNN layer mentioned above.

Finally, we consider combining features learned through different kernel sizes (**COM**), as depicted in Figure 3. For example, we can have a combination of kernels that learn filters over 3-grams with others that learn over 5-grams. This is achieved simply by applying in parallel two or more sets of kernels to the input and concatenating their respective outputs (Kim, 2014).

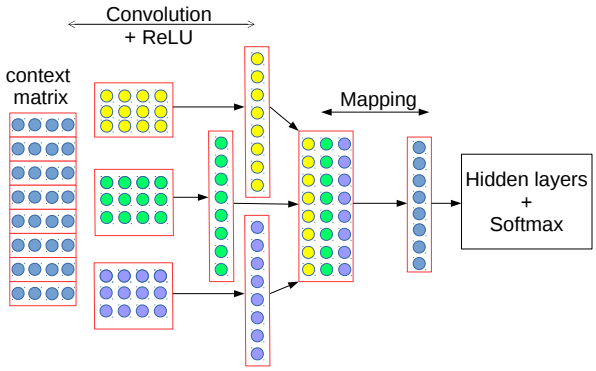


Figure 2: Convolutional layer on top of the context matrix.

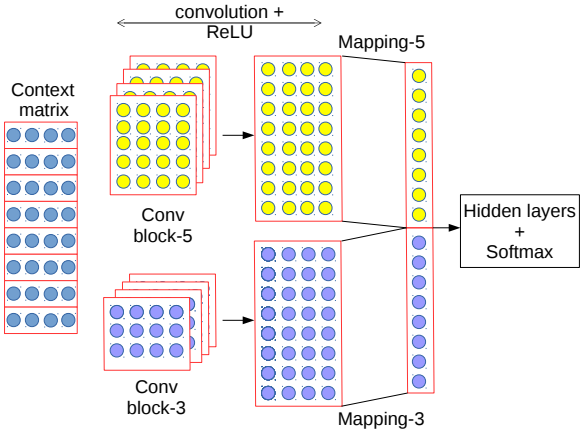


Figure 3: Combining kernels with different sizes. We concatenate the outputs of 2 convolutional blocks with kernel size of 5 and 3 respectively.

4 Experiments

We evaluate our model on three English corpora of different sizes and genres, the first two of which have been used for language modeling evaluation before. The **Penn Treebank** contains one million words of newspaper text with 10K words in the vocabulary. We reuse the preprocessing and training/test/validation division from Mikolov et

al. (2014). **Europarl-NC** is a 64-million word corpus that was developed for a Machine Translation shared task (Bojar et al., 2015), combining Europarl data (from parliamentary debates in the European Union) and News Commentary data. We preprocessed the corpus with tokenization and true-casing tools from the Moses toolkit (Koehn et al., 2007). The vocabulary is composed of words that occur at least 3 times in the training set and contains approximately 60K words. We use the validation and test set of the MT shared task. Finally, we took a subset of the **ukWaC** corpus, which was constructed by crawling UK websites (Baroni et al., 2009). The training subset contains 200 million words and the vocabulary consists of the 200K words that appear more than 5 times in the training subset. The validation and test sets are different subsets of the ukWaC corpus, both containing 120K words. We preprocessed the data similarly to what we did for Europarl-NC.

We train our models using Stochastic Gradient Descent (SGD), which is relatively simple to tune compared to other optimization methods that involve additional hyper parameters (such as alpha in RMSprop) while being still fast and effective. SGD is commonly used in similar work (Devlin et al., 2014; Zaremba et al., 2014; Sukhbaatar et al., 2015). The learning rate is kept fixed during a single epoch, but we reduce it by a fixed proportion every time the validation perplexity increases by the end of the epoch. The values for learning rate, learning rate shrinking and mini-batch sizes as well as context size are fixed once and for all based on insights drawn from previous work (Hai Son et al., 2011; Sukhbaatar et al., 2015; Devlin et al., 2014) as well as experimentation with the Penn Treebank validation set.

Specifically, the learning rate is set to 0.05, with mini-batch size of 128 (we do not take the average of loss over the batch, and the training set is shuffled). We multiply the learning rate by 0.5 every time we shrink it and clip the gradients if their norm is larger than 12. The network parameters are initialized randomly on a range from -0.01 to 0.01 and the context size is set to 16. In Section 6 we show that this large context window is fully exploited.

For the base FFNN and CNN we varied embedding sizes (and thus, number of kernels) $k = 128, 256$. For $k = 128$ we explore the simple CNN, incrementally adding MLPConv and COM variations (in that order) and, alternatively, using a ML-CNN. For $k = 256$, we only explore the former three alternatives (i.e. all but the ML-CNN). For the kernel size, we set it to $w = 3$ words for the simple CNN (out of options $3, 5, 7, 9$), whereas for the COM variant we use $w = 3$ and 5, based on experimentation on PTB. However, we observed the models to be generally robust to this parameter. Dropout rates are tuned specifically for each combination of model and dataset based on the validation perplexity. We also add small dropout ($p = 0.05$–0.15) when we train the networks on the smaller corpus (Penn Treebank).

The experimental results for recurrent neural network language models, such as Recurrent Neural Networks (RNN) and Long-Short Term Memory models (LSTM), on the Penn Treebank are quoted from previous work; for Europarl-NC, we train our own models (we also report the performance of these in-house trained RNN and LSTM models on the Penn Treebank for reference). Specifically, we train LSTMs with embedding size $k = 256$ and number of layers $L = 2$ as well as $k = 512$ with $L = 1, 2$. We train one RNN with $k = 512$ and $L = 2$. To train these models, we use the published source code from Zaremba et al. (2014). Our own models are also implemented in Torch7 for easier comparison.[1] Finally, we selected the best performing convolutional and recurrent language models on Europarl-NC and the Baseline FFLM to be evaluated on the ukWaC corpus.

For all models trained on Europarl-NC and ukWaC, we speed up training by approximating the softmax with Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), with the parameters being set following previous work (Chen et al., 2015). Concretely, for each predicted word, we sample 10 words from the unigram distribution, and the normalization factor is such that $\ln Z = 9$. [2]

For comparison, we also implemented a simpler version of the FFNN without dropout and highway layers (Bengio et al., 2006). These networks have two hidden layers (Arisoy et al., 2012) with the size

[1] Available at https://github.com/quanpn90/NCE_CNNLM.

[2] We also experimented with Hierarchical Softmax (Mikolov et al., 2011) and found out that the NCE gave better performance in terms of speed and perplexity.

of 2 times the embedding size (k), thus having the same number of parameters as our baseline.

5 Results

Our experimental results are summarized in Table 1.

First of all, we can see that, even though the FFNN gives a very competitive performance,[3] the addition of convolutional layers clearly improves it even further. Concretely, we observe a solid 11-26% reduction of perplexity compared to the feed-forward network after using MLP Convolution, depending on the setup and corpus. CNN alone yields a sizable improvement (5-24%), while MLP-Conv, in line with our expectations, adds another approximately 2-5% reduction in perplexity. A final (smaller) improvement comes from combining kernels of size 3 and 5, which can be attributed to a more expressive model that can learn patterns of n-grams of different sizes. In contrast to the successful two variants above, the multi-layer CNN did not help in better capturing the regularities of text, but rather the opposite: the more convolutional layers were stacked, the worse the performance. This also stands in contrast to the tradition of convolutional networks in Computer Vision, where using very deep convolutional neural networks is key to having better models. Deep convolution for text representation is in contrast rather rare, and to our knowledge it has only been successfuly applied to sentence representation (Kalchbrenner et al., 2014). We conjecture that the reason why deep CNNs may not be so effective for language could be the effect of the convolution on the data: The convolution output for an image is akin to a new, more abstract image, which yet again can be subject to new convolution operations, whereas the textual counterpart may no longer have the same properties, in the relevant aspects, as the original linguistic input.

Regarding the comparison with a stronger LSTM, our models can perform competitively under the same embedding dimension (e.g. see $k = 256$ of $k = 512$) on the first two datasets. However, the LSTM can be easily scaled using larger models, as shown in Zaremba et al. (2014), which gives the best known results to date. This is not an option for our model, which heavily overfits with large hidden layers (around 1000) even with very large dropout values. Furthermore, the experiments on the larger ukWaC corpus show an even clearer advantage for the LSTM, which seems to be more efficient at harnessing this volume of data, than in the case of the two smaller corpora.

To sum up, we have established that the results of our CNN model are well above those of simple feed forward networks and recurrent neural networks. While they are below state of the art LSTMs, they are able to perform competitively with them for small and moderate-size models. Scaling to larger sizes may be today the main roadblock for CNNs to reach the same performances as large LSTMs in language modeling.

6 Model Analysis

In what follows, we obtain insights into the inner workings of the CNN by looking into the linguistic patterns that the kernels learn to extract and also studying the temporal information extracted by the network in relation to its prediction capacity.

Learned patterns To get some insight into the kind of patterns that each kernel is learning to detect, we fed trigrams from the validation set of the Penn Treebank to each of the kernels, and extracted the ones that most highly activated the kernel, similarly to what was done in Kalchbrenner et al. (2014). Some examples are shown in Figure 4. Since the word windows are made of embeddings, we can expect patterns with similar embeddings to have close activation outputs. This is borne out in the analysis: The kernels specialize in distinct features of the data, including more syntactic-semantic constructions (cf. the "comparative kernel" including *as ... as* patterns, but also *of more than*) and more lexical or topical features (cf. the "ending-in-month-name" kernel). Even in the more lexicalized features, however, we see linguistic regularities at different levels being condensed in a single kernel: For instance, the "spokesman" kernel detects phrases consisting of an indefinite determiner, a company name (or the word *company* itself) and the word "spokesman". We hypothesize that the convolutional layer adds an "I identify one specific feature, but at a high level of

[3] In our experiments, increasing the number of fully connected layers of the FFNN is harmful. Two hidden layers with highway connections is the best setting we could find.

Model	k	w	Penn Treebank			Europarl-NC			ukWaC		
			val	test	#p	val	test	#p	val	test	#p
FFNN (Bengio et al., 2006)	128	-	156	147	4.5	-	-	-	-	-	-
Baseline FFNN	128	-	114	109	4.5	-	-	-	-	-	-
+CNN	128	3	108	102	4.5	-	-	-	-	-	-
+MLPConv	128	3	102	97	4.5	-	-	-	-	-	-
+MLPConv+COM	128	3+5	96	**92**	8	-	-	-	-	-	-
+ML-CNN (2 layers)	128	3	113	108	8	-	-	-	-	-	-
+ML-CNN (4 layers)	128	3	130	124	8	-	-	-	-	-	-
FFNN (Bengio et al., 2006)	256	-	161	152	8.2	-	-	-	-	-	-
Baseline FFNN	256	-	110	105	8.2	133	174	48	136	147	156
+CNN	256	3	104	98	8.3	112	133	48	-	-	-
+MLPConv	256	3	97	93	8.3	107	128	48	108	**116**	156
+MLPConv+COM	256	3+5	95	**91**	18	108	**128**	83	-	-	-
+MLPConv+COM	512	3+5	96	92	52	-	-	-	-	-	-
Model	k	L	Penn Treebank			Europarl-NC			ukWaC		
			val	test	#p	val	test	#p	val	test	#p
RNN (Mikolov et al., 2014)	300	1	133	129	6	-	-	-	-	-	-
LSTM (Mikolov et al., 2014)	300	1	120	115	6.3	-	-	-	-	-	-
LSTM (Zaremba et al., 2014)	1500	2	82	**78**	48	-	-	-	-	-	-
LSTM (trained in-house)	256	2	108	103	5.1	137	155	31	-	-	-
LSTM (trained in-house)	512	1	123	118	12	133	149	62	-	-	-
LSTM (trained in-house)	512	2	94	90	11	114	**124**	63	79	**83**	205
RNN (trained in-house)	512	2	129	121	10	152	173	61	-	-	-

Table 1: Results on Penn Treebank and Europarl-NC. Figure of merit is perplexity (lower is better). Legend: k: embedding size (also number of kernels for the convolutional models and hidden layer size for the recurrent models); w: kernel size; *val*: results on validation data; *test*: results on test data; *#p*: number of parameters in millions; L: number of layers.

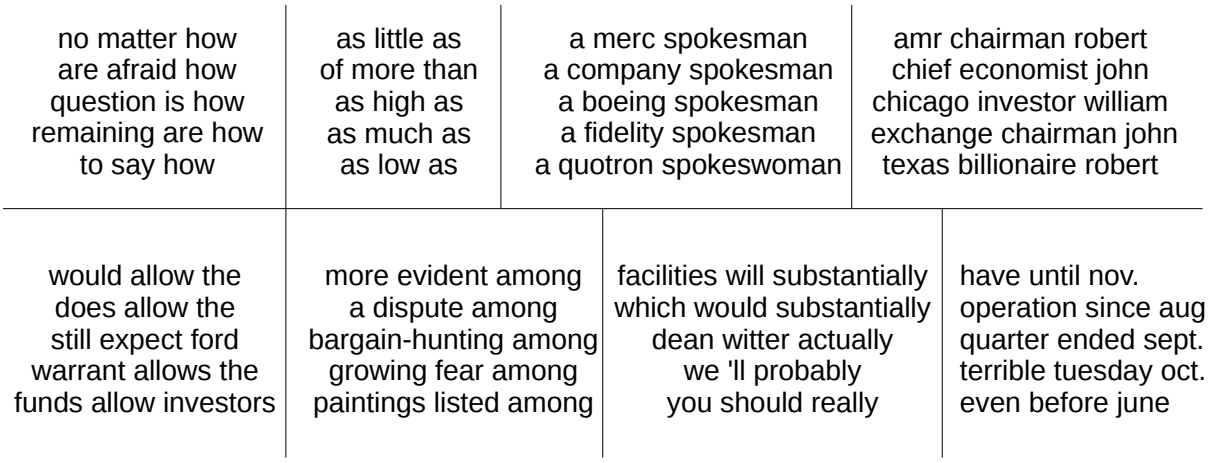


Figure 4: Some example phrases that have highest activations for 8 example kernels (each box), extracted from the validation set of the Penn Treebank. Model trained with 256 kernels for 256-dimensional word vectors.

abstraction" dimension to a feed-forward neural network, similarly to what has been observed in image classification (Krizhevsky et al., 2012).

Temporal information To the best of our knowledge, the longest context used in feed-forward language models is 10 tokens (Hai Son et al., 2012),

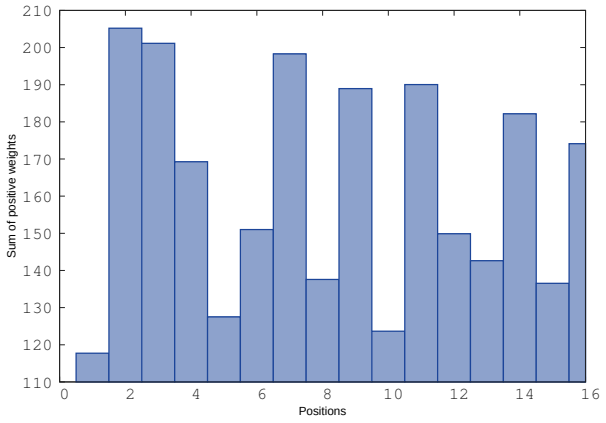


Figure 5: The distribution of positive weights over context positions, where 1 is the position closest to the predicted word.

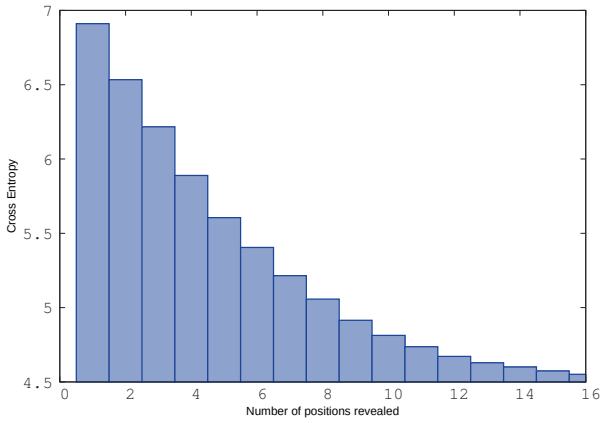


Figure 6: Perplexity change over position, by incrementally revealing the Mapping's weights corresponding to each position.

where no significant change in terms of perplexity was observed for bigger context sizes, even though in that work only same-sentence contexts were considered. In our experiments, we use a larger context size of 16 while removing the sentence boundary limit (as commonly done in n-gram language models) such that the network can take into account the words in the previous sentences.

To analyze whether all this information was effectively used, we took our best model, the CNN+MLPConv+COM model with embedding size of 256 (fifth line of second block in Table 1), and we identified the weights in the model that map the convolutional output (of size $n \times k$) to a lower dimensional vector (the "mapping" layer in Figure 2). Recall that the output of the convolutional layer is a matrix indexed by time step and kernel index containing the activation of the kernel when convolved with a window of text centered around the word at the given time step. Thus, output units of the above mentioned mapping predicate over an ensemble of kernel activations for each time step. We can identify the patterns that they learn to detect by extracting the time-kernel combinations for which they have positive weights (since we have ReLU activations, negative weights are equivalent to ignoring a feature). First, we asked ourselves whether these units tend to be more focused on the time steps closer to the target or not. To test this, we calculated the sum of the positive weights for each position in time using an average of the mappings that correspond to each output unit. The results are shown in Figure 5. As could be expected, positions that are close to the token to be predicted have many active units (local context is very informative; see positions 2-4). However, surprisingly, positions that are actually far from the target are also quite active. It seems like the CNN is putting quite a lot of effort on characterizing long-range dependencies.

Next, we checked that the information extracted from the positions that are far in the past are actually used for prediction. To measure this, we artificially lesioned the network so it would only read the features from a given range of time steps (words in the context). To lesion the network we manually masked the weights of the mapping that focus on times outside of the target range by setting them to zero. We started using only the word closest to the final position and sequentially unmasked earlier positions until the full context was used again. The result of this experiment is presented in Figure 6, and it confirms our previous observation that positions that are the farthest away contribute to the predictions of the model. The perplexity drops dramatically as the first positions are unmasked, and then decreases more slowly, approximately in the form of a power law ($f(x) \propto x^{-0.9}$). Even though the effect is smaller, the last few positions still contribute to the final perplexity.

7 Conclusion

In this work, we have investigated the potential of Convolutional Neural Networks for one prominent NLP task, language modeling, a sequential predic-

tion task. We incorporate a CNN layer on top of a strong feed-forward model enhanced with modern techniques like Highway Layers and Dropout. Our results show a solid 11-26% reduction in perplexity with respect to the feed-forward model across three corpora of different sizes and genres when the model uses MLP Convolution and combines kernels of different window sizes. However, even without these additions we show CNNs to effectively learn language patterns that allow it to significantly decrease the model perplexity.

In our view, this improvement responds to two key properties of CNNs, highlighted in the analysis. First, as we have shown, they are able to integrate information from larger context windows, using information from words that are as far as 16 positions away from the predicted word. Second, as we have qualitatively shown, the kernels learn to detect specific patterns at a high level of abstraction. This is analogous to the role of convolutions in Computer Vision. The analogy, however, has limits; for instance, a deeper model stacking convolution layers harms performance in language modeling, while it greatly helps in Computer Vision. We conjecture that this is due to the differences in the nature of visual vs. linguistic data. The convolution creates sort of abstract images that still retain significant properties of images. When applied to language, it detects important textual features but distorts the input, such that it is not text anymore.

As for recurrent models, even if our model outperforms RNNs, it is well below state-of-the-art LSTMs. Since CNNs are quite different in nature, we believe that a fruitful line of future research could focus on integrating the convolutional layer into a recurrent structure for language modeling, as well as other sequential problems, perhaps capturing the best of both worlds.

Acknowledgments

We thank Marco Baroni and three anonymous reviewers for fruitful feedback. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 655577 (LOVe); ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES) and the Erasmus Mundus Scholarship for Joint Master Programs. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used in our research.

References

Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Xie Chen, Xunying Liu, Mark JF Gales, and Philip C Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5411–5415. IEEE.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. 2015. Recent ad-

vances in convolutional neural networks. *CoRR*, abs/1512.07108.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, page 6.

Le Hai Son, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5524–5527. IEEE.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 1–10. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Generalizing and Hybridizing Count-based and Neural Language Models

Graham Neubig[†] and **Chris Dyer**[‡]
[†]Carnegie Mellon University, USA
[‡]Google DeepMind, United Kingdom

Abstract

Language models (LMs) are statistical models that calculate probabilities over sequences of words or other discrete symbols. Currently two major paradigms for language modeling exist: count-based n-gram models, which have advantages of scalability and test-time speed, and neural LMs, which often achieve superior modeling performance. We demonstrate how both varieties of models can be unified in a single modeling framework that defines a set of probability distributions over the vocabulary of words, and then dynamically calculates mixture weights over these distributions. This formulation allows us to create novel hybrid models that combine the desirable features of count-based and neural LMs, and experiments demonstrate the advantages of these approaches.[1]

1 Introduction

Language models (LMs) are statistical models that, given a sentence $w_1^I := w_1, \ldots, w_I$, calculate its probability $P(w_1^I)$. LMs are widely used in applications such as machine translation and speech recognition, and because of their broad applicability they have also been widely studied in the literature. The most traditional and broadly used language modeling paradigm is that of count-based LMs, usually smoothed n-grams (Witten and Bell, 1991; Chen and Goodman, 1996). Recently, there has been a focus on LMs based on neural networks (Nakamura et al., 1990; Bengio et al., 2006; Mikolov et al., 2010), which have shown impressive improvements in performance over count-based LMs. On the other hand, these neural LMs also come at the cost of increased computational complexity at both training and test time, and even the largest reported neural LMs (Chen et al., 2015; Williams et al., 2015) are trained on a fraction of the data of their count-based counterparts (Brants et al., 2007).

In this paper we focus on a class of LMs, which we will call *mixture of distributions LMs* (MODLMs; §2). Specifically, we define MODLMs as all LMs that take the following form, calculating the probabilities of the next word in a sentence w_i given preceding context $\boldsymbol{c}$ according to a mixture of several component probability distributions $P_k(w_i|\boldsymbol{c})$:

$$P(w_i|\boldsymbol{c}) = \sum_{k=1}^{K} \lambda_k(\boldsymbol{c}) P_k(w_i|\boldsymbol{c}). \quad (1)$$

Here, $\lambda_k(\boldsymbol{c})$ is a function that defines the mixture weights, with the constraint that $\sum_{k=1}^{K} \lambda_k(\boldsymbol{c}) = 1$ for all $\boldsymbol{c}$. This form is not new in itself, and widely used both in the calculation of smoothing coefficients for n-gram LMs (Chen and Goodman, 1996), and interpolation of LMs of various varieties (Jelinek and Mercer, 1980).

The main contribution of this paper is to demonstrate that depending on our definition of $\boldsymbol{c}$, $\lambda_k(\boldsymbol{c})$, and $P_k(w_i|\boldsymbol{c})$, Eq. 1 can be used to describe not only n-gram models, but also feed-forward (Nakamura et al., 1990; Bengio et al., 2006; Schwenk, 2007) and

[1]Work was performed while GN was at the Nara Institute of Science and Technology and CD was at Carnegie Mellon University. Code and data to reproduce experiments is available at `http://github.com/neubig/modlm`

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1163–1172,
Austin, Texas, November 1-5, 2016.

recurrent (Mikolov et al., 2010; Sundermeyer et al., 2012) neural network LMs (§3). This observation is useful theoretically, as it provides a single mathematical framework that encompasses several widely used classes of LMs. It is also useful practically, in that this new view of these traditional models allows us to create new models that combine the desirable features of n-gram and neural models, such as:

neurally interpolated n-gram LMs (§4.1), which learn the interpolation weights of n-gram models using neural networks, and

neural/n-gram hybrid LMs (§4.2), which add a count-based n-gram component to neural models, allowing for flexibility to add large-scale external data sources to neural LMs.

We discuss learning methods for these models (§5) including a novel method of randomly dropping out more easy-to-learn distributions to prevent the parameters from falling into sub-optimal local minima.

Experiments on language modeling benchmarks (§6) find that these models outperform baselines in terms of performance and convergence speed.

2 Mixture of Distributions LMs

As mentioned above, MODLMs are LMs that take the form of Eq. 1. This can be re-framed as the following matrix-vector multiplication:

$$\boldsymbol{p}_{\boldsymbol{c}}^\intercal = D_{\boldsymbol{c}} \boldsymbol{\lambda}_{\boldsymbol{c}}^\intercal,$$

where $\boldsymbol{p}_{\boldsymbol{c}}$ is a vector with length equal to vocabulary size, in which the jth element $p_{\boldsymbol{c},j}$ corresponds to $P(w_i = j|\boldsymbol{c})$, $\boldsymbol{\lambda}_{\boldsymbol{c}}$ is a size K vector that contains the mixture weights for the distributions, and $D_{\boldsymbol{c}}$ is a J-by-K matrix, where element $d_{\boldsymbol{c},j,k}$ is equivalent to the probability $P_k(w_i = j|\boldsymbol{c})$.[2] An example of this formulation is shown in Fig. 1.

Note that all columns in D represent probability distributions, and thus must sum to one over the J words in the vocabulary, and that all $\boldsymbol{\lambda}$ must sum to 1 over the K distributions. Under this condition, the vector $\boldsymbol{p}$ will represent a well-formed probability distribution as well. This conveniently allows us to calculate the probability of a single word $w_i = j$ by calculating the product of the jth row of $D_{\boldsymbol{c}}$ and $\boldsymbol{\lambda}_{\boldsymbol{c}}^\intercal$

$$P_k(w_i = j|\boldsymbol{c}) = \boldsymbol{d}_{\boldsymbol{c},j} \boldsymbol{\lambda}_{\boldsymbol{c}}^\intercal.$$

In the sequel we show how this formulation can be used to describe several existing LMs (§3) as well as several novel model structures that are more powerful and general than these existing models (§4).

[2] We omit the subscript $\boldsymbol{c}$ when appropriate.

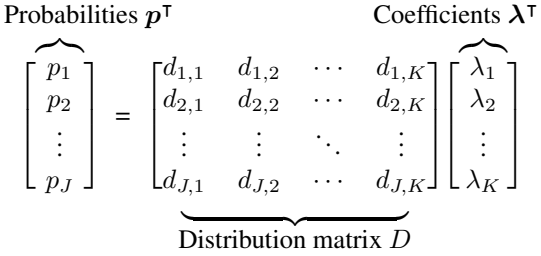


Figure 1: MODLMs as linear equations

3 Existing LMs as Linear Mixtures

3.1 n-gram LMs as Mixtures of Distributions

First, we discuss how count-based interpolated n-gram LMs fit within the MODLM framework.

Maximum likelihood estimation: n-gram models predict the next word based on the previous N-1 words. In other words, we set $\boldsymbol{c} = w_{i-N+1}^{i-1}$ and calculate $P(w_i|w_{i-N+1}^{i-1})$. The maximum-likelihood (ML) estimate for this probability is

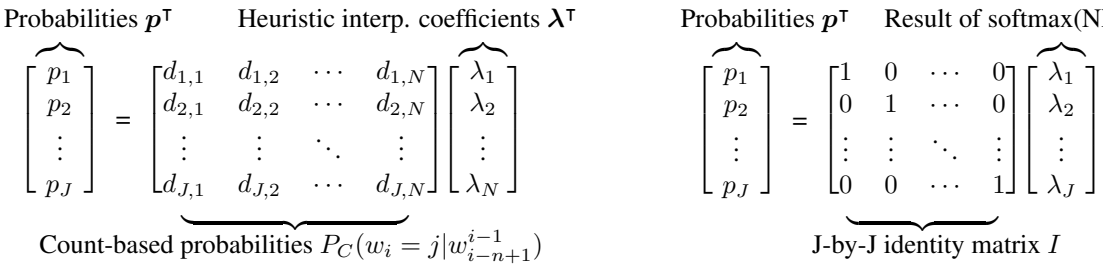
$$P_{ML}(w_i|w_{i-N+1}^{i-1}) = c(w_{i-N+1}^{i})/c(w_{i-N+1}^{i-1}),$$

where $c(\cdot)$ counts frequency in the training corpus.

Interpolation: Because ML estimation assigns zero probability to word sequences where $c(w_{i-N+1}^{i}) = 0$, n-gram models often interpolate the ML distributions for sequences of length 1 to N. The simplest form is static interpolation

$$P(w_i|w_{i-n+1}^{i-1}) = \sum_{n=1}^{N} \lambda_{S,n} P_{ML}(w_i|w_{i-n+1}^{i-1}). \quad (2)$$

$\boldsymbol{\lambda}_S$ is a vector where $\lambda_{S,n}$ represents the weight put on the distribution $P_{ML}(w_i|w_{i-n+1}^{i-1})$. This can be expressed as linear equations (Fig. 2a) by setting the nth column of D to the ML distribution $P_{ML}(w_i|w_{i-n+1}^{i-1})$, and $\boldsymbol{\lambda}(\boldsymbol{c})$ equal to $\boldsymbol{\lambda}_S$.



(a) Interpolated n-grams as MODLMs

(b) Neural LMs as MODLMs

Figure 2: Interpretations of existing models as mixtures of distributions

Static interpolation can be improved by calculating $\boldsymbol{\lambda}(\boldsymbol{c})$ dynamically, using heuristics based on the frequency counts of the context (Good, 1953; Katz, 1987; Witten and Bell, 1991). These methods define a context-sensitive fallback probability $\alpha(w_{i-n+1}^{i-1})$ for order n models, and recursively calculate the probability of the higher order models from the lower order models:

$$P(w_i|w_{i-n+1}^{i-1}) = \alpha(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1}) + (1 - \alpha(w_{i-n+1}^{i-1}))P_{ML}(w_i|w_{i-n+1}^{i-1}). \quad (3)$$

To express this as a linear mixture, we convert $\alpha(w_{i-n+1}^{i-1})$ into the appropriate value for $\lambda_n(w_{i-N+1}^{i-1})$. Specifically, the probability assigned to each $P_{ML}(w_i|w_{i-n+1}^{i-1})$ is set to the product of the fallbacks α for all higher orders and the probability of not falling back $(1 - \alpha)$ at the current level:

$$\lambda_n(w_{i-N+1}^{i-1}) = (1-\alpha(w_{i-n+1}^{i-1})) \prod_{\tilde{n}=n+1}^{N} \alpha(w_{i-\tilde{n}+1}^{i-1}).$$

Discounting: The widely used technique of discounting (Ney et al., 1994) defines a fixed discount d and subtracts it from the count of each word before calculating probabilities:

$$P_D(w_i|w_{i-n+1}^{i-1}) = (c(w_{i-n+1}^{i}) - d)/c(w_{i-n+1}^{i-1}).$$

Discounted LMs then assign the remaining probability mass after discounting as the fallback probability

$$\beta_D(w_{i-n+1}^{i-1}) = 1 - \sum_{j=1}^{J} P_D(w_i = j|w_{i-n+1}^{i-1}),$$
$$P(w_i|w_{i-n+1}^{i-1}) = \beta_D(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1}) + P_D(w_i|w_{i-n+1}^{i-1}). \quad (4)$$

In this case, $P_D(\cdot)$ does not add to one, and thus violates the conditions for MODLMs stated in §2, but it is easy to turn discounted LMs into interpolated LMs by normalizing the discounted distribution:

$$P_{ND}(w_i|w_{i-n+1}^{i-1}) = \frac{P_D(w_i|w_{i-n+1}^{i-1})}{\sum_{j=1}^{J} P_D(w_i = j|w_{i-n+1}^{i-1})},$$

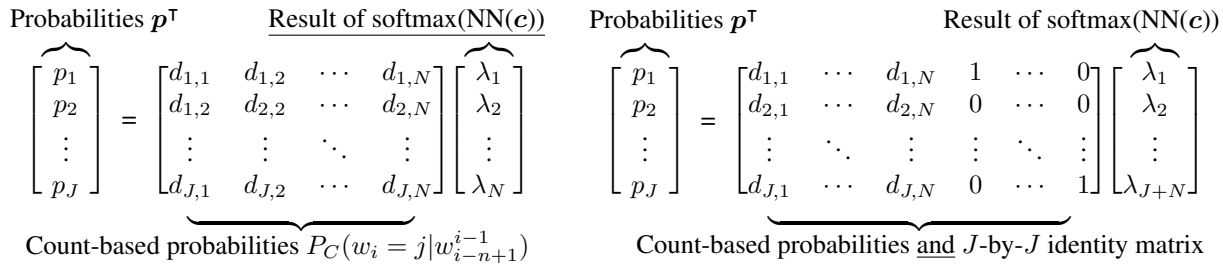
which allows us to replace $\beta(\cdot)$ for $\alpha(\cdot)$ and $P_{ND}(\cdot)$ for $P_{ML}(\cdot)$ in Eq. 3, and proceed as normal.

Kneser–Ney (KN; Kneser and Ney (1995)) and Modified KN (Chen and Goodman, 1996) smoothing further improve discounted LMs by adjusting the counts of lower-order distributions to more closely match their expectations as fallbacks for higher order distributions. Modified KN is currently the defacto standard in n-gram LMs despite occasional improvements (Teh, 2006; Durrett and Klein, 2011), and we will express it as $P_{KN}(\cdot)$.

3.2 Neural LMs as Mixtures of Distributions

In this section we demonstrate how neural network LMs can also be viewed as an instantiation of the MODLM framework.

Feed-forward neural network LMs: Feed-forward LMs (Bengio et al., 2006; Schwenk, 2007) are LMs that, like n-grams, calculate the probability of the next word based on the previous words. Given context w_{i-N+1}^{i-1}, these words are converted into real-valued word representation vectors $\boldsymbol{r}_{i-N+1}^{i-1}$, which are concatenated into an overall representation vector $\boldsymbol{q} = \oplus(\boldsymbol{r}_{i-N+1}^{i-1})$, where $\oplus(\cdot)$ is the vector concatenation function. $\boldsymbol{q}$ is then run through a series of affine transforms and nonlinearities defined as function NN($\boldsymbol{q}$) to obtain a vector $\boldsymbol{h}$. For example, for a one-layer neural net-



(a) Neurally interpolated n-gram LMs

(b) Neural/n-gram hybrid LMs

Figure 3: Two new expansions to n-gram and neural LMs made possible in the MODLM framework

work with a tanh non-linearity we can define

$$\text{NN}(\boldsymbol{q}) := \tanh(\boldsymbol{q}W_q + \boldsymbol{b}_q), \tag{5}$$

where W_q and $\boldsymbol{b}_q$ are weight matrix and bias vector parameters respectively. Finally, the probability vector $\boldsymbol{p}$ is calculated using the softmax function $\boldsymbol{p} = \text{softmax}(\boldsymbol{h}W_s + \boldsymbol{b}_s)$, similarly parameterized.

As these models are directly predicting $\boldsymbol{p}$ with no concept of mixture weights $\boldsymbol{\lambda}$, they cannot be interpreted as MODLMs as-is. However, we can perform a trick shown in Fig. 2b, not calculating $\boldsymbol{p}$ directly, but instead calculating mixture weights $\boldsymbol{\lambda} = \text{softmax}(\boldsymbol{h}W_s + \boldsymbol{b}_s)$, and defining the MODLM's distribution matrix D as a J-by-J identity matrix. This is equivalent to defining a linear mixture of J Kronecker δ_j distributions, the jth of which assigns a probability of 1 to word j and zero to everything else, and estimating the mixture weights with a neural network. While it may not be clear why it is useful to define neural LMs in this somewhat roundabout way, we describe in §4 how this opens up possibilities for novel expansions to standard models.

Recurrent neural network LMs: LMs using recurrent neural networks (RNNs) (Mikolov et al., 2010) consider not the previous few words, but also maintain a hidden state summarizing the sentence up until this point by re-defining the net in Eq. 5 as

$$\text{RNN}(\boldsymbol{q}_i) := \tanh(\boldsymbol{q}_i W_q + \boldsymbol{h}_{i-1} W_h + \boldsymbol{b}_q),$$

where $\boldsymbol{q}_i$ is the current input vector and $\boldsymbol{h}_{i-1}$ is the hidden vector at the previous time step. This allows for consideration of long-distance dependencies beyond the scope of standard n-grams, and LMs using RNNs or long short-term memory (LSTM) networks (Sundermeyer et al., 2012) have posted large improvements over standard n-grams and feed-forward models. Like feed-forward LMs, LMs using RNNs can be expressed as MODLMs by predicting $\boldsymbol{\lambda}$ instead of predicting $\boldsymbol{p}$ directly.

4 Novel Applications of MODLMs

This section describes how we can use this framework of MODLMs to design new varieties of LMs that combine the advantages of both n-gram and neural network LMs.

4.1 Neurally Interpolated n-gram Models

The first novel instantiation of MODLMs that we propose is *neurally interpolated n-gram models*, shown in Fig. 3a. In these models, we set D to be the same matrix used in n-gram LMs, but calculate $\boldsymbol{\lambda}(\boldsymbol{c})$ using a neural network model. As $\boldsymbol{\lambda}(\boldsymbol{c})$ is learned from data, this framework has the potential to allow us to learn more intelligent interpolation functions than the heuristics described in §3.1. In addition, because the neural network only has to calculate a softmax over N distributions instead of J vocabulary words, training and test efficiency of these models can be expected to be much greater than that of standard neural network LMs.

Within this framework, there are several design decisions. First, how we decide D: do we use the maximum likelihood estimate P_{ML} or KN estimated distributions P_{KN}? Second, what do we provide as input to the neural network to calculate the mixture weights? To provide the neural net with the same information used by interpolation heuristics used in traditional LMs, we first calculate three features for each of the N contexts w_{i-n+1}^{i-1}: a binary feature indicating whether the context has been observed in the training corpus ($c(w_{i-n+1}^{i-1}) > 0$), the log frequency of the context counts ($\log(c(w_{i-n+1}^{i-1}))$ or

zero for unobserved contexts), and the log frequency of the number of unique words following the context ($\log(u(w_{i-n+1}^{i-1}))$ or likewise zero). When using discounted distributions, we also use the log of the sum of the discounted counts as a feature. We can also optionally use the word representation vector $\boldsymbol{q}$ used in neural LMs, allowing for richer representation of the input, but this may or may not be necessary in the face of the already informative count-based features.

4.2 Neural/n-gram Hybrid Models

Our second novel model enabled by MODLMs is *neural/n-gram hybrid models*, shown in Fig. 3b. These models are similar to neurally interpolated n-grams, but D is augmented with J additional columns representing the Kronecker δ_j distributions used in the standard neural LMs. In this construction, $\boldsymbol{\lambda}$ is still a stochastic vector, but its contents are both the mixture coefficients for the count-based models and direct predictions of the probabilities of words. Thus, the learned LM can use count-based models when they are deemed accurate, and deviate from them when deemed necessary.

This model is attractive conceptually for several reasons. First, it has access to all information used by both neural and n-gram LMs, and should be able to perform as well or better than both models. Second, the efficiently calculated n-gram counts are likely sufficient to capture many phenomena necessary for language modeling, allowing the neural component to focus on learning only the phenomena that are not well modeled by n-grams, requiring fewer parameters and less training time. Third, it is possible to train n-grams from much larger amounts of data, and use these massive models to bootstrap learning of neural nets on smaller datasets.

5 Learning Mixtures of Distributions

While the MODLM formulations of standard heuristic n-gram LMs do not require learning, the remaining models are parameterized. This section discusses the details of learning these parameters.

5.1 Learning MODLMs

The first step in learning parameters is defining our training objective. Like most previous work on LMs (Bengio et al., 2006), we use a negative log-likelihood loss summed over words w_i in every sentence $\boldsymbol{w}$ in corpus $\mathcal{W}$

$$L(\mathcal{W}) = -\sum_{\boldsymbol{w}\in\mathcal{W}}\sum_{w_i\in\boldsymbol{w}} \log P(w_i|\boldsymbol{c}),$$

where $\boldsymbol{c}$ represents all words preceding w_i in $\boldsymbol{w}$ that are used in the probability calculation. As noted in Eq. 2, $P(w_i = j|\boldsymbol{c})$ can be calculated efficiently from the distribution matrix $D_{\boldsymbol{c}}$ and mixture function output $\boldsymbol{\lambda}_{\boldsymbol{c}}$.

Given that we can calculate the log likelihood, the remaining parts of training are similar to training for standard neural network LMs. As usual, we perform forward propagation to calculate the probabilities of all the words in the sentence, back-propagate the gradients through the computation graph, and perform some variant of stochastic gradient descent (SGD) to update the parameters.

5.2 Block Dropout for Hybrid Models

While the training method described in the previous section is similar to that of other neural network models, we make one important modification to the training process specifically tailored to the hybrid models of §4.2.

This is motivated by our observation (detailed in §6.3) that the hybrid models, despite being strictly more expressive than the corresponding neural network LMs, were falling into poor local minima with higher training error than neural network LMs. This is because at the very beginning of training, the count-based elements of the distribution matrix in Fig. 3b are already good approximations of the target distribution, while the weights of the single-word δ_j distributions are not yet able to provide accurate probabilities. Thus, the model learns to set the mixture proportions of the δ elements to near zero and rely mainly on the count-based n-gram distributions.

To encourage the model to use the δ mixture components, we adopt a method called *block dropout* (Ammar et al., 2016). In contrast to standard dropout (Srivastava et al., 2014), which drops out single nodes or connections, block dropout randomly drops out entire subsets of network nodes. In our case, we want to prevent the network from overusing the count-based n-gram distributions, so for a randomly selected portion of the training examples (here, 50%) we disable all n-gram distributions and

force the model to rely on only the δ distributions. To do so, we zero out all elements in $\boldsymbol{\lambda}(\boldsymbol{c})$ that correspond to n-gram distributions, and re-normalize over the rest of the elements so they sum to one.

5.3 Network and Training Details

Finally, we note design details that were determined based on preliminary experiments.

Network structures: We used both feed-forward networks with $\tanh$ non-linearities and LSTM (Hochreiter and Schmidhuber, 1997) networks. Most experiments used single-layer 200-node networks, and 400-node networks were used for experiments with larger training data. Word representations were the same size as the hidden layer. Larger and multi-layer networks did not yield improvements.

Training: We used ADAM (Kingma and Ba, 2015) with a learning rate of 0.001, and minibatch sizes of 512 words. This led to faster convergence than standard SGD, and more stable optimization than other update rules. Models were evaluated every 500k-3M words, and the model with the best development likelihood was used. In addition to the block dropout of §5.2, we used standard dropout with a rate of 0.5 for both feed-forward (Srivastava et al., 2014) and LSTM (Pham et al., 2014) nets in the neural LMs and neural/n-gram hybrids, but not in the neurally interpolated n-grams, where it resulted in slightly worse perplexities.

Features: If parameters are learned on the data used to train count-based models, they will heavily over-fit and learn to trust the count-based distributions too much. To prevent this, we performed 10-fold cross validation, calculating count-based elements of D for each fold with counts trained on the other 9/10. In addition, the count-based contextual features in §4.1 were normalized by subtracting the training set mean, which improved performance.

6 Experiments

6.1 Experimental Setup

In this section, we perform experiments to evaluate the neurally interpolated n-grams (§6.2) and neural/n-gram hybrids (§6.3), the ability of our models to take advantage of information from large data sets (§6.4), and the relative performance compared to post-facto static interpolation of already-trained models (§6.5). For the main experiments, we evaluate on two corpora: the Penn Treebank (PTB) data set prepared by Mikolov et al. (2010),[3] and the first 100k sentences in the English side of the ASPEC corpus (Nakazawa et al., 2015)[4] (details in Tab. 1). The PTB corpus uses the standard vocabulary of 10k words, and for the ASPEC corpus we use a vocabulary of the 20k most frequent words. Our implementation is included as supplementary material.

PTB	Sent	Word	ASP	Sent	Word
train	42k	890k	train	100k	2.1M
valid	3.4k	70k	valid	1.8k	45k
test	3.8k	79k	test	1.8k	46k

Table 1: Data sizes for the PTB and ASPEC corpora.

Dst./Ft.	HEUR	FF	LSTM
ML/C	220.5/265.9	146.6/164.5	144.4/162.7
ML/CR	-	145.7/163.9	142.6/158.4
KN/C	140.8/156.5	138.9/152.5	136.8/151.1
KN/CR	-	136.9/153.0	**135.2/149.1**

Table 2: PTB/ASPEC perplexities for traditional heuristic (HEUR) and proposed neural net (FF or LSTM) interpolation methods using ML or KN distributions, and count (C) or count+word representation (CR) features.

6.2 Results for Neurally Interpolated n-grams

First, we investigate the utility of neurally interpolated n-grams. In all cases, we use a history of $N = 5$ and test several different settings for the models:

Estimation type: $\boldsymbol{\lambda}(\boldsymbol{c})$ is calculated with heuristics (HEUR) or by the proposed method using feed-forward (FF), or LSTM nets.

Distributions: We compare $P_{ML}(\cdot)$ and $P_{KN}(\cdot)$. For heuristics, we use Witten-Bell for ML and the appropriate discounted probabilities for KN.

Input features: As input features for the neural network, we either use only the count-based features (C) or count-based features together with the word representation for the single previous word (CR).

From the results shown in Tab. 2, we can first see that when comparing models using the same set of

[3] http://rnnlm.org/simple-examples.tgz
[4] http://lotus.kuee.kyoto-u.ac.jp/ASPEC/

input distributions, the neurally interpolated model outperforms corresponding heuristic methods. We can also see that LSTMs have a slight advantage over FF nets, and models using word representations have a slight advantage over those that use only the count-based features. Overall, the best model achieves a relative perplexity reduction of 4-5% over KN models. Interestingly, even when using simple ML distributions, the best neurally interpolated n-gram model nearly matches the heuristic KN method, demonstrating that the proposed model can automatically learn interpolation functions that are nearly as effective as carefully designed heuristics.[5]

6.3 Results for Neural/n-gram Hybrids

In experiments with hybrid models, we test a neural/n-gram hybrid LM using LSTM networks with both Kronecker δ and KN smoothed 5-gram distributions, trained either with or without block dropout. As our main baseline, we compare to LSTMs with only δ distributions, which have reported competitive numbers on the PTB data set (Zaremba et al., 2014).[6] We also report results for heuristically smoothed KN 5-gram models, and the best neurally interpolated n-grams from the previous section for reference.

The results, shown in Tab. 3, demonstrate that similarly to previous research, LSTM LMs (2) achieve a large improvement in perplexity over n-gram models, and that the proposed neural/n-gram hybrid method (5) further reduces perplexity by 10-11% relative over this strong baseline.

Comparing models without (4) and with (5) the proposed block dropout, we can see that this method contributes significantly to these gains. To examine this more closely, we show the test perplexity for the three models using δ distributions in Fig. 5, and the amount of the probability mass in $\boldsymbol{\lambda}(\boldsymbol{c})$ assigned to the non-δ distributions in the hybrid models. From this, we can see that the model with block dropout quickly converges to a better result than the LSTM LM, but the model without converges to a worse result, assigning too much probability mass to the dense count-based distributions, demonstrating the learning problems mentioned in §5.2.

It is also of interest to examine exactly why the proposed model is doing better than the more standard methods. One reason can be found in the behavior with regards to low-frequency words. In Fig. 4, we show perplexities for words that appear n times or less in the training corpus, for $n = 10$, $n = 100$, $n = 1000$ and $n = \infty$ (all words). From the results, we can first see that if we compare the baselines, LSTM language models achieve better perplexities overall but n-gram language models tend to perform better on low-frequency words, corroborating the observations of Chen et al. (2015).

	Dist.	Interp.	PPL
(1)	KN	HEUR	140.8/156.5
(2)	δ	LSTM	105.9/116.9
(3)	KN	LSTM	135.2/149.1
(4)	KN,δ	LSTM -BIDO	108.4/130.4
(5)	KN,δ	LSTM +BIDO	95.3 /104.5

Table 3: PTB/ASPEC perplexities for traditional KN (1) and LSTM LMs (2), neurally interpolated n-grams (3), and neural/n-gram hybrid models without (4) and with (5) block dropout.

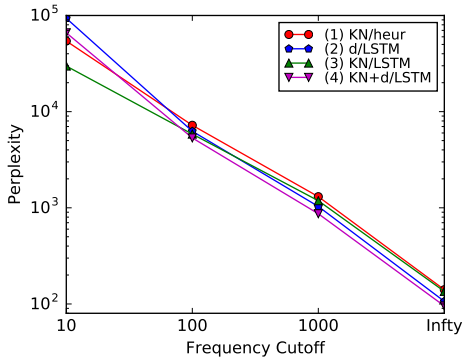


Figure 4: Perplexities of (1) standard n-grams, (2) standard LSTMs, (3) neurally interpolated n-grams, and (4) neural/n-gram hybrids on lower frequency words.

[5]Neurally interpolated n-grams are also more efficient than standard neural LMs, as mentioned in §4.1. While a standard LSTM LM calculated 1.4kw/s on the PTB data, the neurally interpolated models using LSTMs and FF nets calculated 11kw/s and 58kw/s respectively, only slightly inferior to 140kw/s of heuristic KN.

[6]Note that unlike this work, we opt to condition only on in-sentence context, not inter-sentential dependencies, as training through gradient calculations over sentences is more straightforward and because examining the effect of cross-boundary information is not central to the proposed method. Thus our baseline numbers are not directly comparable (i.e. have higher perplexity) to previous reported results on this data, but we still feel that the comparison is appropriate.

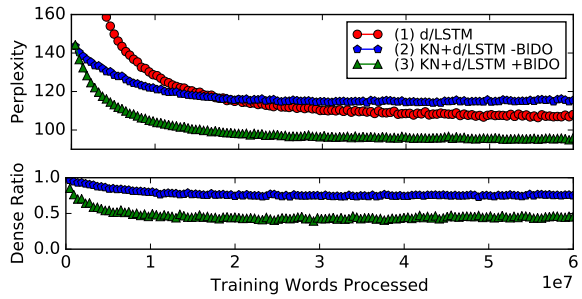


Figure 5: Perplexity and dense distribution ratio of the baseline LSTM LM (1), and the hybrid method without (2) and with (3) block dropout.

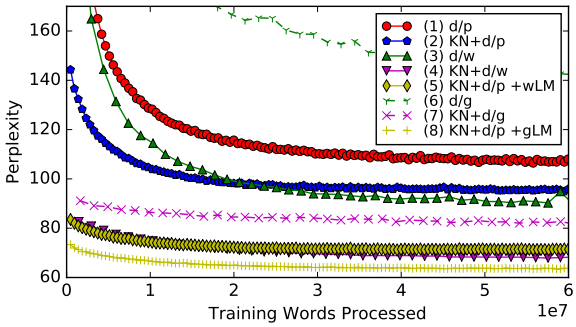


Figure 6: Models trained on PTB (1,2), PTB+WSJ (3,4,5) or PTB+WSJ+GW (6,7,8) using standard neural LMs (1,3,6), neural/n-gram hybrids trained all data (2,4,7), or hybrids trained on PTB with additional n-gram distributions (5,8).

The neurally interpolated n-gram models consistently outperform standard KN-smoothed n-grams, demonstrating their superiority within this model class. In contrast, the neural/n-gram hybrid models tend to follow a pattern more similar to that of LSTM language models, similarly with consistently higher performance.

6.4 Results for Larger Data Sets

To examine the ability of the hybrid models to use counts trained over larger amounts of data, we perform experiments using two larger data sets:

WSJ: The PTB uses data from the 1989 Wall Street Journal, so we add the remaining years between 1987 and 1994 (1.81M sents., 38.6M words).

GW: News data from the English Gigaword 5th Edition (LDC2011T07, 59M sents., 1.76G words).

We incorporate this data either by training net parameters over the whole large data, or by separately training count-based n-grams on each of PTB, WSJ, and GW, and learning net parameters on only PTB data. The former has the advantage of training the net on much larger data. The latter has two main advantages: 1) when the smaller data is of a particular domain the mixture weights can be learned to match this in-domain data; 2) distributions can be trained on data such as Google n-grams (LDC2006T13), which contain n-gram counts but not full sentences.

In the results of Fig. 6, we can first see that the neural/n-gram hybrids significantly outperform the traditional neural LMs in the scenario with larger data as well. Comparing the two methods for incorporating larger data, we can see that the results are mixed depending on the type and size of the data being used. For the WSJ data, training on all data slightly outperforms the method of adding distributions, but when the GW data is added this trend reverses. This can be explained by the fact that the GW data differs from the PTB test data, and thus the effect of choosing domain-specific interpolation coefficients was more prominent.

6.5 Comparison with Static Interpolation

Finally, because the proposed neural/n-gram hybrid models combine the advantages of neural and n-gram models, we compare with the more standard method of training models independently and combining them with static interpolation weights tuned on the validation set using the EM algorithm. Tab. 4 shows perplexities for combinations of a standard neural model (or δ distributions) trained on PTB, and count based distributions trained on PTB, WSJ, and GW are added one-by-one using the standard static and proposed LSTM interpolation methods. From the results, we can see that when only PTB data is used, the methods have similar results, but with the more diverse data sets the proposed method edges out its static counterpart.[7]

[7] In addition to better perplexities, neural/n-gram hybrids are trained in a single pass instead of performing post-facto interpolation, which may give advantages when training for other objectives (Auli and Gao, 2014; Li et al., 2015).

Interp	δ+PTB	+WSJ	+GW
Lin.	**95.1**	70.5	65.8
LSTM	95.3	**68.3**	**63.5**

Table 4: PTB perplexity for interpolation between neural (δ) LMs and count-based models.

7 Related Work

A number of alternative methods focus on interpolating LMs of multiple varieties such as in-domain and out-of-domain LMs (Bulyko et al., 2003; Bacchiani et al., 2006; Gülçehre et al., 2015). Perhaps most relevant is Hsu (2007)'s work on learning to interpolate multiple LMs using log-linear models. This differs from our work in that it learns functions to estimate the fallback probabilities $\alpha_n(\boldsymbol{c})$ in Eq. 3 instead of $\boldsymbol{\lambda}(\boldsymbol{c})$, and does not cover interpolation of n-gram components, non-linearities, or the connection with neural network LMs. Also conceptually similar is work on adaptation of n-gram LMs, which start with n-gram probabilities (Della Pietra et al., 1992; Kneser and Steinbiss, 1993; Rosenfeld, 1996; Iyer and Ostendorf, 1999) and adapt them based on the distribution of the current document, albeit in a linear model. There has also been work incorporating binary n-gram features into neural language models, which allows for more direct learning of n-gram weights (Mikolov et al., 2011), but does not afford many of the advantages of the proposed model such as the incorporation of count-based probability estimates. Finally, recent works have compared n-gram and neural models, finding that neural models often perform better in perplexity, but n-grams have their own advantages such as effectiveness in extrinsic tasks (Baltescu and Blunsom, 2015) and better modeling of rare words (Chen et al., 2015).

8 Conclusion and Future Work

In this paper, we proposed a framework for language modeling that generalizes both neural network and count-based n-gram LMs. This allowed us to learn more effective interpolation functions for count-based n-grams, and to create neural LMs that incorporate information from count-based models.

As the framework discussed here is general, it is also possible that they could be used in other tasks that perform sequential prediction of words such as neural machine translation (Sutskever et al., 2014) or dialog response generation (Sordoni et al., 2015). In addition, given the positive results using block dropout for hybrid models, we plan to develop more effective learning methods for mixtures of sparse and dense distributions.

Acknowledgements

We thank Kevin Duh, Austin Matthews, Shinji Watanabe, and anonymous reviewers for valuable comments on earlier drafts. This work was supported in part by JSPS KAKENHI Grant Number 16H05873, and the Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. One parser, many languages. *CoRR*, abs/1602.01595.

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected bleu training for recurrent neural network language models. In *Proc. ACL*, pages 136–142.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. Map adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *Proc. NAACL*, pages 820–829.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, volume 194, pages 137–186.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proc. EMNLP*, pages 858–867.

Ivan Bulyko, Mari Ostendorf, and Andreas Stolcke. 2003. Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures. In *Proc. HLT*, pages 7–9.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. ACL*, pages 310–318.

W. Chen, D. Grangier, and M. Auli. 2015. Strategies for Training Large Vocabulary Neural Language Models. *ArXiv e-prints*, December.

Stephen Della Pietra, Vincent Della Pietra, Robert L Mercer, and Salim Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. In *Proc. ACL*, pages 103–106.

Greg Durrett and Dan Klein. 2011. An empirical investigation of discounting in cross-domain language models. In *Proc. ACL*.

Irving J Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

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

Bo-June Hsu. 2007. Generalized linear interpolation of language models. In *Proc. ASRU*, pages 136–140.

Rukmini M Iyer and Mari Ostendorf. 1999. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *Speech and Audio Processing, IEEE Transactions on*, 7(1):30–39.

Frederick Jelinek and Robert Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *Workshop on pattern recognition in practice*.

Slava M Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proc. ICLR*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. ICASSP*, volume 1, pages 181–184. IEEE.

Reinhard Kneser and Volker Steinbiss. 1993. On the dynamic adaptation of stochastic language models. In *Proc. ICASSP*, pages 586–589.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. InterSpeech*, pages 1045–1048.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Proc. ASRU*, pages 196–201. IEEE.

Masami Nakamura, Katsuteru Maruyama, Takeshi Kawabata, and Kiyohiro Shikano. 1990. Neural network approach to word category prediction for English texts. In *Proc. COLING*.

Toshiaki Nakazawa, Hideya Mino, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2015. Overview of the 2nd Workshop on Asian Translation. In *Proc. WAT*.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8(1):1–38.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Proc. ICFHR*, pages 285–290.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10(3):187–228.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. NAACL*, pages 196–205.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proc. InterSpeech*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical report, School of Computing, National Univ. of Singapore.

Will Williams, Niranjani Prasad, David Mrva, Tom Ash, and Tony Robinson. 2015. Scaling recurrent neural network language models. In *Proc. ICASSP*.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Reasoning about Pragmatics with Neural Listeners and Speakers

Jacob Andreas and **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,klein}@cs.berkeley.edu

Abstract

We present a model for contrastively describing scenes, in which context-specific behavior results from a combination of inference-driven pragmatics and learned semantics. Like previous learned approaches to language generation, our model uses a simple feature-driven architecture (here a pair of neural "listener" and "speaker" models) to ground language in the world. Like inference-driven approaches to pragmatics, our model actively reasons about listener behavior when selecting utterances. For training, our approach requires only ordinary captions, annotated *without* demonstration of the pragmatic behavior the model ultimately exhibits. In human evaluations on a referring expression game, our approach succeeds 81% of the time, compared to 69% using existing techniques.

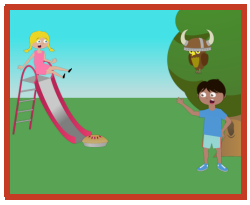
1 Introduction

We present a model for describing scenes and objects by reasoning about context and listener behavior. By incorporating standard neural modules for image retrieval and language modeling into a probabilistic framework for pragmatics, our model generates rich, contextually appropriate descriptions of structured world representations.

This paper focuses on a *reference game* RG played between a listener L and a speaker S.

1. Reference candidates r_1 and r_2 are revealed to both players.
2. S is secretly assigned a random target $t \in \{1, 2\}$.
3. S produces a description $d = S(t, r_1, r_2)$, which is shown to L.
4. L chooses $c = L(d, r_1, r_2)$.
5. Both players win if $c = t$.

(RG)



(a) target

(b) distractor

the owl is sitting in the tree

(c) description

Figure 1: Sample output from our model. When presented with a target image (a) in contrast with a distractor image (b), the model generates a description (c). This description mentions a *tree*, the distinguishing object present in (a) but not in (b), and situates it with respect to other objects and events in the scene.

Figure 1 shows an example drawn from a standard captioning dataset (Zitnick et al., 2014).

In order for the players to win, S's description d must be *pragmatic*: it must be informative, fluent, concise, and must ultimately encode an understanding of L's behavior. In Figure 1, for example, *the owl is wearing a hat* and *the owl is sitting in the tree* are both accurate descriptions of the target image, but only the second allows a human listener to succeed with high probability. RG is the focus of many papers in the computational pragmatics literature: it provides a concrete generation task while eliciting a broad range of pragmatic behaviors, including conversational implicature (Benotti and Traum, 2009) and context dependence (Smith et al., 2013). Existing computational models of pragmatics can be divided into two broad lines of work, which we term the *direct* and *derived* approaches.

Direct models (see Section 2 for examples) are based on a representation of S. They learn pragmatic behavior by example. Beginning with datasets annotated for the specific task they are trying to

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182,
Austin, Texas, November 1-5, 2016.

solve (e.g. examples of humans playing RG), direct models use feature-based architectures to predict appropriate behavior without a listener representation. While quite general in principle, such models require training data annotated specifically with pragmatics in mind; such data is scarce in practice.

Derived models, by contrast, are based on a representation of L. They first instantiate a *base listener* L0 (intended to simulate a naïve, non-pragmatic listener). They then form a *reasoning speaker* S1, which chooses a description that causes L0 to behave correctly. Existing derived models couple hand-written grammars and hand-engineered listener models with sophisticated inference procedures. They exhibit complex behavior, but are restricted to small domains where grammar engineering is practical.

The approach we present in this paper aims to capture the best aspects of both lines of work. Like direct approaches, we use machine learning to acquire a complete grounded generation model from data, without domain knowledge in the form of a hand-written grammar or hand-engineered listener model. But like derived approaches, we use this learning to construct a *base* model, and embed it within a higher-order model that reasons about listener responses. As will be seen, this reasoning step allows the model to make use of weaker supervision than previous data-driven approaches, while exhibiting robust behavior in a variety of contexts.

Our goal is to build a derived model that scales to real-world datasets without domain engineering. Independent of the application to RG, our model also belongs to the family of neural image captioning models that have been a popular subject of recent study (Xu et al., 2015). Nevertheless, our approach appears to be:

- the first such captioning model to reason explicitly about listeners
- the first learned approach to pragmatics that requires only *non-pragmatic* training data

Following previous work, we evaluate our model on RG, though the general architecture could be applied to other tasks where pragmatics plays a core role. Using a large dataset of abstract scenes like the one shown in Figure 1, we run a series of games with humans in the role of L and our system in the role of S. We find that the descriptions generated by our model result in correct interpretation 17% more often than a recent learned baseline system. We use these experiments to explore various other aspects of computational pragmatics, including tradeoffs between adequacy and fluency, and between computational efficiency and expressive power.[1]

2 Related Work

Direct pragmatics As an example of the direct approach mentioned in the introduction, FitzGerald et al. (2013) collect a set of human-generated referring expressions about abstract representations of sets of colored blocks. Given a set of blocks to describe, their model directly learns a maximum-entropy distribution over the set of logical expressions whose denotation is the target set. Other research, focused on referring expression generation from a computer vision perspective, includes that of Mao et al. (2015) and Kazemzadeh et al. (2014).

Derived pragmatics Derived approaches, sometimes referred to as "rational speech acts" models, include those of Smith et al. (2013), Vogel et al. (2013), Golland et al. (2010), and Monroe and Potts (2015). These couple template-driven language generation with probabilistic or game-theoretic reasoning frameworks to produce contextually appropriate language: intelligent listeners reason about the behavior of reflexive speakers, and even higher-order speakers reason about these listeners. Experiments (Frank et al., 2009) show that derived approaches explain human behavior well, but both computational and representational issues restrict their application to simple reference games. They require domain-specific engineering, controlled world representations, and pragmatically annotated training data.

An extensive literature on computational pragmatics considers its application to tasks other than RG, including instruction following (Anderson et al., 1991) and discourse analysis (Jurafsky et al., 1997).

[1] Models, human annotations, and code to generate all tables and figures in this paper can be found at `http://github.com/jacobandreas/pragma`.

Representing language and the world In addition to the pragmatics literature, the approach proposed in this paper relies extensively on recently developed tools for multimodal processing of language and unstructured representations like images. These includes both image retrieval models, which select an image from a collection given a textual description (Socher et al., 2014), and neural conditional language models, which take a content representation and emit a string (Donahue et al., 2015).

3 Approach

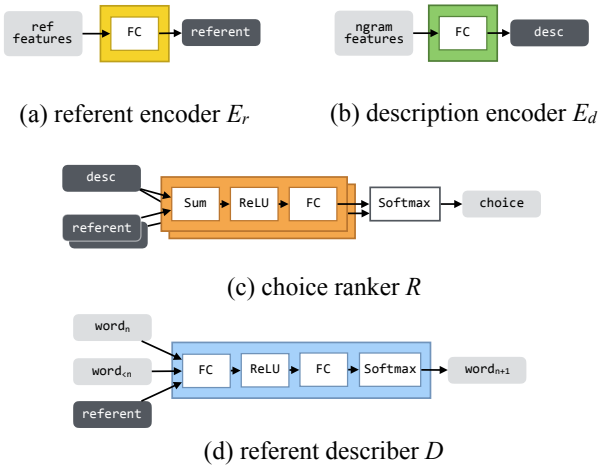
Our goal is to produce a model that can play the role of the speaker S in RG. Specifically, given a target referent (e.g. scene or object) r and a distractor r', the model must produce a description d that uniquely identifies r. For training, we have access to a set of *non-contrastively* captioned referents $\{(r_i, d_i)\}$: each training description d_i is generated for its associated referent r_i in isolation. There is no guarantee that d_i would actually serve as a good referring expression for r_i in any particular context. We must thus use the training data to ground language in referent representations, but rely on reasoning to produce pragmatics.

Our model architecture is compositional and hierarchical. We begin in Section 3.2 by describing a collection of "modules": basic computational primitives for mapping between referents, descriptions, and reference judgments, here implemented as linear operators or small neural networks. While these modules appear as substructures in neural architectures for a variety of tasks, we put them to novel use in constructing a reasoning pragmatic speaker.

Section 3.3 describes how to assemble two base models: a *literal speaker*, which maps from referents to strings, and a *literal listener*, which maps from strings to reference judgments. Section 3.4 describes how these base models are used to implement a top-level *reasoning speaker*: a learned, probabilistic, derived model of pragmatics.

3.1 Preliminaries

Formally, we take a description d to consist of a sequence of words $d_1, d_2, \ldots, d_n$, drawn from a vocabulary of known size. For encoding, we also assume access to a feature representation $f(d)$ of the sentence (for purposes of this paper, a vector of indicator features on n-grams). These two views—as a sequence of words d_i and a feature vector $f(d)$—form the basis of module interactions with language.



(a) referent encoder E_r (b) description encoder E_d

(c) choice ranker R

(d) referent describer D

Figure 2: Diagrams of modules used to construct speaker and listener models. "FC" is a fully-connected layer (a matrix multiply) and "ReLU" is a rectified linear unit. The encoder modules (a,b) map from feature representations (in gray) to embeddings (in black), while the ranker (c) and describer modules (d) respectively map from embeddings to decisions and strings.

Referent representations are similarly simple. Because the model never generates referents—only conditions on them and scores them—a vector-valued feature representation of referents suffices. Our approach is completely indifferent to the nature of this representation. While the experiments in this paper use a vector of indicator features on objects and actions present in abstract scenes (Figure 1), it would be easy to instead use pre-trained convolutional representations for referring to natural images. As with descriptions, we denote this feature representation $f(r)$ for referents.

3.2 Modules

All listener and speaker models are built from a kit of simple building blocks for working with multimodal representations of images and text:

1. a **referent encoder** E_r
2. a **description encoder** E_d
3. a **choice ranker** R
4. a **referent describer** D

These are depicted in Figure 2, and specified more formally below. All modules are parameterized by weight matrices, written with capital letters W_1, W_2, etc.; we refer to the collection of weights for all modules together as W.

Encoders The referent and description encoders produce a linear embedding of referents and descriptions in a common vector space.

$$\text{Referent encoder:} \quad E_r(r) = W_1 f(r) \tag{1}$$

$$\text{Description encoder:} \quad E_d(d) = W_2 f(d) \tag{2}$$

Choice ranker The choice ranker takes a string encoding and a collection of referent encodings, assigns a score to each (string, referent) pair, and then transforms these scores into a distribution over referents. We write $R(e_i|e_{-i}, e_d)$ for the probability of choosing i in contrast to the alternative; for example, $R(e_2|e_1, e_d)$ is the probability of answering "2" when presented with encodings e_1 and e_2.

$$\begin{aligned} s_1 &= w_3^\top \rho(W_4 e_1 + W_5 e_d) \\ s_2 &= w_3^\top \rho(W_4 e_2 + W_5 e_d) \\ R(e_i|e_{-i}, e_d) &= \frac{e^{s_i}}{e^{s_1} + e^{s_2}} \end{aligned} \tag{3}$$

(Here ρ is a rectified linear activation function.)

Referent describer The referent describer takes an image encoding and outputs a description using a (feedforward) conditional neural language model. We express this model as a distribution $p(d_{n+1}|d_n, d_{<n}, e_r)$, where d_n is an indicator feature on the last description word generated, $d_{<n}$ is a vector of indicator features on all other words previously generated, and e_r is a referent embedding. This is a "2-plus-skip-gram" model, with local positional history features, global position-independent history features, and features on the referent being described. To implement this probability distribution, we first use a multilayer perceptron to compute a vector of scores s (one s_i for each vocabulary item): $s = W_6 \rho(W_7[d_n, d_{<n}, e_i])$. We then normalize these to obtain probabilities: $p_i = e^{s_i} / \sum_j e^{s_j}$. Finally, $p(d_{n+1}|d_n, d_{<n}, e_r) = p_{d_{n+1}}$.

3.3 Base models

From these building blocks, we construct a pair of base models. The first of these is a **literal listener** L0, which takes a description and a set of referents, and chooses the referent most likely to be described. This serves the same purpose as the base listener in the general derived approach described in the introduction. We additionally construct a **literal speaker** S0, which takes a referent in isolation and outputs a description. The literal speaker is used for efficient inference over the space of possible descriptions, as described in Section 3.4. L0 is, in essence, a retrieval model, and S0 is neural captioning model.

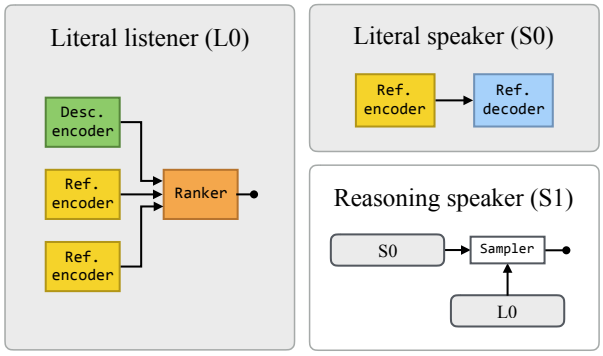


Figure 3: Schematic depictions of models. The literal listener L0 maps from descriptions and reference candidates to reference decisions. The literal speaker S0 maps directly from scenes to descriptions, ignoring context, while the reasoning speaker uses samples from S0 and scores from both L0 and S0 to produce contextually-appropriate captions.

Both of the base models are probabilistic: L0 produces a distribution over referent choices, and S0 produces a distribution over strings. They are depicted with shaded backgrounds in Figure 3.

Literal listener Given a description d and a pair of candidate referents r_1 and r_2, the literal listener embeds both referents and passes them to the ranking module, producing a distribution over choices i.

$$\begin{aligned} e_d &= E_d(d) \\ e_1 &= E_r(r_1) \\ e_2 &= E_r(r_2) \\ p_{\text{L0}}(i|d, r_1, r_2) &= R(e_i|e_{-i}, e_d) \end{aligned} \tag{4}$$

That is, $p_{\text{L0}}(1|d, r_1, r_2) = R(e_1|e_2, e_d)$ and vice-versa. This model is trained contrastively, by solving the following optimization problem:

$$\max_W \sum_j \log p_{\text{L0}}(1|d_j, r_j, r') \tag{5}$$

Here r' is a random distractor chosen uniformly from the training set. For each training example (r_i, d_i), this objective attempts to maximize the probability that the model chooses r_i as the referent of d_i over a random distractor.

This contrastive objective ensures that our approach is applicable even when there is not a naturally-occurring source of target–distractor pairs, as previous work (Golland et al., 2010; Monroe and Potts, 2015) has required. Note that this can also be viewed as a version of the loss described by Smith and Eisner (2005), where it approximates a likelihood objective that encourages L0 to prefer r_i to every other possible referent simultaneously.

Literal speaker As in the figure, the literal speaker is obtained by composing a referent encoder with a describer, as follows:

$$e = E_r(f(r))$$
$$p_{\text{S0}}(d|r) = D_d(d|e)$$

As with the listener, the literal speaker should be understood as producing a distribution over strings. It is trained by maximizing the conditional likelihood of captions in the training data:

$$\max_W \sum_i \log p_{\text{S0}}(d_i|r_i) \tag{6}$$

These base models are intended to be the minimal learned equivalents of the hand-engineered speakers and hand-written grammars employed in previous derived approaches (Golland et al., 2010). The neural encoding/decoding framework implemented by the modules in the previous subsection provides a simple way to map from referents to descriptions and descriptions to judgments without worrying too much about the details of syntax or semantics. Past work amply demonstrates that neural conditional language models are powerful enough to generate fluent and accurate (though not necessarily pragmatic) descriptions of images or structured representations (Donahue et al., 2015).

3.4 Reasoning model

As described in the introduction, the general derived approach to pragmatics constructs a base listener and then selects a description that makes it behave correctly. Since the assumption that listeners will behave deterministically is often a poor one, it is common for such derived approaches to implement *probabilistic* base listeners, and maximize the probability of correct behavior.

The neural literal listener L0 described in the preceding section is such a probabilistic listener. Given a target i and a pair of candidate referents r_1 and r_2, it is natural to specify the behavior of a reasoning speaker as simply:

$$\max_d p_{\text{L0}}(i|d, r_1, r_2) \tag{7}$$

At a first glance, the only thing necessary to implement this model is the representation of the literal listener itself. When the set of possible utterances comes from a fixed vocabulary (Vogel et al., 2013) or a grammar small enough to exhaustively enumerate (Smith et al., 2013) the operation $\max_d$ in Equation 7 is practical.

For our purposes, however, we would like the model to be capable of producing arbitrary utterances. Because the score p_{L0} is produced by a discriminative listener model, and does not factor along the words of the description, there is no dynamic program that enables efficient inference over the space of all strings.

We instead use a sampling-based optimization procedure. The key ingredient here is a good *proposal distribution* from which to sample sentences likely to be assigned high weight by the model listener. For this we turn to the literal speaker S0 described in the previous section. Recall that this speaker produces a distribution over plausible descriptions of isolated images, while ignoring pragmatic context. We can use it as a source of candidate descriptions, to be reweighted according to the expected behavior of L0. The full specification of a sampling neural reasoning speaker is as follows:

1. Draw samples $d_1, \ldots d_n \sim p_{\text{S0}}(\cdot|r_i)$.
2. Score samples: $p_k = p_{\text{L0}}(i|d_k, r_1, r_2)$.
3. Select d_k with $k = \arg\max p_k$.

While primarily to enable efficient inference, we can also use the literal speaker to serve a different purpose: "regularizing" model behavior towards choices that are adequate and fluent, rather than exploiting strange model behavior. Past work has re-

stricted the set of utterances in a way that guarantees fluency. But with an imperfect learned listener model, and a procedure that optimizes this listener's judgments directly, the speaker model might accidentally discover the kinds of pathological optima that neural classification models are known to exhibit (Goodfellow et al., 2014)—in this case, sentences that cause exactly the right response from L0, but no longer bear any resemblance to human language use. To correct this, we allow the model to consider two questions: as before, "how likely is it that a listener would interpret this sentence correctly?", but additionally "how likely is it that a speaker would produce it?"

Formally, we introduce a parameter λ that trades off between L0 and S0, and take the reasoning model score in step 2 above to be:

$$p_k = p_{\text{S0}}(d_k|r_i)^{\lambda} \cdot p_{\text{L0}}(i|d_k, r_1, r_2)^{1-\lambda} \quad (8)$$

This can be viewed as a weighted *joint* probability that a sentence is both uttered by the literal speaker and correctly interpreted by the literal listener, or alternatively in terms of Grice's conversational maxims (Grice, 1970): L0 encodes the maxims of *quality* and *relation*, ensuring that the description contains enough information for L to make the right choice, while S0 encodes the maxim of *manner*, ensuring that the description conforms with patterns of human language use. Responsibility for the maxim of *quantity* is shared: L0 ensures that the model doesn't say too little, and S0 ensures that the model doesn't say too much.

4 Evaluation

We evaluate our model on the reference game RG described in the introduction. In particular, we construct instances of RG using the Abstract Scenes Dataset introduced by Zitnick and Parikh (2013). Example scenes are shown in Figure 1 and Figure 4. The dataset contains pictures constructed by humans and described in natural language. Scene representations are available both as rendered images and as feature representations containing the identity and location of each object; as noted in Section 3.1, we use this feature set to produce our referent representation $f(r)$. This dataset was previously used for a variety of language and vision tasks (e.g. Ortiz et al. (2015), Zitnick et al. (2014)). It consists of 10,020 scenes, each annotated with up to 6 captions.

The abstract scenes dataset provides a more challenging version of RG than anything we are aware of in the existing computational pragmatics literature, which has largely used the TUNA corpus of isolated object descriptions (Gatt et al., 2007) or small synthetic datasets (Smith et al., 2013). By contrast, the abstract scenes data was generated by humans looking at complex images with numerous objects, and features grammatical errors, misspellings, and a vocabulary an order of magnitude larger than TUNA. Unlike previous work, we have no prespecified in-domain grammar, and no direct supervision of the relationship between scene features and lexemes.

We perform a human evaluation using Amazon Mechanical Turk. We begin by holding out a development set and a test set; each held-out set contains 1000 scenes and their accompanying descriptions. For each held-out set, we construct two sets of 200 paired (target, distractor) scenes: **All**, with up to four differences between paired scenes, and **Hard**, with exactly one difference between paired scenes. (We take the number of differences between scenes to be the number of objects that appear in one scene but not the other.)

We report two evaluation metrics. *Fluency* is determined by showing human raters isolated sentences, and asking them to rate linguistic quality on a scale from 1–5. *Accuracy* is success rate at RG: as in Figure 1, humans are shown two images and a model-generated description, and asked to select the image matching the description.

In the remainder of this section, we measure the tradeoff between fluency and accuracy that results from different mixtures of the base models (Section 4.1), measure the number of samples needed to obtain good performance from the reasoning listener (Section 4.2), and attempt to approximate the reasoning listener with a monolithic "compiled" listener (Section 4.3). In Section 4.4 we report final accuracies for our approach and baselines.

# samples	1	10	100	1000
Accuracy (%)	66	75	83	85

Table 1: S1 accuracy vs. number of samples.

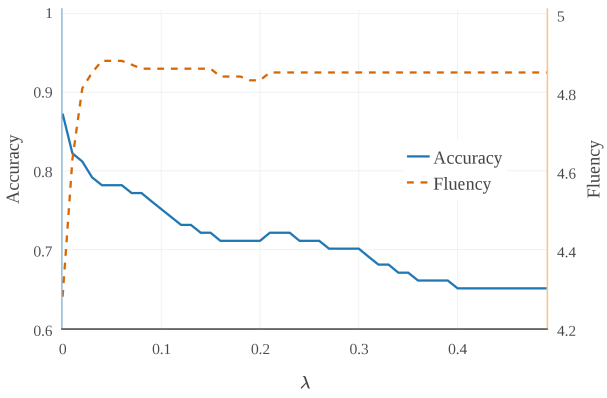


Figure 5: Tradeoff between speaker and listener models, controlled by the parameter λ in Equation 8. With $\lambda = 0$, all weight is placed on the literal listener, and the model produces highly discriminative but somewhat disfluent captions. With $\lambda = 1$, all weight is placed on the literal speaker, and the model produces fluent but generic captions.

4.1 How good are the base models?

To measure the performance of the base models, we draw 10 samples d_{jk} for a subset of 100 pairs $(r_{1,j}, r_{2,j})$ in the Dev-All set. We collect human fluency and accuracy judgments for each of the 1000 total samples. This allows us to conduct a post-hoc search over values of λ: for a range of λ, we compute the average accuracy and fluency of the highest scoring sample. By varying λ, we can view the tradeoff between accuracy and fluency that results from interpolating between the listener and speaker model—setting $\lambda = 0$ gives samples from p_{L0}, and $\lambda = 1$ gives samples from p_{S0}.

Figure 5 shows the resulting accuracy and fluency for various values of λ. It can be seen that relying entirely on the listener gives the highest accuracy but degraded fluency. However, by adding only a very small weight to the speaker model, it is possible to achieve near-perfect fluency without a substantial decrease in accuracy. Example sentences for an individual reference game are shown in Figure 5; increasing λ causes captions to become more generic. For the remaining experiments in this paper, we take $\lambda = 0.02$, finding that this gives excellent performance on both metrics.

On the development set, $\lambda = 0.02$ results in an **average fluency of 4.8** (compared to 4.8 for the literal speaker $\lambda = 1$). This high fluency can be confirmed by inspection of model samples (Figure 4). We thus focus on **accuracy** or the remainder of the evaluation.

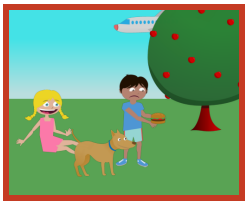
4.2 How many samples are needed?

Next we turn to the computational efficiency of the reasoning model. As in all sampling-based inference, the number of samples that must be drawn from the proposal is of critical interest—if too many samples are needed, the model will be too slow to use in practice. Having fixed $\lambda = 0.02$ in the preceding section, we measure accuracy for versions of the reasoning model that draw 1, 10, 100, and 1000 samples. Results are shown in Table 1. We find that gains continue up to 100 samples.

4.3 Is reasoning necessary?

Because they do not require complicated inference procedures, direct approaches to pragmatics typically enjoy better computational efficiency than derived ones. Having built an accurate derived speaker, can we bootstrap a more efficient direct speaker?

To explore this, we constructed a "compiled" speaker model as follows: Given reference candidates r_1 and r_2 and target t, this model produces embeddings e_1 and e_2, concatenates them together into a "contrast embedding" $[e_t, e_{-t}]$, and then feeds this whole embedding into a string decoder module. Like S0, this model generates captions without the need for discriminative rescoring; unlike S0, the contrast embedding means this model can in principle learn to produce pragmatic captions, if given access to pragmatic training data. Since no such training data exists, we train the compiled model on

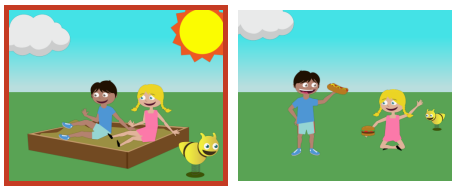


(a) target

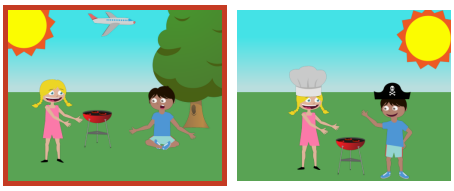
(b) distractor

(prefer L0)	`0.0`	*a hamburger on the ground*
	`0.1`	*mike is holding the burger*
(prefer S0)	`0.2`	*the airplane is in the sky*

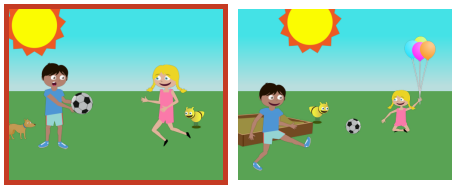
Figure 5: Captions for the same pair with varying λ. Changing λ alters both the naturalness and specificity of the output.



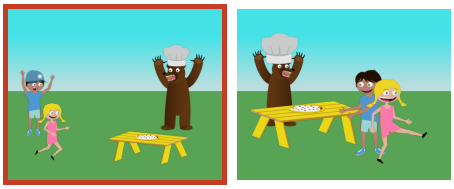
(a) *the sun is in the sky*
[contrastive]



(d) *the plane is flying in the sky*
[contrastive]



(c) *the dog is standing beside jenny*
[contrastive]



(b) *mike is wearing a chef's hat*
[non-contrastive]

Figure 4: Figure 4: Four randomly-chosen samples from our model. For each, the target image is shown on the left, the distractor image is shown on the right, and description generated by the model is shown below. All descriptions are fluent, and generally succeed in uniquely identifying the target scene, even when they do not perfectly describe it (e.g. (c)). These samples are broadly representative of the model's performance (Table 2).

	Dev acc. (%)		Test acc. (%)	
Model	All	Hard	All	Hard
Literal (S0)	66	54	64	53
Contrastive	71	54	69	58
Reasoning (S1)	**83**	**73**	**81**	**68**

Table 2: Success rates at RG on abstract scenes. "Literal" is a captioning baseline corresponding to the base speaker S0. "Contrastive" is a reimplementation of the approach of Mao et al. (2015). "Reasoning" is the model from this paper. All differences between our model and baselines are significant ($p < 0.05$, Binomial).

	# of differences				
	1	2	3	4	Mean
Literal (S0)	50	66	70	78	66 (%)
Reasoning	64	86	88	94	83
Compiled (S1)	44	72	80	80	69

Table 3: Comparison of the "compiled" pragmatic speaker model with literal and explicitly reasoning speakers. The models are evaluated on subsets of the development set, arranged by difficulty: column headings indicate the number of differences between the target and distractor scenes.

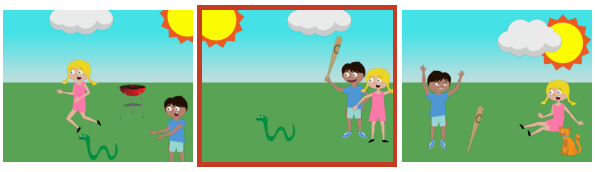
captions sampled from the reasoning speaker itself.

This model is evaluated in Table 3. While the distribution of scores is quite different from that of the base model (it improves noticeably over S0 on scenes with 2–3 differences), the overall gain is negligible (the difference in mean scores is not significant). The compiled model significantly underperforms the reasoning model. These results suggest either that the reasoning procedure is not easily approximated by a shallow neural network, or that example descriptions of randomly-sampled training pairs (which are usually easy to discriminate) do not provide a strong enough signal for a reflex learner to recover pragmatic behavior.

4.4 Final evaluation

Based on the following sections, we keep $\lambda = 0.02$ and use 100 samples to generate predictions. We evaluate on the test set, comparing this **Reasoning** model S1 to two baselines: **Literal**, an image captioning model trained normally on the abstract scene captions (corresponding to our L0), and **Contrastive**, a model trained with a soft contrastive objective, and previously used for visual referring expression generation (Mao et al., 2015).

Results are shown in Table 2. Our reasoning model outperforms both the literal baseline and previous work by a substantial margin, achieving an improvement of 17% on all pairs set and 15% on hard



(a) (b) (c)

(b vs. a) *mike is holding a baseball bat*
(b vs. c) *the snake is slithering away from mike and jenny*

Figure 6: Descriptions of the same image in different contexts. When the target scene (b) is contrasted with the left (a), the system describes a bat; when the target scene is contrasted with the right (c), the system describes a snake.

pairs.[2] Figures 4 and 6 show various representative descriptions from the model.

5 Conclusion

We have presented an approach for learning to generate pragmatic descriptions about general referents, even without training data collected in a pragmatic context. Our approach is built from a pair of simple neural base models, a listener and a speaker, and a high-level model that reasons about their outputs in order to produce pragmatic descriptions. In an evaluation on a standard referring expression game, our model's descriptions produced correct behavior in human listeners significantly more often than existing baselines.

It is generally true of existing derived approaches to pragmatics that much of the system's behavior requires hand-engineering, and generally true of direct approaches (and neural networks in particular) that training is only possible when supervision is available for the precise target task. By synthesizing these two approaches, we address both problems, obtaining pragmatic behavior without domain knowledge and without targeted training data. We believe that this general strategy of using reasoning to obtain novel contextual behavior from neural decoding models might be more broadly applied.

[2] For comparison, a model with hand-engineered pragmatic behavior—trained using a feature representation with indicators on only those objects that appear in the target image but not the distractor—produces an accuracy of 78% and 69% on all and hard development pairs respectively. In addition to performing slightly worse than our reasoning model, this alternative approach relies on the structure of scene representations and cannot be applied to more general pragmatics tasks.

References

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC map task corpus. *Language and speech*, 34(4):351–366.

Luciana Benotti and David Traum. 2009. A computational account of comparative implicatures for a spoken dialogue agent. In *Proceedings of the Eighth International Conference on Computational Semantics*, pages 4–17. Proceedings of the Annual Meeting of the Association for Computational Linguistics.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Michael C Frank, Noah D Goodman, Peter Lai, and Joshua B Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the 31st annual conference of the cognitive science society*, pages 1228–1233.

Albert Gatt, Ielka Van Der Sluis, and Kees Van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 49–56. Proceedings of the Annual Meeting of the Association for Computational Linguistics.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing*, pages 410–419. Association for Computational Linguistics.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Herbert P Grice. 1970. Logic and conversation.

Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, Van Ess-Dykema, et al. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 88–95. IEEE.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 787–798.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *arXiv preprint arXiv:1511.02283*.

Will Monroe and Christopher Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*, Amsterdam, December. ILLC.

Luis Gilberto Mateos Ortiz, Clemens Wolff, and Mirella Lapata. 2015. Learning to interpret and describe abstract scenes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1505–1515.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Nathaniel J Smith, Noah Goodman, and Michael Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems*, pages 3039–3047.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Adam Vogel, Max Bodoia, Christopher Potts, and Daniel Jurafsky. 2013. Emergence of Gricean maxims from multi-agent decision theory. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1072–1081.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

C Zitnick and Devi Parikh. 2013. Bringing semantics into focus using visual abstraction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3009–3016.

C Lawrence Zitnick, Ramakrishna Vedantam, and Devi Parikh. 2014. Adopting abstract images for semantic scene understanding.

Generating Topical Poetry

Marjan Ghazvininejad[†], Xing Shi[†], Yejin Choi[‡], and Kevin Knight[†]
[†]Information Sciences Institute & Computer Science Department
University of Southern California
{ghazvini,xingshi,knight}@isi.edu
[‡]Computer Science & Engineering, University of Washington
yejin@cs.washington.edu

Abstract

We describe Hafez, a program that generates any number of distinct poems on a user-supplied topic. Poems obey rhythmic and rhyme constraints. We describe the poetry-generation algorithm, give experimental data concerning its parameters, and show its generality with respect to language and poetic form.

1 Introduction

Automatic algorithms are starting to generate interesting, creative text, as evidenced by recent distinguishability tests that ask whether a given story, poem, or song was written by a human or a computer.[1] In this paper, we describe Hafez, a program that generates any number of distinct poems on a user-supplied topic. Figure 1 shows an overview of the system, which sets out these tasks:

- Vocabulary. We select a specific, large vocabulary of words for use in our generator, and we compute stress patterns for each word.
- Related words. Given a user-supplied topic, we compute a large set of related words.
- Rhyme words. From the set of related words, we select pairs of rhyming words to end lines.
- Finite-state acceptor (FSA). We build an FSA with a path for every conceivable sequence of vocabulary words that obeys formal rhythm constraints, with chosen rhyme words in place.
- Path extraction. We select a fluent path through the FSA, using a recurrent neural network (RNN) for scoring.

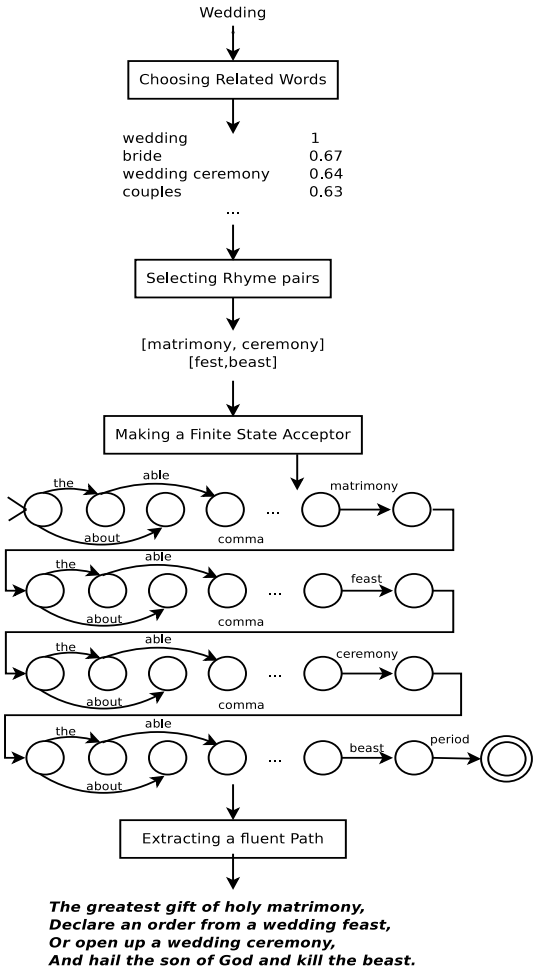


Figure 1: Overview of Hafez converting a user-supplied topic word (*wedding*) into a four-line iambic pentameter stanza.

[1]For example, in the 2016 Dartmouth test *bit.ly/20WGLF3*, no automatic sonnet-writing system passed indistinguishability, though ours was selected as the best of the submitted systems.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1183–1191,
Austin, Texas, November 1-5, 2016.

Sections 3-7 describe how we address these tasks. After this, we show results of Hafez generating 14-line classical sonnets with rhyme scheme ABAB CDCD EFEF GG, written in iambic pentameter (ten syllables per line with alternating stress: "da-DUM da-DUM da-DUM . . . "). We then show experiments on Hafez's parameters and conclude by showing the generality of the approach with respect to language and poetic form.

2 Prior Work

Automated poem generation has been a popular but challenging research topic (Manurung et al., 2000; Gervas, 2001; Diaz-Agudo et al., 2002; Manurung, 2003; Wong and Chun, 2008; Jiang and Zhou, 2008; Netzer et al., 2009). Recent work attempts to solve this problem by applying grammatical and semantic templates (Oliveira, 2009; Oliveira, 2012), or by modeling the task as statistical machine translation, in which each line is a "translation" of the previous line (Zhou et al., 2009; He et al., 2012). Yan et al. (2013) proposes a method based on summarization techniques for poem generation, retrieving candidate sentences from a large corpus of poems based on a user's query and clustering the constituent terms, summarizing each cluster into a line of a poem. Greene et al. (2010) use unsupervised learning to estimate the stress patterns of words in a poetry corpus, then use these in a finite-state network to generate short English love poems.

Several deep learning methods have recently been proposed for generating poems. Zhang and Lapata (2014) use an RNN model to generate 4-line Chinese poems. They force the decoder to rhyme the second and fourth lines, trusting the RNN to control rhythm. Yi et al. (2016) also propose an attention-based bidirectional RNN model for generating 4-line Chinese poems. The only such work which tries to generate longer poems is from Wang et al. (2016), who use an attention-based LSTM model for generation iambic poems. They train on a small dataset and do not use an explicit system for constraining rhythm and rhyme in the poem.

Novel contributions of our work are:

- We combine finite-state machinery with deep learning, guaranteeing formal correctness of our poems, while gaining coherence of long-distance RNNs.
- By using words related to the user's topic as rhyme words, we design a system that can generate poems with topical coherence. This allows us to generate longer topical poems.
- We extend our method to other poetry formats and languages.

3 Vocabulary

To generate a line of iambic pentameter poetry, we arrange words to form a sequence of ten syllables alternating between stressed and unstressed. For example:

```
   010     1  0    10      101
Attending on his golden pilgramage
```

Following Ghazvininejad and Knight (2015), we refer to unstressed syllables with 0 and stressed syllables with 1, so that the form of a Shakespearean sonnet is $((01)^5)^{14}$. To get stress patterns for individual words, we use CMU pronunciation dictionary,[2] collapsing primary and secondary stresses. For example:

```
CAFETERIA  K AE2 F AH0 T IH1 R IY0 AH0
```

becomes

```
CAFETERIA 10100
```

The first two columns of Table 1 show other examples. From the 125,074 CMU dictionary word types, we can actually only use words whose stress pattern matches the iambic pattern (alternating 1s and 0s). However, we make an exception for words that end in ...100 (such as *spatula*). To mimic how human poets employ such words, we convert all "...100" patterns to "...101". This leaves us with a 106,019 word types.

Words with multiple syllable-stress patterns present a challenge. For example, our program may use the word *record* in a "...10..." context, but if it is a verb in that context, a human reader will pronounce it as "01", breaking the intended rhythm. To guarantee that our poems scan properly, we eject all ambiguous words from our vocabulary. This problem is especially acute with monosyllabic words, as most have a stress that depends on context. Greene et al. (2010) apply the EM algorithm to align

[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

word	stress pattern	strict rhyme class	slant rhyme class (coarse version)
needing	10	IY1 D IH0 NG	IY1 * IH0 NG
ordinary	1010	EH1 R IY0	EH1 * IY0
obligate	101	EY1 T	last syllable stressed, no slant rhyme

Table 1: Sample word analyses.

human-written sonnets with assumed meter, extracting P(0|word) and P(1|word) probabilities. Using their method, we eject all monosyllabic words except those with P(0|word) > 0.9 or P(1|word) > 0.9. A consequence is that our poetry generator avoids the words *to*, *it*, *in*, and *is*, which actually forces the system into novel territory. This yields 16,139 monosyllabic and 87,282 multisyllabic words.

Because our fluency module (Section 7) is restricted to 20,000 word types, we further pare down our vocabulary by removing words that are not found in the 20k-most-frequent list derived from the song lyrics corpus we use for fluency. After this step, our final vocabulary contains 14,368 words (4833 monosyllabic and 9535 multisyllabic).

4 Topically Related Words and Phrases

After we receive a user-supplied topic, the first step in our poem generation algorithm is to build a scored list of 1000 words/phrases that are related to that topic. For example:

- **User-supplied input topic:** *colonel*
- **Output:** *colonel* (1.00), *lieutenant_colonel* (0.77), *brigadier_general* (0.73), *commander* (0.67) ... *army* (0.55) ...

This problem is different from finding synonyms or hypernyms in WordNet (Miller, 1995). For example, while Banerjee and Pedersen (2003) use WordNet to assign a 1.0 similarity score between *car* and *automobile*, they only give a 0.3 similarity between *car* and *gasoline*.

A second method is to use pointwise mutual information (PMI). Let t be the topic/phrase, and let w be a candidate related word. We collect a set of sentences S that contain t, and sort candidates by

$$\frac{\text{Proportion of sentences in } S \text{ containing } w}{\text{P}(w) \text{ in general text}}$$

Table 2 shows that PMI has a tendency to assign a high score to low frequency words (Bouma, 2009; Role and Nadif, 2011; Damani, 2013).

A third method is word2vec (Mikolov et al., 2013a), which provides distributed word representations. We train a continuous-bag-of-words model[3] with window size 8 and 40 and word vector dimension 200. We score candidate related words/phrases with cosine to topic-word vector. We find that a larger window size works best (Pennington et al., 2014; Levy and Goldberg, 2014).

Table 2 shows examples. The training corpus for word2vec has a crucial effect on the quality of the related words. We train word2vec models on the English Gigaword corpus,[4] a song lyrics corpus, and the first billion characters from Wikipedia.[5] The Gigaword corpus produces related words that are too newsy, while the song lyrics corpus does not cover enough topics. Hence, we train on Wikipedia. To obtain related phrases as well as words, we apply the method of Mikolov et al. (2013b) to the Wikipedia corpus, which replaces collocations like *Los Angeles* with single tokens like *Los_Angeles*. Word2vec then builds vectors for phrases as well as words. When the user supplies a multi-word topic, we use its phrase vector if available. Otherwise, we create the vector topic by element wise addition of its words' vectors.

5 Choosing Rhyme Words

We next fill in the right-hand edge of our poem by selecting pairs of rhyming words/phrases and assigning them to lines. In a Shakespearean sonnet with rhyme scheme ABAB CDCD EFEF GG, there are seven pairs of rhyme words to decide on.

5.1 Strict Rhyme

The strict definition of English rhyme is that the sounds of two words must match from the last stressed vowel onwards. In a masculine rhyme,

[3] https://code.google.com/archive/p/word2vec/
[4] https://catalog.ldc.upenn.edu/LDC2011T07
[5] http://mattmahoney.net/dc/enwik9.zip

Method	Window	Corpus	Phrases?	Related words
PMI	n/a	Gigaword	no	*croquet, Romai, Carisbo, NTTF, showcourts ...*
CBOW	8	Gigaword	no	*squash, badminton, golf, soccer, racquetball ...*
CBOW	40	Gigaword	no	*singles, badminton, squash, ATP, WTA ...*
CBOW	40	Song Lyrics	no	*high-heel, Reebok, steel-toed, basketball, Polos ...*
CBOW	40	Wikipedia	no	*volleyball, racquet, Wimbledon, athletics, doubles ...*
CBOW	40	Wikipedia	yes	*singles titles, grass courts, tennis club, hardcourt ...*

Table 2: Different methods for extracting words related to the topic *tennis*.

the last syllable is stressed; in a feminine rhyme, the penultimate syllable is stressed. We collect phoneme and stress information from the CMU pronunciation dictionary. We pre-compute strict rhyme classes for words (see Table 1) and hash the vocabulary into those classes.

5.2 Slant Rhyme

In practice, human poets do not always use strict rhymes. To give ourselves more flexibility in choosing rhyme pairs, we allow for slant (or half) rhymes. By inspecting human rhyming data, we develop this operational definition of slant rhyme:

1. Let s1 and s2 be two potentially-rhyming phoneme sequences.
2. Replace ER with UH R in both sequences.
3. Let v1 and v2 be the last stressed vowels in s1 and s2.
4. Let w1 and w2 be last vowels in s1 and s2.
5. Let s1 = a1 v1 x1 w1 c1. Likewise, let s2 = a2 v2 x2 w2 c2.
6. Output NO under any of these circumstances: (a) v1 $\neq$ v2, (b) w1 $\neq$ w2, (c) c1 $\neq$ c2, (d) a1 $\neq$ NULL and a2 $\neq$ NULL and a1 = a2.
7. If x1 and x2 are single phonemes:
   (a) If x1 $\sim$ x2, then output YES.[6]
   (b) Otherwise, output NO.
8. If x1 and x2 contain different numbers of vowels, output NO.
9. Let p1 and q1 be the first and last phonemes of x1. Let p2 and q2 be the same for x2.
10. If (p1 = p2) and (q1 $\sim$ q2), output YES.
11. If (p1 $\sim$ p2) and (q1 = q1), output YES.
12. Otherwise, output NO.

[6] x $\sim$ y if phonemes x and y are similar. Two phonemes are similar if their pairwise score according to (Hirjee and Brown, 2010) is greater than -0.6. This includes 98 pairs, such as L/R, S/SH, and OY/UH.

Words whose last syllable is stressed do not participate in slant rhymes.

Example slant rhymes taken from our generated poems include *Viking/fighting*, *snoopy/spooky*, *baby/crazy* and *comic/ironic*. We pre-compute a coarse version of slant rhyme classes (Table 1) with the pattern "v_i * w_i c_i". If two words hash to the same coarse class, then we subsequently accept or reject depending on the similarity of the intermediate phonemes.

5.3 Non-Topical Rhyming Words

For rare topics, we may not have enough related words to locate seven rhyming pairs. For example, we generate 1000 related words for the topic *Viking*, but only 32 of them are found in our 14,368-word vocabulary. To give a chance for all topical words/phrases to be used as rhyme words, for each strict rhyme class, we add the most common word in our song lyric corpus to the list of related words. In addition, we add words from popular rhyme pairs[7] (like *do/you* and *go/know*) to the list of related words with a low topic similarity score.

5.4 Rhyme word selection

We first hash all related words/phrases into rhyme classes. Each collision generates a candidate rhyme pair (s1, s2), which we score with the maximum of cosine(s1, topic) and cosine(s2, topic). So that we can generate many different sonnets on the same topic, we choose rhyme pairs randomly with probability proportional to their score. After choosing a pair (s1, s2), we remove it, along with any other candidate pair that contains s1 or s2. Because a poem's beginning and ending are more important, we assign the first rhyme pair to the last two lines of the sonnet,

[7] http://slate.me/OhTKCA

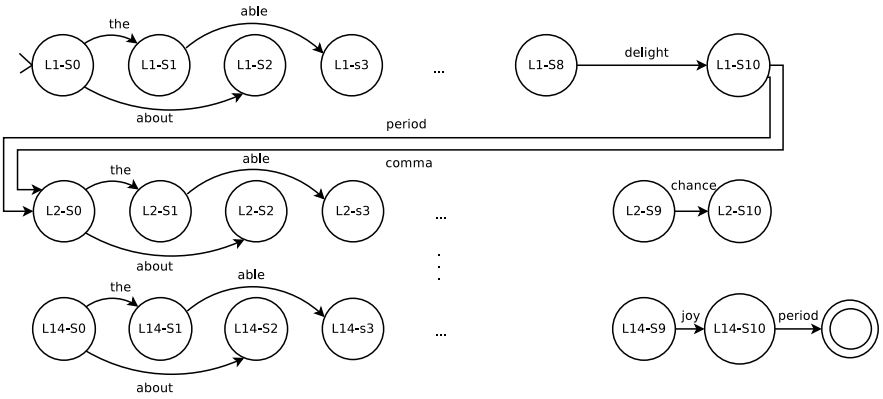


Figure 2: An FSA compactly encoding all word sequences that obey formal sonnet constraints, and dictating the right-hand edge of the poem via rhyming, topical words *delight*, *chance*, ... and *joy*.

then assign other pairs from beginning of the sonnet towards the end.

6 Constructing FSA of Possible Poems

After choosing rhyme words, we create a large finite-state acceptor (FSA) that compactly encodes all word sequences that use these rhyme words and also obey formal sonnet constraints:

- Each sonnet contains 14 lines.
- Lines are in iambic pentameter, with stress pattern $(01)^5$. Following poetic convention, we also use $(01)^50$, allowing feminine rhyming.
- Each line ends with the chosen rhyme word/phrase for that line.
- Each line is punctuated with comma or period, except for the 4th, 8th, 12th, and 14th lines, which are punctuated with period.

To implement these constraints, we create FSA states that record line number and syllable count. For example, FSA state L2-S3 (Figure 2) signifies "I am in line 2, and I have seen 3 syllables so far". From each state, we create arcs for each feasible word in the vocabulary. For example, we can move from state L1-S1 to state L1-S3 by consuming any word with stress pattern 10 (such as *table* or *active*). When moving between lines (e.g., from L1-S10 to L2-S1), we employ arcs labeled with punctuation marks.

To fix the rhyme words at the end of each line, we delete all arcs pointing to the line-final state, except for the arc labeled with the chosen rhyme word. For speed, we pre-compute the entire FSA; once we receive the topic and choose rhyme words, we only need to carry out the deletion step.

In the resulting FSA, each path is *formally* a sonnet. However, most of the paths through the FSA are meaningless. One FSA generated from the topic *natural language* contains 10^{229} paths, including this randomly-selected one:

Of pocket solace ammunition grammar.
An tile pretenders spreading logical.
An stories Jackie gallon posing banner.
An corpses Kato biological ...

Hence, we need a way to search and rank this large space.

7 Path extraction through FSA with RNN

To locate fluent paths, we need a scoring function and a search procedure. For example, we can build a n-gram word language model (LM)—itself a large weighted FSA. Then we can take a weighted intersection of our two FSAs and return the highest-scoring path. While this can be done efficiently with dynamic programming, we find that n-gram models have a limited attention span, yielding poor poetry.

Instead, we use an RNN language model (LM). We collect 94,882 English songs (32m word tokens) as our training corpus,[8] and train[9] a two-layer recurrent network with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997).[10]

When decoding with the LM, we employ a beam

[8] http://www.mldb.org/

[9] We use the toolkit: https://github.com/isi-nlp/Zoph_RNN

[10] We use a minibatch of 128, a hidden state size of 1000, and a dropout rate of 0.2. The output vocabulary size is 20,000. The learning rate is initially set as 0.7 and starts to decay by 0.83 once the perplexity on a development set starts to increase. All parameters are initialized within range $[-0.08, +0.08]$, and the gradients are re-scaled when the global norm is larger than 5.

search that is further guided by the FSA. Each beam state $C_{t,i}$ is a tuple of $(h, s, word, score)$, where h is the hidden states of LSTM at step t in ith state, and s is the FSA state at step t in ith state. The model generates one word at each step.

At the beginning, $h_{0,0}$ is the initial hidden state of LSTM, $s_{0,0}$ is the start state of FSA, $word_{0,0}$ = <START> and $score_{0,0} = 0$. To expand a beam state $C_{t,i}$, we first feed $h_{t,i}$ and $word$ into the LM and get an updated hidden state h_{next}. The LM also returns a probability distribution $P(V)$ over the entire vocabulary V for next word. Then, for each succeeding state s_{suc} of $s_{t,i}$ in the FSA and the word w_{next} over each edge from $s_{t,i}$ to s_{suc}, we form a new state $(h_{next}, s_{suc}, w_{next}, score_{t,i} + log(P(w_{next})))$ and push it into next beam.

Because we fix the rhyme word at the end of each line, when we expand the beam states immediately before the rhyme word, the FSA states in those beam states have only one succeeding state—LN-S10, where $N = [1, 14]$, and only one succeeding word, the fixed rhyme word. For our beam size b = 50, the chance is quite low that in those b words there exists any suitable word to precede that rhyme word. We solve this by generating the whole sonnet *in reverse*, starting from the final rhyme word. Thus, when we expand the state L1-S8, we can choose from almost every word in vocabulary instead of just b possible words. The price to pay is that at the beginning of each line, we need to hope in those b words there exists some that are suitable to succeed comma or period.

Because we train on song lyrics, our LM tends to generate repeating words, like *never ever ever ever ever*. To solve this problem, we apply a penalty to those words that already generated in previous steps during the beam search.

To create a poem that fits well with the predetermined rhyme words at the end of each line, the LM model tends to choose "safe" words that are frequent and suitable for any topic, such as pronouns, adverbs, and articles. During decoding, we apply a reward on all topically related words (generated in Section 4) in the non-rhyming portion of the poem.

Finally, to further encourage the system to follow the topic, we train an encoder-decoder sequence-to-sequence model (Sutskever et al., 2014). For training, we select song lyric rhyme words and assemble them in reverse order (encoder side), and we pair this with the entire reversed lyric (decoder side). At generation time, we put all the selected rhyme words on the source side, and let the model to generate the poem conditioned on those rhyme words. In this way, when the model tries to generate the last line of the poem, it already knows all fourteen rhyme words, thus possessing better knowledge of the requested topic. We refer to generating poems using the RNN LM as the "generation model" and to this model as the "translation model".

Bipolar Disorder

Existence enters your entire nation.
A twisted mind reveals becoming manic,
An endless modern ending medication,
Another rotten soul becomes dynamic.

Or under pressure on genetic tests.
Surrounded by controlling my depression,
And only human torture never rests,
Or maybe you expect an easy lesson.

Or something from the cancer heart disease,
And I consider you a friend of mine.
Without a little sign of judgement please,
Deliver me across the borderline.

An altered state of manic episodes,
A journey through the long and winding roads.

Figure 3: Sample sonnet generated from the topic phrase *bipolar disorder*.

8 Results and Analysis

Sample outputs produced by our best system are shown in Figures 3 and 4. We find that they generally stay on topic and are fairly creative. If we request a poem on the topic *Vietnam*, we may see the phrase *Honky Tonkin Resolution*; a different topic leads the system to rhyme *Dirty Harry* with *Bloody Mary*. In this section, we present experiments we used to select among different versions of our poem generator.

The first experiment tests the effect of encouraging topical words in the body of the poem, via a direct per-word bonus. For 40 different topics, we generate 2 sonnets with and without encouragement, using the same set of rhyme words. Then we ask 23 human judges to choose the better sonnet. Each judge compares sonnets for 10 different topics. Table 3 shows that using topical words increases the

Love at First Sight

An early morning on a rainy night,
Relax and make the other people happy,
Or maybe get a little out of sight,
And wander down the streets of Cincinnati.

Girlfriend

Another party started getting heavy.
And never had a little bit of Bobby,
Or something going by the name of Eddie,
And got a finger on the trigger sloppy.

Noodles

The people wanna drink spaghetti alla,
And maybe eat a lot of other crackers,
Or sit around and talk about the salsa,
A little bit of nothing really matters.

Civil War

Creating new entire revolution,
An endless nation on eternal war,
United as a peaceful resolution,
Or not exist together any more.

Figure 4: Sample stanzas generated from different topic phrases.

Preference	Encourages	Does Not Encourage	Cannot Decide
Sonnets	**54**%	18%	28%

Table 3: Users prefer the system that encourages the use of related words in the body (non-rhyme) portion of the poem. 40 poems are tested with 23 judges.

quality of the sonnets.

Next, we compare the translation model with generation model. For each of 40 topics, we generate one poem with generation model and one poem with translation model, using the same set of rhyme words. We ask 25 human judges to chose the better poem. Each judge compares sonnets for 10 different topics. This experiment is run separately for sonnets and stanzas. Table 4 shows how the translation model generates better poems, and Figure 5 compares two stanzas.

We check for plagiarism, as it is common for optimal-searching RNNs to repeat large sections of the training data. We hypothesize that strong conditions on rhyme, meter, repetition, and ambiguously-stressed words will all mitigate against plagiarism.

Gen	Another tiny thousand ashes scattered. And never hardly ever really own, Or many others have already gathered, The only human being left alone.
Trans	Being **buried** under ashes scattered, Many faces we forgotten own, About a hundred thousand **soldiers** gathered, And I remember standing all alone.

Figure 5: Stanzas generated with and without a encoder-decoder translation model for topic *death*.

Preference	Generation Model	Translation Model	Cannot Decide
Stanzas	26%	**43**%	31%
Sonnets	21%	**57**%	22%

Table 4: Users prefer poems created with the encoder-decoder translation model over those that use only the RNN language model in generation mode. 40 poems are tested with 25 judges.

We find that on average, each sonnet copies only 1.2 5-grams from the training data. If we relax the repeated-word penalty and the iambic meter, this number increases to 7.9 and 10.6 copied 5-grams, respectively. Considering the lack of copying, we find the RNN-generated grammar to be quite good. The most serious—and surprisingly common—grammatical error is the wrong use of *a* and *an*, which we fix in a post-processing step.

9 Other Languages and Formats

To show the generality of our approach, we modify our system to generate Spanish-language poetry from a Spanish topic. We use these resources:

- A song lyric corpus for training our RNN. We download 97,775 Spanish song lyrics from LyricWikia,[11] which amounts to 20m word tokens and 219k word types.
- A Spanish Wikipedia dump[12] consisting of 885m word tokens, on which we run word2vec to find words and phrases related to the topic.

Our vocabulary consists of the 20k most frequent lyric words. For each word, we compute its syllable-stress pattern and its rhyme class (see Figure 6). Because Spanish writing is quite phonetic, we can retrieve this from the letter strings of the vocabulary.

[11] http://lyrics.wikia.com/wiki/Category:Language/Spanish

[12] https://dumps.wikimedia.org/eswiki/20160305/eswiki-20160305-pages-meta-current.xml.bz2

word	stress	rhyme	v-	-v
consultado	0010	-ado		yes
aduciendo	0010	-endo	yes	yes
régimen	100	-egimen		
hospital	001	-al	yes	

Figure 6: Sample word analyses needed to construct Spanish Hafez. v- and -v indicate whether the word starts and/or ends with a vowel sound.

For any given vocabulary word:[13]

1. We remove silent h, and convert y into i.
2. We count the number of syllables by isolating vowel groups. In such groups, weak vowels (i, u) attached to strong vowels (a, e, o) do not form separate syllables, unless they are accented (*dí-as* versus *dios*). Strong clusters are broken into separate syllables (eg, *ca-er*).
3. We determine which vowel (and therefore syllable) is stressed. If any vowel is accented, it is stressed. If the word is accent-free, then the second-to-last syllable is stressed, unless the word ends in a consonant other than n or s, in which case the last syllable is stressed.
4. We form the word's rhyme class by breaking off a letter suffix starting at the last stressed vowel (as in English). Weak vowels do not participate (e.g., *tienda* $\rightarrow$ *-enda*, not *-ienda*). We remove h from the rhyme, so *búho* rhymes with *continúo*. Because rhyming is easier in Spanish than English, we do not need slant rhyme.

Most Spanish poetic formats enforce some number of syllables per line, without meter. However, there are two caveats when counting syllables:

1. *Sinalefa* merges vowels across word boundaries. Thus, *la_obra* is counted as two syllables instead of three, and *va_a_hacer* is counted as two syllables instead of four. A line may therefore have more words than syllables.
2. For the last word of a line (only), we count up to its last stressed syllable, then add one. This means that even though iambic meter is not employed, we still need stress patterns to correctly count syllables.

We implement these constraints in the FSA framework, now with separate states for "I have seen M syllables, and the last word ended in a vowel sound" and "I have seen M syllables, and the last word ended in a consonant sound." Technically speaking, the FSA includes single-state cycles for the Spanish word a, due to *sinalefa*. Line-ending states can only be reached by words that have their syllable count adjusted as in point 2 above.

[13] http://community.dur.ac.uk/m.p.thompson/verse.htm

Mariposa

Quieres saber dónde está el escorpión,
Ni ayer ni antes vos sos corona dorada.
Ya os ves más tal cual tortuga pintada,
A él nos gusta andar con cola marrón.

Ella es quién son las alas de algún gorrión.
Si al fin podés ver tu imagen manchada,
O hoy vas bajo un cielo azul plateada,
Por qué estás tan lejos del aguijón.

No hay luz que al sol se enreda en tus palmera.
Ay por qué eres víbora venenosa,
Sin querer igual a un enredadera.

Y si aún sueñas con ser mariposa,
En vez de abrir los ojos y espera,
Sabes muy bien que el amor no es gran cosa.

Figure 7: Sample Spanish poem generated in classical *soneta* form, on the topic *mariposa* (butterfly).

Figure 7 shows a sample Spanish output. The format is the classical Spanish *soneta*, which consists of 14 eleven-syllable lines under the rhyme scheme ABBA ABBA CDC DCD. This scheme requires us to choose up to four words with the same rhyme.

Overall, we also find Spanish outputs to be fluent, fairly creative, and on topic. Grammatical problems are a bit more common than in our English generator—for example, adjacent words sometimes disagree in number or gender. The RNN generalizations that permit these errors no doubt also permit creative phrasings.

10 Conclusion

We have described Hafez, a poetry generation system that combines hard format constraints with a deep-learning recurrent network. The system uses special techniques, such as rhyme-word choice and encoder-decoder modeling, to keep the poem on topic. We hope that future work will provide more discourse structure and function to automatic poetry, while maintaining the syntax, semantics, and creative phrasing we observe.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by DARPA (W911NF-15-1-0543) and NSF (IIS-1524371).

References

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proc. IJCAI*.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proc. Biennial GSCL Conference*.

Om P. Damani. 2013. Improving pointwise mutual information (PMI) by incorporating significant co-occurrence. In *Proc. ACL*.

Belen Diaz-Agudo, Pablo Gervas, and Pedro Gonzalez-Calero. 2002. Poetry generation in COLIBRI. In *Proc. ECCBR*.

Pablo Gervas. 2001. An expert system for the composition of formal Spanish poetry. *Knowledge-Based Systems*, 14(3).

Marjan Ghazvininejad and Kevin Knight. 2015. How to memorize a random 60-bit string. In *Proc. NAACL*.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proc. EMNLP*.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese classical poems with statistical machine translation models. In *Proc. AAAI*.

Hussein Hirjee and Daniel Brown. 2010. Using automated rhyme detection to characterize rhyming style in rap music. In *Empirical Musicology Review*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8).

Long Jiang and Ming Zhou. 2008. Generating Chinese couplets using a statistical MT approach. In *Proc. COLING*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. ACL*.

Hisar Manurung, Graeme Ritchie, and Henry Thompson. 2000. Towards a computational model of poetry generation. In *Proc. AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

Hisar Manurung. 2003. An evolutionary algorithm approach to poetry generation. *Ph.D. thesis, University of Edinburgh.*

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*.

George Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In *Proc. NAACL Workshop on Computational Approaches to Linguistic Creativity*.

Hugo Oliveira. 2009. Automatic generation of poetry: an overview. In *Proc. 1st Seminar of Art, Music, Creativity and Artificial Intelligence*.

Hugo Oliveira. 2012. PoeTryMe: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1.

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

Franois Role and Mohamed Nadif. 2011. Handling the impact of low frequency events on co-occurrence based measures of word similarity—a case study of pointwise mutual information. In *Knowledge Discovery and Information Retrieval*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. *arXiv:1604.06274*.

Martin Wong and Andy Chun. 2008. Automatic haiku generation using VSM. In *Proc. ACACOS*.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. I, Poet: Automatic Chinese poetry composition through a generative summarization framework under constrained optimization. In *Proc. IJCAI*.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating chinese classical poems with RNN encoder-decoder. *arXiv:1604.01537*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proc. EMNLP*.

Ming Zhou, Long Jiang, and Jing He. 2009. Generating Chinese couplets and quatrain using a statistical approach. In *Proc. Pacific Asia Conference on Language, Information and Computation*.

Deep Reinforcement Learning for Dialogue Generation

Jiwei Li[1], **Will Monroe**[1], **Alan Ritter**[2], **Michel Galley**[3], **Jianfeng Gao**[3] **and Dan Jurafsky**[1]
[1]Stanford University, Stanford, CA, USA
[2]Ohio State University, OH, USA
[3]Microsoft Research, Redmond, WA, USA
{jiweil,wmonroe4,jurafsky}@stanford.edu, ritter.1492@osu.edu
{mgalley,jfgao}@microsoft.com

Abstract

Recent neural models of dialogue generation offer great promise for generating responses for conversational agents, but tend to be shortsighted, predicting utterances one at a time while ignoring their influence on future outcomes. Modeling the future direction of a dialogue is crucial to generating coherent, interesting dialogues, a need which led traditional NLP models of dialogue to draw on reinforcement learning. In this paper, we show how to integrate these goals, applying deep reinforcement learning to model future reward in chatbot dialogue. The model simulates dialogues between two virtual agents, using policy gradient methods to reward sequences that display three useful conversational properties: informativity, coherence, and ease of answering (related to forward-looking function). We evaluate our model on diversity, length as well as with human judges, showing that the proposed algorithm generates more interactive responses and manages to foster a more sustained conversation in dialogue simulation. This work marks a first step towards learning a neural conversational model based on the long-term success of dialogues.

1 Introduction

Neural response generation (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016a; Wen et al., 2015; Yao et al., 2015; Luan et al., 2016; Xu et al., 2016; Wen et al., 2016; Li et al., 2016b; Su et al., 2016) is of growing interest. The LSTM sequence-to-sequence (SEQ2SEQ) model (Sutskever et al., 2014) is one type of neural generation model that maximizes the probability of generating a response given the previous dialogue turn. This approach enables the incorporation of rich context when mapping between consecutive dialogue turns (Sordoni et al., 2015) in a way not possible, for example, with MT-based dialogue models (Ritter et al., 2011).

Despite the success of SEQ2SEQ models in dialogue generation, two problems emerge: First, SEQ2SEQ models are trained by predicting the next dialogue turn in a given conversational context using the maximum-likelihood estimation (MLE) objective function. However, it is not clear how well MLE approximates the real-world goal of chatbot development: teaching a machine to converse with humans, while providing interesting, diverse, and informative feedback that keeps users engaged. One concrete example is that SEQ2SEQ models tend to generate highly generic responses such as "*I don't know*" regardless of the input (Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a). This can be ascribed to the high frequency of generic responses found in the training set and their compatibility with a diverse range of conversational contexts. Yet *"I don't know"* is apparently not a good action to take, since it closes the conversation down.

Another common problem, illustrated in the two sample conversations on the left of Table 1, is that the system becomes stuck in an infinite loop of repetitive responses. This is due to MLE-based SEQ2SEQ models' inability to account for repetition. In example 2 (bottom left), the dialogue falls into an infinite loop after three turns, with both agents generating dull, generic utterances like *i don't know what you are talking about* and *you don't know what you are saying*. Looking at the entire conversation, utterance (4) turns out to be a bad action to take because it offers no way of continuing the conversation.[1]

[1]A similar rule is often suggested in improvisational comedy: https://en.wikipedia.org/wiki/Yes,_and...

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1192–1202,
Austin, Texas, November 1-5, 2016.

Baseline mutual information model (Li et al. 2015)	Proposed reinforcement learning model
A: Where are you going? (1)	A: Where are you going? (1)
B: I'm going to the restroom. (2)	B: I'm going to the police station. (2)
A: See you later. (3)	A: I'll come with you. (3)
B: See you later. (4)	B: No, no, no, no, you're not going anywhere. (4)
A: See you later. (5)	A: Why? (5)
B: See you later. (6)	B: I need you to stay here. (6)
...	A: I don't know what you are talking about. (7)
...	...
A: how old are you? (1)	A: How old are you? (1)
B: I'm 16. (2)	B: I'm 16. Why are you asking? (2)
A: 16? (3)	A I thought you were 12. (3)
B: I don't know what you are talking about. (4)	B: What made you think so? (4)
A: You don't know what you are saying. (5)	A: I don't know what you are talking about. (5)
B: I don't know what you are talking about . (6)	B: You don't know what you are saying. (6)
A: You don't know what you are saying. (7)	...
...	...

Table 1: **Left Column**: Dialogue simulation between two agents using a 4-layer LSTM encoder-decoder trained on the OpenSubtitles dataset. The first turn (index 1) is input by the authors. Then the two agents take turns conversing, taking as input the other agent's prior generated turn. The output is generated using the mutual information model (Li et al., 2015) in which an N-best list is first obtained using beam search based on p(t|s) and reranked by linearly combining the backward probability p(s|t), where t and s respectively denote targets and sources. **Right Column**: Dialogue simulated using the proposed reinforcement learning model. The new model has more forward-looking utterances (questions like "Why are you asking?" and offers like "I'll come with you") and lasts longer before it falls into conversational black holes.

These challenges suggest we need a conversation framework that has the ability to (1) integrate developer-defined rewards that better mimic the true goal of chatbot development and (2) model the long-term influence of a generated response in an ongoing dialogue.

To achieve these goals, we draw on the insights of reinforcement learning, which have been widely applied in MDP and POMDP dialogue systems (see Related Work section for details). We introduce a neural reinforcement learning (RL) generation method, which can optimize long-term rewards designed by system developers. Our model uses the encoder-decoder architecture as its backbone, and simulates conversation between two virtual agents to explore the space of possible actions while learning to maximize expected reward. We define simple heuristic approximations to rewards that characterize good conversations: good conversations are forward-looking (Allwood et al., 1992) or interactive (a turn suggests a following turn), informative, and coherent. The parameters of an encoder-decoder RNN define a policy over an infinite action space consisting of all possible utterances. The agent learns a policy by optimizing the long-term developer-defined reward from ongoing dialogue simulations using policy gradient methods (Williams, 1992), rather than the MLE objective defined in standard SEQ2SEQ models.

Our model thus integrates the power of SEQ2SEQ systems to learn compositional semantic meanings of utterances with the strengths of reinforcement learning in optimizing for long-term goals across a conversation. Experimental results (sampled results at the right panel of Table 1) demonstrate that our approach fosters a more sustained dialogue and manages to produce more interactive responses than standard SEQ2SEQ models trained using the MLE objective.

2 Related Work

Efforts to build statistical dialog systems fall into two major categories.

The first treats dialogue generation as a source-to-target transduction problem and learns mapping rules between input messages and responses from a massive amount of training data. Ritter et al. (2011) frames the response generation problem as a statisti-

cal machine translation (SMT) problem. Sordoni et al. (2015) improved Ritter et al.'s system by rescoring the outputs of a phrasal SMT-based conversation system with a neural model that incorporates prior context. Recent progress in SEQ2SEQ models inspire several efforts (Vinyals and Le, 2015) to build end-to-end conversational systems which first apply an encoder to map a message to a distributed vector representing its semantics and generate a response from the message vector. Serban et al. (2016) propose a hierarchical neural model that captures dependencies over an extended conversation history. Li et al. (2016a) propose mutual information between message and response as an alternative objective function in order to reduce the proportion of generic responses produced by SEQ2SEQ systems.

The other line of statistical research focuses on building task-oriented dialogue systems to solve domain-specific tasks. Efforts include statistical models such as Markov Decision Processes (MDPs) (Levin et al., 1997; Levin et al., 2000; Walker et al., 2003; Pieraccini et al., 2009), POMDP (Young et al., 2010; Young et al., 2013; Gašic et al., 2013a; Gašic et al., 2014) models, and models that statistically learn generation rules (Oh and Rudnicky, 2000; Ratnaparkhi, 2002; Banchs and Li, 2012; Nio et al., 2014). This dialogue literature thus widely applies reinforcement learning (Walker, 2000; Schatzmann et al., 2006; Gasic et al., 2013b; Singh et al., 1999; Singh et al., 2000; Singh et al., 2002) to train dialogue policies. But task-oriented RL dialogue systems often rely on carefully limited dialogue parameters, or hand-built templates with state, action and reward signals designed by humans for each new domain, making the paradigm difficult to extend to open-domain scenarios.

Also relevant is prior work on reinforcement learning for language understanding - including learning from delayed reward signals by playing text-based games (Narasimhan et al., 2015; He et al., 2016), executing instructions for Windows help (Branavan et al., 2011), or understanding dialogues that give navigation directions (Vogel and Jurafsky, 2010).

Our goal is to integrate the SEQ2SEQ and reinforcement learning paradigms, drawing on the advantages of both. We are thus particularly inspired by recent work that attempts to merge these paradigms, including Wen et al. (2016)— training an end-to-end task-oriented dialogue system that links input representations to slot-value pairs in a database— or Su et al. (2016), who combine reinforcement learning with neural generation on tasks with real users, showing that reinforcement learning improves dialogue performance.

3 Reinforcement Learning for Open-Domain Dialogue

In this section, we describe in detail the components of the proposed RL model.

The learning system consists of two agents. We use p to denote sentences generated from the first agent and q to denote sentences from the second. The two agents take turns talking with each other. A dialogue can be represented as an alternating sequence of sentences generated by the two agents: $p_1, q_1, p_2, q_2, ..., p_i, q_i$. We view the generated sentences as actions that are taken according to a policy defined by an encoder-decoder recurrent neural network language model.

The parameters of the network are optimized to maximize the expected future reward using policy search, as described in Section 4.3. Policy gradient methods are more appropriate for our scenario than Q-learning (Mnih et al., 2013), because we can initialize the encoder-decoder RNN using MLE parameters that already produce plausible responses, before changing the objective and tuning towards a policy that maximizes long-term reward. Q-learning, on the other hand, directly estimates the future expected reward of each action, which can differ from the MLE objective by orders of magnitude, thus making MLE parameters inappropriate for initialization. The components (states, actions, reward, etc.) of our sequential decision problem are summarized in the following sub-sections.

3.1 Action

An action a is the dialogue utterance to generate. The action space is infinite since arbitrary-length sequences can be generated.

3.2 State

A state is denoted by the previous two dialogue turns $[p_i, q_i]$. The dialogue history is further transformed to a vector representation by feeding the concatenation of p_i and q_i into an LSTM encoder model as

described in Li et al. (2016a).

3.3 Policy

A policy takes the form of an LSTM encoder-decoder (i.e., $p_{RL}(p_{i+1}|p_i, q_i)$) and is defined by its parameters. Note that we use a stochastic representation of the policy (a probability distribution over actions given states). A deterministic policy would result in a discontinuous objective that is difficult to optimize using gradient-based methods.

3.4 Reward

r denotes the reward obtained for each action. In this subsection, we discuss major factors that contribute to the success of a dialogue and describe how approximations to these factors can be operationalized in computable reward functions.

Ease of answering A turn generated by a machine should be easy to respond to. This aspect of a turn is related to its *forward-looking function*: the constraints a turn places on the next turn (Schegloff and Sacks, 1973; Allwood et al., 1992). We propose to measure the ease of answering a generated turn by using the negative log likelihood of responding to that utterance with a dull response. We manually constructed a list of dull responses $\mathbb{S}$ consisting 8 turns such as "I don't know what you are talking about", "I have no idea", etc., that we and others have found occur very frequently in SEQ2SEQ models of conversations. The reward function is given as follows:

$$r_1 = -\frac{1}{N_{\mathbb{S}}} \sum_{s \in \mathbb{S}} \frac{1}{N_s} \log p_{\text{seq2seq}}(s|a) \qquad (1)$$

where $N_{\mathbb{S}}$ denotes the cardinality of $N_{\mathbb{S}}$ and N_s denotes the number of tokens in the dull response s. Although of course there are more ways to generate dull responses than the list can cover, many of these responses are likely to fall into similar regions in the vector space computed by the model. A system less likely to generate utterances in the list is thus also less likely to generate other dull responses.

p_{seq2seq} represents the likelihood output by SEQ2SEQ models. It is worth noting that p_{seq2seq} is different from the stochastic policy function $p_{RL}(p_{i+1}|p_i, q_i)$, since the former is learned based on the MLE objective of the SEQ2SEQ model while the latter is the policy optimized for long-term future reward in the RL setting. r_1 is further scaled by the length of target $\mathbb{S}$.

Information Flow We want each agent to contribute new information at each turn to keep the dialogue moving and avoid repetitive sequences. We therefore propose penalizing semantic similarity between consecutive turns from the same agent. Let h_{p_i} and $h_{p_{i+1}}$ denote representations obtained from the encoder for two consecutive turns p_i and p_{i+1}. The reward is given by the negative log of the cosine similarity between them:

$$r_2 = -\log \cos(h_{p_i}, h_{p_{i+1}}) = -\log \frac{h_{p_i} \cdot h_{p_{i+1}}}{\|h_{p_i}\| \|h_{p_{i+1}}\|} \qquad (2)$$

Semantic Coherence We also need to measure the adequacy of responses to avoid situations in which the generated replies are highly rewarded but are ungrammatical or not coherent. We therefore consider the mutual information between the action a and previous turns in the history to ensure the generated responses are coherent and appropriate:

$$r_3 = \frac{1}{N_a} \log p_{\text{seq2seq}}(a|q_i, p_i) + \frac{1}{N_{q_i}} \log p^{\text{backward}}_{\text{seq2seq}}(q_i|a) \qquad (3)$$

$p_{\text{seq2seq}}(a|p_i, q_i)$ denotes the probability of generating response a given the previous dialogue utterances $[p_i, q_i]$. $p^{\text{backward}}_{\text{seq2seq}}(q_i|a)$ denotes the backward probability of generating the previous dialogue utterance q_i based on response a. $p^{\text{backward}}_{\text{seq2seq}}$ is trained in a similar way as standard SEQ2SEQ models with sources and targets swapped. Again, to control the influence of target length, both $\log p_{\text{seq2seq}}(a|q_i, p_i)$ and $\log p^{\text{backward}}_{\text{seq2seq}}(q_i|a)$ are scaled by the length of targets.

The final reward for action a is a weighted sum of the rewards discussed above:

$$r(a, [p_i, q_i]) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 \qquad (4)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$. We set $\lambda_1 = 0.25$, $\lambda_2 = 0.25$ and $\lambda_3 = 0.5$. A reward is observed after the agent reaches the end of each sentence.

4 Simulation

The central idea behind our approach is to simulate the process of two virtual agents taking turns talking with each other, through which we can explore the

state-action space and learn a policy $p_{RL}(p_{i+1}|p_i, q_i)$ that leads to the optimal expected reward. We adopt an AlphaGo-style strategy (Silver et al., 2016) by initializing the RL system using a general response generation policy which is learned from a fully supervised setting.

4.1 Supervised Learning

For the first stage of training, we build on prior work of predicting a generated target sequence given dialogue history using the supervised SEQ2SEQ model (Vinyals and Le, 2015). Results from supervised models will be later used for initialization.

We trained a SEQ2SEQ model with attention (Bahdanau et al., 2015) on the OpenSubtitles dataset, which consists of roughly 80 million source-target pairs. We treated each turn in the dataset as a target and the concatenation of two previous sentences as source inputs.

4.2 Mutual Information

Samples from SEQ2SEQ models are often times dull and generic, e.g., "*i don't know*" (Li et al., 2016a) We thus do not want to initialize the policy model using the pre-trained SEQ2SEQ models because this will lead to a lack of diversity in the RL models' experiences. Li et al. (2016a) showed that modeling mutual information between sources and targets will significantly decrease the chance of generating dull responses and improve general response quality. We now show how we can obtain an encoder-decoder model which generates maximum mutual information responses.

As illustrated in Li et al. (2016a), direct decoding from Eq 3 is infeasible since the second term requires the target sentence to be completely generated. Inspired by recent work on sequence level learning (Ranzato et al., 2015), we treat the problem of generating maximum mutual information response as a reinforcement learning problem in which a reward of mutual information value is observed when the model arrives at the end of a sequence.

Similar to Ranzato et al. (2015), we use policy gradient methods (Sutton et al., 1999; Williams, 1992) for optimization. We initialize the policy model p_{RL} using a pre-trained $p_{\text{SEQ2SEQ}}(a|p_i, q_i)$ model. Given an input source $[p_i, q_i]$, we generate a candidate list $A = \{\hat{a}|\hat{a} \sim p_{RL}\}$. For each generated candidate $\hat{a}$, we will obtain the mutual information score $m(\hat{a}, [p_i, q_i])$ from the pre-trained $p_{\text{SEQ2SEQ}}(a|p_i, q_i)$ and $p^{\text{backward}}_{\text{SEQ2SEQ}}(q_i|a)$. This mutual information score will be used as a reward and back-propagated to the encoder-decoder model, tailoring it to generate sequences with higher rewards. We refer the readers to Zaremba and Sutskever (2015) and Williams (1992) for details. The expected reward for a sequence is given by:

$$J(\theta) = \mathbb{E}[m(\hat{a}, [p_i, q_i])] \tag{5}$$

The gradient is estimated using the likelihood ratio trick:

$$\nabla J(\theta) = m(\hat{a}, [p_i, q_i]) \nabla \log p_{RL}(\hat{a}|[p_i, q_i]) \tag{6}$$

We update the parameters in the encoder-decoder model using stochastic gradient descent. A curriculum learning strategy is adopted (Bengio et al., 2009) as in Ranzato et al. (2015) such that, for every sequence of length T we use the MLE loss for the first L tokens and the reinforcement algorithm for the remaining $T - L$ tokens. We gradually anneal the value of L to zero. A baseline strategy is employed to decrease the learning variance: an additional neural model takes as inputs the generated target and the initial source and outputs a baseline value, similar to the strategy adopted by Zaremba and Sutskever (2015). The final gradient is thus:

$$\nabla J(\theta) = \nabla \log p_{RL}(\hat{a}|[p_i, q_i])[m(\hat{a}, [p_i, q_i]) - b] \tag{7}$$

4.3 Dialogue Simulation between Two Agents

We simulate conversations between the two virtual agents and have them take turns talking with each other. The simulation proceeds as follows: at the initial step, a message from the training set is fed to the first agent. The agent encodes the input message to a vector representation and starts decoding to generate a response output. Combining the immediate output from the first agent with the dialogue history, the second agent updates the state by encoding the dialogue history into a representation and uses the decoder RNN to generate responses, which are subsequently fed back to the first agent, and the process is repeated.

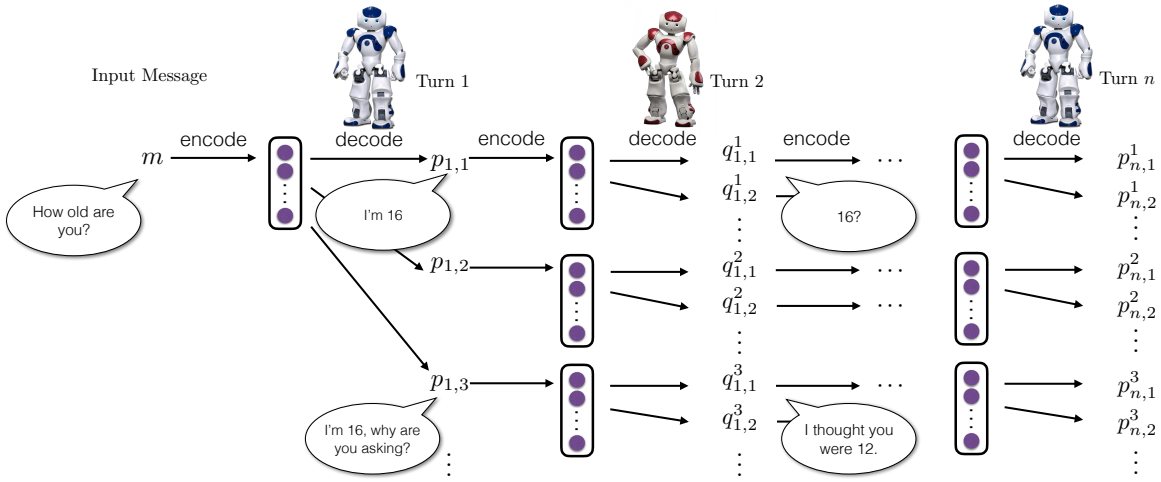


Figure 1: Dialogue simulation between the two agents.

Optimization We initialize the policy model p_{RL} with parameters from the mutual information model described in the previous subsection. We then use policy gradient methods to find parameters that lead to a larger expected reward. The objective to maximize is the expected future reward:

$$J_{RL}(\theta) = \mathbb{E}_{p_{RL}(a_{1:T})}[\sum_{i=1}^{i=T} R(a_i, [p_i, q_i])] \quad (8)$$

where $R(a_i, [p_i, q_i])$ denotes the reward resulting from action a_i. We use the likelihood ratio trick (Williams, 1992; Glynn, 1990; Aleksandrov et al., 1968) for gradient updates:

$$\nabla J_{RL}(\theta) \approx \sum_i \nabla \log p(a_i|p_i, q_i) \sum_{i=1}^{i=T} R(a_i, [p_i, q_i]) \quad (9)$$

We refer readers to Williams (1992) and Glynn (1990) for more details.

4.4 Curriculum Learning

A curriculum Learning strategy is again employed in which we begin by simulating the dialogue for 2 turns, and gradually increase the number of simulated turns. We generate 5 turns at most, as the number of candidates to examine grows exponentially in the size of candidate list. Five candidate responses are generated at each step of the simulation.

5 Experimental Results

In this section, we describe experimental results along with qualitative analysis. We evaluate dialogue generation systems using both human judgments and two automatic metrics: conversation length (number of turns in the entire session) and diversity.

5.1 Dataset

The dialogue simulation requires high-quality initial inputs fed to the agent. For example, an initial input of "why ?" is undesirable since it is unclear how the dialogue could proceed. We take a subset of 10 million messages from the OpenSubtitles dataset and extract 0.8 million sequences with the lowest likelihood of generating the response "*i don't know what you are taking about*" to ensure initial inputs are easy to respond to.

5.2 Automatic Evaluation

Evaluating dialogue systems is difficult. Metrics such as BLEU (Papineni et al., 2002) and perplexity have been widely used for dialogue quality evaluation (Li et al., 2016a; Vinyals and Le, 2015; Sordoni et al., 2015), but it is widely debated how well these automatic metrics are correlated with true response quality (Liu et al., 2016; Galley et al., 2015). Since the goal of the proposed system is not to predict the highest probability response, but rather the long-term success of the dialogue, we do not employ BLEU or perplexity for evaluation[2].

[2]We found the RL model performs worse on BLEU score. On a random sample of 2,500 conversational pairs, single reference BLEU scores for RL models, mutual information models and vanilla SEQ2SEQ models are respectively 1.28, 1.44 and 1.17. BLEU is highly correlated with perplexity in generation tasks.

Model	# of simulated turns
SEQ2SEQ	2.68
mutual information	3.40
RL	4.48

Table 2: The average number of simulated turns from standard SEQ2SEQ models, mutual information model and the proposed RL model.

Length of the dialogue The first metric we propose is the length of the simulated dialogue. We say a dialogue ends when one of the agents starts generating dull responses such as "*i don't know*" [3] or two consecutive utterances from the same user are highly overlapping[4].

The test set consists of 1,000 input messages. To reduce the risk of circular dialogues, we limit the number of simulated turns to be less than 8. Results are shown in Table 2. As can be seen, using mutual information leads to more sustained conversations between the two agents. The proposed RL model is first trained based on the mutual information objective and thus benefits from it in addition to the RL model. We observe that the RL model with dialogue simulation achieves the best evaluation score.

Diversity We report degree of diversity by calculating the number of distinct unigrams and bigrams in generated responses. The value is scaled by the total number of generated tokens to avoid favoring long sentences as described in Li et al. (2016a). The resulting metric is thus a type-token ratio for unigrams and bigrams.

For both the standard SEQ2SEQ model and the proposed RL model, we use beam search with a beam size 10 to generate a response to a given input message. For the mutual information model, we first generate n-best lists using $p_{\text{SEQ2SEQ}}(t|s)$ and then linearly re-rank them using $p_{\text{SEQ2SEQ}}(s|t)$. Results are presented in Table 4. We find that the proposed RL model generates more diverse outputs when compared against both the vanilla SEQ2SEQ model and the mutual information model.

Human Evaluation We explore three settings for human evaluation: the first setting is similar to what was described in Li et al. (2016a), where we employ crowdsourced judges to evaluate a random sample of 500 items. We present both an input message and the generated outputs to 3 judges and ask them to decide which of the two outputs is better (denoted as *single-turn general quality*). Ties are permitted. Identical strings are assigned the same score. We measure the improvement achieved by the RL model over the mutual information model by the mean difference in scores between the models.

For the second setting, judges are again presented with input messages and system outputs, but are asked to decide which of the two outputs is easier to respond to (denoted as *single-turn ease to answer*). Again we evaluate a random sample of 500 items, each being assigned to 3 judges.

For the third setting, judges are presented with simulated conversations between the two agents (denoted as *multi-turn general quality*). Each conversation consists of 5 turns. We evaluate 200 simulated conversations, each being assigned to 3 judges, who are asked to decide which of the simulated conversations is of higher quality.

Results for human evaluation are shown in Table 5. The proposed RL system does not introduce a significant boost in single-turn response quality (winning 40 percent of time and losing 36 percent of time). This is in line with our expectations, as the RL model is not optimized to predict the next utterance, but rather to increase long-term reward. The RL system produces responses that are significantly easier to answer than does the mutual information system, as demonstrated by the *single-turn ease to answer* setting (winning 52 percent of time and losing 23 percent of time), and also significantly higher quality multi-turn dialogues, as demonstrated by the *multi-turn general quality* setting (winning 72 percent of time).

Qualitative Analysis and Discussion We show a random sample of generated responses in Table 3 and simulated conversations in Table 1 at the beginning of the paper. From Table 3, we can see that the RL based agent indeed generates more interactive

Since the RL model is trained based on future reward rather than MLE, it is not surprising that the RL based models achieve lower BLEU score.

[3] We use a simple rule matching method, with a list of 8 phrases that count as dull responses. Although this can lead to both false-positives and -negatives, it works pretty well in practice.

[4] Two utterances are considered to be repetitive if they share more than 80 percent of their words.

Input Message	Mutual Information Model	RL
How old are you?	I'm 16.	I'm 16, why are you asking?
What's your full name?	I have no idea.	What's yours?
I don't want to go home tonight.	Really?	Why?
Do you have any feelings for me?	I don't know what you are talking about.	I don't want to hurt your feelings.
How much time do you have here?	Not long enough. Sorry, sir.	Ten seconds.
Shall we get started?	Of course!	Yes. We've got a lot of work to do here.
Do you play football?	No, i don't	Yes. I love football!
We'd have to talk to him.	I mean, he's a good guy	About what ?
How come you never say it?	Because I don't want to hurt you.	I don't think it's a good idea to say it.

Table 3: Sampled responses generated from the mutual information models and the proposed RL model.

Model	Unigram	Bigram
SEQ2SEQ	0.0062	0.015
mutual information	0.011	0.031
RL	0.017	0.041

Table 4: Diversity scores (type-token ratios) for the standard SEQ2SEQ model, mutual information model and the proposed RL model.

Setting	RL-win	RL-lose	Tie
single-turn general quality	0.40	0.36	0.24
single-turn ease to answer	0.52	0.23	0.25
multi-turn general quality	0.72	0.12	0.16

Table 5: RL gains over the mutual information system based on pairwise human judgments.

responses than the other baselines. We also find that the RL model has a tendency to end a sentence with another question and hand the conversation over to the user. From Table 1, we observe that the RL model manages to produce more interactive and sustained conversations than the mutual information model.

During error analysis, we found that although we penalize repetitive utterances in consecutive turns, the dialogue sometimes enters a cycle with length greater than one, as shown in Table 6. This can be ascribed to the limited amount of conversational history we consider. Another issue observed is that the model sometimes starts a less relevant topic during the conversation. There is a tradeoff between relevance and less repetitiveness, as manifested in the reward function we define in Eq 4.

The fundamental problem, of course, is that the manually defined reward function can't possibly cover the crucial aspects that define an ideal conversation. While the heuristic rewards that we defined are amenable to automatic calculation, and do capture some aspects of what makes a good conversation, ideally the system would instead receive real rewards from humans. Another problem with the current model is that we can only afford to explore a very small number of candidates and simulated turns since the number of cases to consider grow exponentially.

A: What's your name ?
B: Daniel.
A: How old are you ?
B. Twelve. What's your name ?
A. Daniel.
B: How old are you ?
A: Twelve. What's your name ?
B: Daniel.
A: How old are you ?
B ...

Table 6: An simulated dialogue with a cycle longer than one.

6 Conclusion

We introduce a reinforcement learning framework for neural response generation by simulating dialogues between two agents, integrating the strengths of neural SEQ2SEQ systems and reinforcement learning for dialogue. Like earlier neural SEQ2SEQ models, our framework captures the compositional models of the meaning of a dialogue turn and generates semantically appropriate responses. Like reinforcement learning dialogue systems, our framework is able to generate utterances that optimize future reward, successfully capturing global properties of a good conversation. Despite the fact that our model uses very simple, operationable heuristics for capturing these global properties, the framework generates more diverse, interactive responses that foster a more sustained conversation.

Acknowledgement

We would like to thank Chris Brockett, Bill Dolan and other members of the NLP group at Microsoft Research for insightful comments and suggestions. We also want to thank Kelvin Guu, Percy Liang, Chris Manning, Sida Wang, Ziang Xie and other members of the Stanford NLP groups for useful discussions. Jiwei Li is supported by the Facebook Fellowship, to which we gratefully acknowledge. This work is partially supported by the NSF via Awards IIS-1514268, IIS-1464128, and by the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF- 15-1-0462. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, DARPA, or Facebook.

References

V. M. Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. 1968. Stochastic optimization. *Engineering Cybernetics*, 5:11–16.

Jens Allwood, Joakim Nivre, and Elisabeth Ahlsén. 1992. On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, 9:1–26.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Rafael E Banchs and Haizhou Li. 2012. IRIS: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

SRK Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proc. of ACL-IJCNLP*, pages 445–450, Beijing, China, July.

Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013a. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of SIGDIAL*.

Milica Gasic, Catherine Breslin, Mike Henderson, Dongkyu Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013b. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *Proceedings of ICASSP 2013*, pages 8367–8371. IEEE.

Milica Gašic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental online adaptation of pomdp-based dialogue managers to extended domains. *In Proceedings on InterSpeech*.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany, August.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 72–79. IEEE.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany, August.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. LSTM based conversation models. *arXiv preprint arXiv:1603.09457*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. 2014. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 355–361. Springer.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? Research in commercial spoken dialog systems. In *Text, Speech and Dialogue*, pages 3–13. Springer.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Adwait Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, 16(3):435–455.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP 2011*, pages 583–593.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(02):97–126.

Emanuel A. Schegloff and Harvey Sacks. 1973. Opening up closings. *Semiotica*, 8(4):289–327.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, February.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL-IJCNLP*, pages 1577–1586.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Nips*, pages 956–962.

Satinder Singh, Michael Kearns, Diane J Litman, Marilyn A Walker, et al. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *AAAI/IAAI*, pages 645–651.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, pages 105–133.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arxiv*.

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

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of ACL 2010*, pages 806–814.

Marilyn A Walker, Rashmi Prasad, and Amanda Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *Proceeedings of INTERSPEECH 2003*.

Marilyn A. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, pages 387–416.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*, pages 1711–1721, Lisbon, Portugal.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. In *NIPS workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural Turing machines. *arXiv preprint arXiv:1505.00521*.

Neural Text Generation from Structured Data with Application to the Biography Domain

Rémi Lebret* EPFL, Switzerland
David Grangier Facebook AI Research
Michael Auli Facebook AI Research

Abstract

This paper introduces a neural model for concept-to-text generation that scales to large, rich domains. It generates biographical sentences from fact tables on a new dataset of biographies from Wikipedia. This set is an order of magnitude larger than existing resources with over 700k samples and a 400k vocabulary. Our model builds on conditional neural language models for text generation. To deal with the large vocabulary, we extend these models to mix a fixed vocabulary with *copy actions* that transfer sample-specific words from the input database to the generated output sentence. To deal with structured data, we allow the model to embed words differently depending on the data fields in which they occur. Our neural model significantly outperforms a Templated Kneser-Ney language model by nearly 15 BLEU.

1 Introduction

Concept-to-text generation renders structured records into natural language (Reiter et al., 2000). A typical application is to generate a weather forecast based on a set of structured meteorological measurements. In contrast to previous work, we scale to the large and very diverse problem of generating biographies based on Wikipedia infoboxes. An infobox is a fact table describing a person, similar to a person subgraph in a knowledge base (Bollacker et al., 2008; Ferrucci, 2012). Similar generation applications include the generation of product descriptions based on a catalog of millions of items with dozens of attributes each.

Previous work experimented with datasets that contain only a few tens of thousands of records such as WEATHERGOV or the ROBOCUP dataset, while our dataset contains over 700k biographies from Wikipedia. Furthermore, these datasets have a limited vocabulary of only about 350 words each, compared to over 400k words in our dataset.

To tackle this problem we introduce a statistical generation model conditioned on a Wikipedia infobox. We focus on the generation of the first sentence of a biography which requires the model to select among a large number of possible fields to generate an adequate output. Such diversity makes it difficult for classical count-based models to estimate probabilities of rare events due to data sparsity. We address this issue by parameterizing words and fields as embeddings, along with a neural language model operating on them (Bengio et al., 2003). This factorization allows us to scale to a larger number of words and fields than Liang et al. (2009), or Kim and Mooney (2010) where the number of parameters grows as the product of the number of words and fields.

Moreover, our approach does not restrict the relations between the field contents and the generated text. This contrasts with less flexible strategies that assume the generation to follow either a hybrid alignment tree (Kim and Mooney, 2010), a probabilistic context-free grammar (Konstas and Lapata, 2013), or a tree adjoining grammar (Gyawali and Gardent, 2014).

Our model exploits structured data both globally and locally. Global conditioning summarizes all information about a personality to understand high-level themes such as that the biography is about a scientist or an artist, while as local conditioning describes the previously generated tokens in terms of the their relationship to the infobox. We analyze the effectiveness of each and demonstrate their complementarity.

2 Related Work

Traditionally, generation systems relied on rules and hand-crafted specifications (Dale et al., 2003; Reiter et al., 2005; Green, 2006; Galanis and Androut-

*Rémi performed this work while interning at Facebook.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1203–1213,
Austin, Texas, November 1-5, 2016.

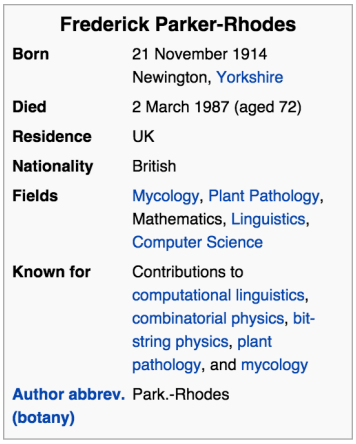
sopoulos, 2007; Turner et al., 2010). Generation is divided into modular, yet highly interdependent, decisions: (1) *content planning* defines which parts of the input fields or meaning representations should be selected; (2) *sentence planning* determines which selected fields are to be dealt with in each output sentence; and (3) *surface realization* generates those sentences.

Data-driven approaches have been proposed to automatically learn the individual modules. One approach first aligns records and sentences and then learns a content selection model (Duboue and McKeown, 2002; Barzilay and Lapata, 2005). Hierarchical hidden semi-Markov generative models have also been used to first determine which facts to discuss and then to generate words from the predicates and arguments of the chosen facts (Liang et al., 2009). Sentence planning has been formulated as a supervised set partitioning problem over facts where each partition corresponds to a sentence (Barzilay and Lapata, 2006). End-to-end approaches have combined sentence planning and surface realization by using explicitly aligned sentence/meaning pairs as training data (Ratnaparkhi, 2002; Wong and Mooney, 2007; Belz, 2008; Lu and Ng, 2011). More recently, content selection and surface realization have been combined (Angeli et al., 2010; Kim and Mooney, 2010; Konstas and Lapata, 2013).

At the intersection of rule-based and statistical methods, hybrid systems aim at leveraging human contributed rules and corpus statistics (Langkilde and Knight, 1998; Soricut and Marcu, 2006; Mairesse and Walker, 2011).

Our approach is inspired by the recent success of neural language models for image captioning (Kiros et al., 2014; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Fang et al., 2015; Xu et al., 2015), machine translation (Devlin et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), and modeling conversations and dialogues (Shang et al., 2015; Wen et al., 2015; Yao et al., 2015).

Our model is most similar to Mei et al. (2016) who use an encoder-decoder style neural network model to tackle the WEATHERGOV and ROBOCUP tasks. Their architecture relies on LSTM units and an attention mechanism which reduces scalability compared to our simpler design.



Frederick Parker-Rhodes

Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Author abbrev. (botany)	Park.-Rhodes

Figure 1: Wikipedia infobox of Frederick Parker-Rhodes. The introduction of his article reads: "Frederick Parker-Rhodes (21 March 1914 – 21 November 1987) was an English linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist.".

3 Language Modeling for Constrained Sentence generation

Conditional language models are a popular choice to generate sentences. We introduce a table-conditioned language model for constraining text generation to include elements from fact tables.

3.1 Language model

Given a sentence $s = w_1, \ldots, w_T$ with T words from vocabulary $\mathcal{W}$, a language model estimates:

$$P(s) = \prod_{t=1}^{T} P(w_t | w_1, \ldots, w_{t-1}) \,. \quad (1)$$

Let $c_t = w_{t-(n-1)}, \ldots, w_{t-1}$ be the sequence of $n-1$ context words preceding w_t. An n-gram language model makes an order n Markov assumption,

$$P(s) \approx \prod_{t=1}^{T} P(w_t | c_t) \,. \quad (2)$$

3.2 Language model conditioned on tables

A table is a set of field/value pairs, where values are sequences of words. We therefore propose language models that are conditioned on these pairs.

Local conditioning refers to the information from the table that is applied to the description of the words which have already generated, i.e. the previous words that constitute the context of the language

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	$\emptyset$	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	$\emptyset$	$\emptyset$	$\emptyset$

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	...	april	...	placeholder	...	john	...	doe
w	1	...	92	...	5302	...	13944	...	unk
z_w	$\emptyset$		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

Figure 2: Table features (right) for an example table (left); $\mathcal{W} \cup \mathcal{Q}$ is the set of all output words as defined in Section 3.3.

model. The table allows us to describe each word not only by its string (or index in the vocabulary) but also by a descriptor of its occurrence in the table. Let $\mathcal{F}$ define the set of all possible fields f. The occurrence of a word w in the table is described by a set of (field, position) pairs.

$$z_w = \left\{(f_i, p_i)\right\}_{i=1}^{m}, \tag{3}$$

where m is the number of occurrences of w. Each pair (f, p) indicates that w occurs in field f at position p. In this scheme, most words are described by the empty set as they do not occur in the table. For example, the word *linguistics* in the table of Figure 1 is described as follows:

$$z_{\text{linguistics}} = \{(\text{fields}, 8); (\text{known for}, 4)\}, \tag{4}$$

assuming words are lower-cased and commas are treated as separate tokens.

Conditioning both on the field type and the position within the field allows the model to encode field-specific regularities, e.g., a number token in a date field is likely followed by a month token; knowing that the number is the first token in the date field makes this even more likely.

The (field, position) description scheme of the table does not allow to express that a token terminates a field which can be useful to capture field transitions. For biographies, the last token of the name field is often followed by an introduction of the birth date like '(' or 'was born'. We hence extend our descriptor to a triplet that includes the position of the token counted from the end of the field:

$$z_w = \left\{(f_i, p_i^+, p_i^-)\right\}_{i=1}^{m}, \tag{5}$$

where our example becomes:

$$z_{\text{linguistics}} = \{(\text{fields}, 8, 4); (\text{known for}, 4, 13)\}.$$

We extend Equation 2 to use the above information as additional conditioning context when generating a sentence s:

$$P(s|z) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}), \tag{6}$$

where $z_{c_t} = z_{w_{t-(n-1)}}, \ldots, z_{w_{t-1}}$ are referred to as the local conditioning variables since they describe the local context (previous word) relations with the table.

Global conditioning refers to information from all tokens and fields of the table, regardless whether they appear in the previous generated words or not. The set of fields available in a table often impacts the structure of the generation. For biographies, the fields used to describe a politician are different from the ones for an actor or an athlete. We introduce global conditioning on the available fields g_f as

$$P(s|z, g_f) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f). \tag{7}$$

Similarly, global conditioning g_w on the available

words occurring in the table is introduced:

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f, g_w). \quad (8)$$

Tokens provide information complementary to fields. For example, it may be hard to distinguish a basketball player from a hockey player by looking only at the field names, e.g. teams, league, position, weight and height, etc. However the actual field tokens such as team names, league name, player's position can help the model to give a better prediction. Here, $g_f \in \{0,1\}^{\mathcal{F}}$ and $g_w \in \{0,1\}^{\mathcal{W}}$ are binary indicators over fixed field and word vocabularies.

Figure 2 illustrates the model with a schematic example. For predicting the next word w_t after a given context c_t, the language model is conditioned on sets of triplets for each word occurring in the table z_{c_t}, along with all fields and words from this table.

3.3 Copy actions

So far we extended the model conditioning with features derived from the fact table. We now turn to using table information when scoring output words. In particular, sentences which express facts from a given table often copy words from the table. We therefore extend our model to also score special field tokens such as `name_1` or `name_2` which are subsequently added to the score of the corresponding words from the field value.

Our model reads a table and defines an output domain $\mathcal{W} \cup \mathcal{Q}$. $\mathcal{Q}$ defines all tokens in the table, which might include out of vocabulary words ($\notin \mathcal{W}$). For instance *Park-Rhodes* in Figure 1 is not in $\mathcal{W}$. However, *Park-Rhodes* will be included in $\mathcal{Q}$ as `name_2` (since it is the second token of the name field) which allows our model to generate it. This mechanism is inspired by recent work on attention based word copying for neural machine translation (Luong et al., 2015) as well as delexicalization for neural dialog systems (Wen et al., 2015). It also builds upon older work such as class-based language models for dialog systems (Oh and Rudnicky, 2000).

4 A Neural Language Model Approach

A feed-forward neural language model (NLM) estimates $P(w_t|c_t)$ with a parametric function $\boldsymbol{\phi}_\theta$ (Equation 1), where θ refers to all learnable parameters of the network. This function is a composition of simple differentiable functions or *layers*.

4.1 Mathematical notations and layers

We denote matrices as bold upper case letters ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$), and vectors as bold lower-case letters ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$). $\mathbf{A}_i$ represents the i^{th} row of matrix $\mathbf{A}$. When $\mathbf{A}$ is a 3-d matrix, then $\mathbf{A}_{i,j}$ represents the vector of the i^{th} first dimension and j^{th} second dimension. Unless otherwise stated, vectors are assumed to be column vectors. We use $[\mathbf{v}_1; \mathbf{v}_2]$ to denote vector concatenation. Next, we introduce the notation for the different layers used in our approach.

Embedding layer. Given a parameter matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, the *embedding layer* is a lookup table that performs an array indexing operation:

$$\boldsymbol{\psi}_{\mathbf{X}}(x_i) = \mathbf{X}_i \in \mathbb{R}^d, \quad (9)$$

where $\mathbf{X}_i$ corresponds to the embedding of the element x_i at row i. When $\mathbf{X}$ is a 3-d matrix, the lookup table takes two arguments:

$$\boldsymbol{\psi}_{\mathbf{X}}(x_i, x_j) = \mathbf{X}_{i,j} \in \mathbb{R}^d, \quad (10)$$

where $\mathbf{X}_{i,j}$ corresponds to the embedding of the pair (x_i, x_j) at index (i, j). The lookup table operation can be applied for a sequence of elements $s = x_1, \ldots, x_T$. A common approach is to concatenate all resulting embeddings:

$$\boldsymbol{\psi}_{\mathbf{X}}(s) = \left[\boldsymbol{\psi}_{\mathbf{X}}(x_1); \ldots; \boldsymbol{\psi}_{\mathbf{X}}(x_T)\right] \in \mathbb{R}^{T \times d}. \quad (11)$$

Linear layer. This layer applies a linear transformation to its inputs $\mathbf{x} \in \mathbb{R}^n$:

$$\boldsymbol{\gamma}_\theta(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (12)$$

where $\theta = \{\mathbf{W}, \mathbf{b}\}$ are the trainable parameters with $\mathbf{W} \in \mathbb{R}^{m \times n}$ being the weight matrix, and $\mathbf{b} \in \mathbb{R}^m$ is the bias term.

Softmax layer. Given a context input c_t, the final layer outputs a score for each word $w_t \in \mathcal{W}$, $\boldsymbol{\phi}_\theta(c_t) \in \mathbb{R}^{|\mathcal{W}|}$. The probability distribution is obtained by applying the softmax activation function:

$$P(w_t = w|c_t) = \frac{\exp(\boldsymbol{\phi}_\theta(c_t, w))}{\sum_{i=1}^{|\mathcal{W}|} \exp(\boldsymbol{\phi}_\theta(c_t, w_i))} \quad (13)$$

4.2 Embeddings as inputs

A key aspect of neural language models is the use of word embeddings. Similar words tend to have similar embeddings and thus share latent features. The probability estimates of those models are smooth functions of these embeddings, and a small change in the features results in a small change in the probability estimates (Bengio et al., 2003). Therefore, neural language models can achieve better generalization for unseen n-grams. Next, we show how we map fact tables to continuous space in similar spirit.

Word embeddings. Formally, the embedding layer maps each context word index to a continuous d-dimensional vector. It relies on a parameter matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{W}| \times d}$ to convert the input c_t into $n-1$ vectors of dimension d:

$$\boldsymbol{\psi}_{\mathbf{E}}(c_t) = \left[\boldsymbol{\psi}_{\mathbf{E}}(w_{t-(n-1)}); \ldots; \boldsymbol{\psi}_{\mathbf{E}}(w_{t-1})\right]. \quad (14)$$

$\mathbf{E}$ can be initialized randomly or with pre-trained word embeddings.

Table embeddings. As described in Section 3.2, the language model is conditioned on elements from the table. Embedding matrices are therefore defined to model both local and global conditioning information. For local conditioning, we denote the maximum length of a sequence of words as l. Each field $f_j \in \mathcal{F}$ is associated with $2 \times l$ vectors of d dimensions, the first l of those vectors embed all possible starting positions $1, \ldots, l$, and the remaining l vectors embed ending positions. This results in two parameter matrices $\mathbf{Z} = \{\mathbf{Z}^+, \mathbf{Z}^-\} \in \mathbb{R}^{|\mathcal{F}| \times l \times d}$. For a given triplet (f_j, p_i^+, p_i^-), $\boldsymbol{\psi}_{\mathbf{Z}^+}(f_j, p_i^+)$ and $\boldsymbol{\psi}_{\mathbf{Z}^-}(f_j, p_i^-)$ refer to the embedding vectors of the start and end position for field f_j, respectively.

Finally, global conditioning uses two parameter matrices $\mathbf{G}^f \in \mathbb{R}^{|\mathcal{F}| \times g}$ and $\mathbf{G}^w \in \mathbb{R}^{|\mathcal{W}| \times g}$. $\boldsymbol{\psi}_{\mathbf{G}^f}(f_j)$ maps a table field f_j into a vector of dimension g, while $\boldsymbol{\psi}_{\mathbf{G}^w}(w_t)$ maps a word w_t into a vector of the same dimension. In general, $\mathbf{G}^w$ shares its parameters with $\mathbf{E}$, provided $d = g$.

Aggregating embeddings. We represent each occurence of a word w as a triplet (field, start, end) where we have embeddings for the start and end position as described above. Often times a particular word w occurs multiple times in a table, e.g., 'linguistics' has two instances in Figure 1. In this case, we perform a component-wise max over the start embeddings of all instances of w to obtain the best features across all occurrences of w. We do the same for end position embeddings:

$$\boldsymbol{\psi}_{\mathbf{Z}}(z_{w_t}) = \Big[\max\left\{\boldsymbol{\psi}_{\mathbf{Z}^+}(f_j, p_i^+), \forall (f_j, p_i^+, p_i^-) \in z_{w_t}\right\}; \max\left\{\boldsymbol{\psi}_{\mathbf{Z}^-}(f_j, p_i^-), \forall (f_j, p_i^+, p_i^-) \in z_{w_t}\right\}\Big] \quad (15)$$

A special no-field embedding is assigned to w_t when the word is not associated to any fields. An embedding $\boldsymbol{\psi}_{\mathbf{Z}}(z_{c_t})$ for encoding the local conditioning of the input c_t is obtained by concatenation.

For global conditioning, we define $\mathcal{F}^q \subset \mathcal{F}$ as the set of all the fields in a given table q, and $\mathcal{Q}$ as the set of all words in q. We also perform max aggregation. This yields the vectors

$$\boldsymbol{\psi}_{\mathbf{G}^f}(g_f) = \max\left\{\boldsymbol{\psi}_{\mathbf{G}^f}(f_j), \forall f_j \in \mathcal{F}^q\right\}, \quad (16)$$

and

$$\boldsymbol{\psi}_{\mathbf{G}^w}(g_w) = \max\left\{\boldsymbol{\psi}_{\mathbf{G}^w}(w_t), \forall w_t \in \mathcal{Q}\right\}. \quad (17)$$

The final embedding which encodes the context input with conditioning is then the concatenation of these vectors:

$$\boldsymbol{\psi}_{\alpha_1}(c_t, z_{c_t}, g_f, g_w) = \left[\boldsymbol{\psi}_{\mathbf{E}}(c_t);\, \boldsymbol{\psi}_{\mathbf{Z}}(z_{c_t});\, \boldsymbol{\psi}_{\mathbf{G}^f}(g_f);\, \boldsymbol{\psi}_{\mathbf{G}^w}(g_w)\right] \in \mathbb{R}^{d^1}, \quad (18)$$

with $\alpha_1 = \{\mathbf{E}, \mathbf{Z}^+, \mathbf{Z}^-, \mathbf{G}^f, \mathbf{G}^w\}$ and $d^1 = (n-1) \times (3 \times d) + (2 \times g)$. For simplification purpose, we define the context input $x = \{c_t, z_{c_t}, g_f, g_w\}$ in the following equations. This context embedding is mapped to a latent context representation using a linear operation followed by a hyperbolic tangent:

$$\mathbf{h}(x) = \tanh\Big(\boldsymbol{\gamma}_{\alpha_2}\big(\boldsymbol{\psi}_{\alpha_1}(x)\big)\Big) \in \mathbb{R}^{\text{nhu}}, \quad (19)$$

where $\alpha_2 = \{\mathbf{W}_2, \mathbf{b}_2\}$, with $\mathbf{W}_2 \in \mathbb{R}^{\text{nhu} \times d^1}$ and $\mathbf{b}_2 \in \mathbb{R}^{\text{nhu}}$.

4.3 In-vocabulary outputs

The hidden representation of the context then goes to another linear layer to produce a real value score for each word in the vocabulary:

$$\boldsymbol{\phi}_{\alpha}^{\mathcal{W}}(x) = \boldsymbol{\gamma}_{\alpha_3}\Big(\mathbf{h}(x)\Big) \in \mathbb{R}^{|\mathcal{W}|}, \quad (20)$$

where $\alpha_3 = \{\mathbf{W}_3, \mathbf{b}_3\}$, with $\mathbf{W}_3 \in \mathbb{R}^{|\mathcal{W}| \times \text{nhu}}$ and $\mathbf{b}_3 \in \mathbb{R}^{|\mathcal{W}|}$, and $\alpha = \{\alpha_1, \alpha_2, \alpha_3\}$.

4.4 Mixing outputs for better copying

Section 3.3 explains that each word w from the table is also associated with z_w, the set of fields in which it occurs, along with the position in that field. Similar to local conditioning, we represent each field and position pair (f_j, p_i) with an embedding $\boldsymbol{\psi}_{\mathbf{F}}(f_j, p_i)$, where $\mathbf{F} \in \mathbb{R}^{|\mathcal{F}| \times l \times d}$. These embeddings are then projected into the same space as the latent representation of context input $\mathbf{h}(x) \in \mathbb{R}^{\text{nhu}}$. Using the max operation over the embedding dimension, each word is finally embedded into a unique vector:

$$\mathbf{q}(w) = \max \Big\{ \tanh\Big(\boldsymbol{\gamma}_\beta\big(\boldsymbol{\psi}_{\mathbf{F}}(f_j, p_i)\big)\Big), \forall (f_j, p_i) \in z_w \Big\}, \quad (21)$$

where $\beta = \{\mathbf{W}_4, \mathbf{b}_4\}$ with $\mathbf{W}_4 \in \mathbb{R}^{\text{nhu} \times d}$, and $\mathbf{b}_4 \in \mathbb{R}^{\text{nhu}}$. A dot product with the context vector produces a score for each word w in the table,

$$\phi_\beta^{\mathcal{Q}}(x, w) = \mathbf{h}(x) \cdot \mathbf{q}(w)\,. \quad (22)$$

Each word $w \in \mathcal{W} \cup \mathcal{Q}$ receives a final score by summing the vocabulary score and the field score:

$$\phi_\theta(x, w) = \phi_\alpha^{\mathcal{W}}(x, w) + \phi_\beta^{\mathcal{Q}}(x, w)\,, \quad (23)$$

with $\theta = \{\alpha, \beta\}$, and where $\phi_\beta^{\mathcal{Q}}(x, w) = 0$ when $w \notin \mathcal{Q}$. The softmax function then maps the scores to a distribution over $\mathcal{W} \cup \mathcal{Q}$,

$$\log P(w|x) = \phi_\theta(x, w) - \log \sum_{w' \in \mathcal{W} \cup \mathcal{Q}} \exp \phi_\theta(x, w')\,.$$

4.5 Training

The neural language model is trained to minimize the negative log-likelihood of a training sentence s with stochastic gradient descent (SGD; LeCun et al. 2012) :

$$L_\theta(s) = -\sum_{t=1}^{T} \log P(w_t | c_t, z_{c_t}, g_f, g_w)\,. \quad (24)$$

5 Experiments

Our neural network model (Section 4) is designed to generate sentences from tables for large-scale problems, where a diverse set of sentence types need to be generated. Biographies are therefore a good framework to evaluate our model, with Wikipedia offering a large and diverse dataset.

5.1 Biography dataset

We introduce a new dataset for text generation, WIKIBIO, a corpus of 728,321 articles from English Wikipedia (Sep 2015). It comprises all biography articles listed by WikiProject Biography[1] which also have a table (infobox). We extract and tokenize the first sentence of each article with Stanford CoreNLP (Manning et al., 2014). All numbers are mapped to a special token, except for years which are mapped to different special token. Field values from tables are similarly tokenized. All tokens are lower-cased. Table 2 summarizes the dataset statistics: on average, the first sentence is twice as short as the table (26.1 vs 53.1 tokens), about a third of the sentence tokens (9.5) also occur in the table. The final corpus has been divided into three sub-parts to provide training (80%), validation (10%) and test sets (10%). The dataset is available for download[2].

5.2 Baseline

Our baseline is an interpolated Kneser-Ney (KN) language model and we use the KenLM toolkit to train 5-gram models without pruning (Heafield et al., 2013). We also learn a KN language model over templates. For that purpose, we replace the words occurring in both the table and the training sentences with a special token reflecting its table descriptor z_w (Equation 3). The introduction section of the table in Figure 1 looks as follows under this scheme: "`name_1 name_2` (`birthdate_1 birthdate_2 birthdate_3` – `deathdate_1 deathdate_2 deathdate_3`) was an english linguist , `fields_3` pathologist , `fields_10` scientist , mathematician , mystic and mycologist ." During inference, the decoder is constrained to emit words from the regular vocabulary or special tokens occurring in the input table. When picking a special token we copy the corresponding word from the table.

5.3 Training setup

For our neural models, we train 11-gram language models ($n = 11$) with a learning rate set to 0.0025.

[1] https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Biography

[2] https://github.com/DavidGrangier/wikipedia-biography-dataset

Model	Perplexity	BLEU	ROUGE	NIST
KN	10.51	2.21	0.38	0.93
NLM	9.40 ± 0.01	2.41 ± 0.33	0.52 ± 0.08	1.27 ± 0.26
+ Local (field, start, end)	8.61 ± 0.01	4.17 ± 0.54	1.48 ± 0.23	1.41 ± 0.11
Template KN	$7.46^{\star}$	19.8	10.7	5.19
Table NLM w/ Local (field, start)	$4.60 \pm 0.01^{\dagger}$	26.0 ± 0.39	19.2 ± 0.23	6.08 ± 0.08
+ Local (field, start, end)	$4.60 \pm 0.01^{\dagger}$	26.6 ± 0.42	19.7 ± 0.25	6.20 ± 0.09
+ Global (field)	$\mathbf{4.30 \pm 0.01}^{\dagger}$	33.4 ± 0.18	23.9 ± 0.12	7.52 ± 0.03
+ Global (field & word)	$4.40 \pm 0.02^{\dagger}$	$\mathbf{34.7 \pm 0.36}$	$\mathbf{25.8 \pm 0.36}$	$\mathbf{7.98 \pm 0.07}$

Table 1: BLEU, ROUGE, NIST and perplexity without copy actions (first three rows) and with copy actions (last five rows). For neural models we report "mean $\pm$ standard deviation" for five training runs with different initialization. Decoding beam width is 5. Perplexities marked with * and † are not directly comparable as the output vocabularies differ slightly.

	Mean	Percentile 5%	Percentile 95%
# tokens per sentence	26.1	13	46
# tokens per table	53.1	20	108
# table tokens per sent.	9.5	3	19
# fields per table	19.7	9	36

Table 2: Dataset statistics

Parameter	Value		
# word types	$\lvert\mathcal{W}\rvert$	=	$20,000$
# field types	$\lvert\mathcal{F}\rvert$	=	$1,740$
Max. # tokens in a field	l	=	10
word/field embedding size	d	=	64
global embedding size	g	=	128
# hidden units	nhu	=	256

Table 3: Model Hyperparameters

Table 3 describes the other hyper-parameters. We include all fields occurring at least 100 times in the training data in $\mathcal{F}$, the set of fields. We include the $20,000$ most frequent words in the vocabulary. The other hyperparameters are set through validation, maximizing BLEU over a validation subset of $1,000$ sentences. Similarly, early stopping is applied: training ends when BLEU stops improving on the same validation subset. One should note that the maximum number of tokens in a field $l = 10$ means that we encode only 10 positions: for longer field values the final tokens are not dropped but their position is capped to 10. We initialize the word embeddings W from Hellinger PCA computed over the set of training biographies. This representation has shown to be helpful for various applications (Lebret and Collobert, 2014).

5.4 Evaluation metrics

We use different metrics to evaluate our models. Performance is first evaluated in terms of perplexity which is the standard metric for language modeling. Generation quality is assessed automatically with BLEU-4, ROUGE-4 (F-measure) and NIST-4[3] (Belz and Reiter, 2006).

6 Results

This section describes our results and discusses the impact of the different conditioning variables.

6.1 The more, the better

The results (Table 1) show that more conditioning information helps to improve the performance of our models. The generation metrics BLEU, ROUGE and NIST all gives the same performance ordering over models. We first discuss models without copy actions (the first three results) and then discuss models with copy actions (the remaining results). Note that the factorization of our models results in three different output domains which makes perplexity comparisons less straightforward: models without copy actions operate over a fixed vocabulary. Template KN adds a fixed set of field/position pairs to this vocabulary while Table NLM models a variable set $\mathcal{Q}$ depending on the input table, see Section 3.3.

Without copy actions. In terms of perplexity the (i) neural language model (NLM) is slightly better

[3]We rely on standard software, NIST mteval-v13a.pl (for NIST, BLEU), and MSR rouge-1.5.5 (for ROUGE).

than an interpolated KN language model, and (ii) adding local conditioning on the field start and end position further improves accuracy. Generation metrics are generally very low but there is a clear improvement when using local conditioning since it allows to learn transitions between fields by linking previous predictions to the table unlike KN or plain NLM.

With copy actions. For experiments with copy actions we use the full local conditioning (Equation 4) in the neural language models. BLEU, ROUGE and NIST all improves when moving from Template KN to Table NLM and more features successively improve accuracy. Global conditioning on the fields improves the model by over 7 BLEU and adding words gives an additional 1.3 BLEU. This is a total improvement of nearly 15 BLEU over the Template Kneser-Ney baseline. Similar observations are made for ROUGE +15 and NIST +2.8.

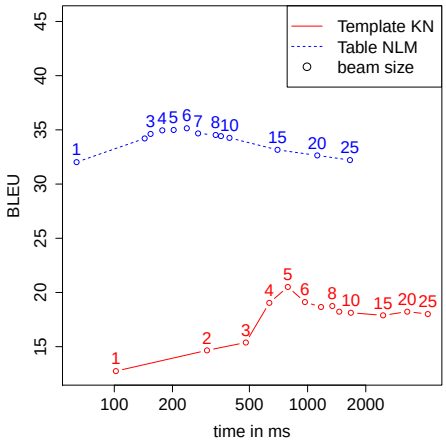


Figure 3: Comparison between our best model (Table NLM) and the baseline (Template KN) for different beam sizes. The x-axis is the average timing (in milliseconds) for generating one sentence. The y-axis is the BLEU score. All results are measured on a subset of 1,000 samples of the validation set.

6.2 Attention mechanism

Our model implements attention over input table fields. For each word w in the table, Equation (23) takes the language model score $\phi_{c_t}^{\mathcal{W}}$ and adds a bias $\phi_{c_t}^{\mathcal{Q}}$. The bias is the dot-product between a representation of the table field in which w occurs and a representation of the context, Equation (22) that summarizes the previously generated fields and words.

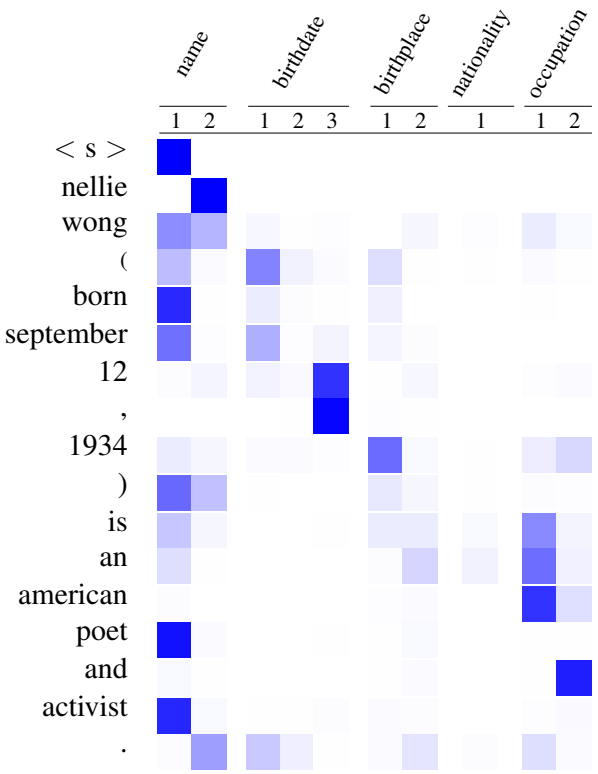


Figure 4: Visualization of attention scores for Nellie Wong's Wikipedia infobox. Each row represents the probability distribution over (field, position) pairs given the previous words (i.e. the words heading the preceding rows as well as the current row). Darker colors depict higher probabilities.

Figure 4 shows that this mechanism adds a large bias to continue a field if it has not generated all tokens from the table, e.g., it emits the word occurring in `name_2` after generating `name_1`. It also nicely handles transitions between field types, e.g., the model adds a large bias to the words occurring in the *occupation* field after emitting the *birthdate*.

6.3 Sentence decoding

We use a standard beam search to explore a larger set of sentences compared to simple greedy search. This allows us to explore K times more paths which comes at a linear increase in the number of forward computation steps for our language model. We compare various beam settings for the baseline Template KN and our Table NLM (Figure 3). The best validation BLEU can be obtained with a beam size of $K = 5$. Our model is also several times faster than the baseline, requiring only about 200 ms per sentence with $K = 5$. Beam search generates many n-gram lookups for Kneser-Ney which requires many

Model	Generated Sentence
Reference	frederick parker-rhodes (21 march 1914 – 21 november 1987) was an english linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist.
Baseline (Template KN)	frederick parker-rhodes (born november 21 , 1914 – march 2 , 1987) was an english cricketer .
Table NLM +Local (field, start)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was an australian rules footballer who played with carlton in the victorian football league (vfl) during the XXXXs and XXXXs .
+ Global (field)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was an english mycology and plant pathology , mathematics at the university of uk .
+ Global (field, word)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was a british computer scientist , best known for his contributions to computational linguistics .

Table 4: First sentence from the current Wikipedia article about Frederick Parker-Rhodes and the sentences generated from the three versions of our table-conditioned neural language model (Table NLM) using the Wikipedia infobox seen in Figure 1.

random memory accesses; while neural models perform scoring through matrix-matrix products, an operation which is more local and can be performed in a block parallel manner where modern graphic processors shine (Kindratenko, 2014).

6.4 Qualitative analysis

Table 4 shows generations for different variants of our model based on the Wikipedia table in Figure 1. First of all, comparing the reference to the fact table reveals that our training data is not perfect. The birth month mentioned in the fact table and the first sentence of the Wikipedia article are different; this may have been introduced by one contributor editing the article and not keeping the information consistent.

All three versions of our model correctly generate the beginning of the sentence by copying the name, the birth date and the death date from the table. The model correctly uses the past tense since the death date in the table indicates that the person has passed away. Frederick Parker-Rhodes was a scientist, but this occupation is not directly mentioned in the table. The model without global conditioning can therefore not predict the right occupation, and it continues the generation with the most common occupation (in Wikipedia) for a person who has died. In contrast, the global conditioning over the fields helps the model to understand that this person was indeed a scientist. However, it is only with the global conditioning on the words that the model can infer the correct occupation, i.e., *computer scientist*.

7 Conclusions

We have shown that our model can generate fluent descriptions of arbitrary people based on structured data. Local and global conditioning improves our model by a large margin and we outperform a Kneser-Ney language model by nearly 15 BLEU. Our task uses an order of magnitude more data than previous work and has a vocabulary that is three orders of magnitude larger.

In this paper, we have only focused on generating the first sentence and we will tackle the generation of longer biographies in future work. Also, the encoding of field values can be improved. Currently, we only attach the field type and token position to each word type and perform a max-pooling for local conditioning. One could leverage a richer representation by learning an encoder conditioned on the field type, e.g. a recurrent encoder or a convolutional encoder with different pooling strategies.

Furthermore, the current training loss function does not explicitly penalize the model for generating incorrect facts, e.g. predicting an incorrect nationality or occupation is currently not considered worse than choosing an incorrect determiner. A loss function that could assess factual accuracy would certainly improve sentence generation by avoiding such mistakes. Also it will be important to define a strategy for evaluating the factual accuracy of a generation, beyond BLEU, ROUGE or NIST.

References

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

R. Barzilay and M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338.

R. Barzilay and M. Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Association for Computational Linguistics.

A. Belz and E. Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *In Proc. EACL06*, pages 313–320.

A. Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(04):431–455.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *International Conference on Management of Data*, pages 1247–1250. ACM.

R. Dale, S. Geldof, and J.-P. Prost. 2003. Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 35–44. Australian Computer Society, Inc.

J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1370–1380.

P. A. Duboue and K. R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of INLG 2002*, pages 89–96.

H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollar, J. Gao, X. He, M. Mitchell, J. C. Platt, L. C. Zitnick, and G. Zweig. 2015. From captions to visual concepts and back. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

D. Ferrucci. 2012. Introduction to this is watson. *IBM Journal of Research and Development*, 56(3.4):1–1.

D. Galanis and I. Androutsopoulos. 2007. Generating multilingual descriptions from linguistically annotated owl ontologies: the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 143–146. Association for Computational Linguistics.

N. Green. 2006. Generation of biomedical arguments for lay readers. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 114–121. Association for Computational Linguistics.

B. Gyawali and C. Gardent. 2014. Surface realisation from knowledge-bases. In *Proc. of ACL*.

K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

A. Karpathy and L. Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

J. Kim and R. J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

V. Kindratenko. 2014. *Numerical Computations with GPUs*. Springer.

R. Kiros, R. Salakhutdinov, and R. S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

I. Konstas and M. Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Int. Res.*, 48(1):305–346, October.

I. Langkilde and K. Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proc. ACL*, pages 704–710.

R. Lebret and R. Collobert. 2014. Word embeddings through hellinger pca. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April. Association for Computational Linguistics.

Y. A LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

W. Lu and H. T. Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622. Association for Computational Linguistics.

M.-T. Luong, I. Sutskever, Q. V Le, O. Vinyals, and W. Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proc. ACL*, pages 11–19.

F. Mairesse and M. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Comput. Linguist.*, 37(3):455–488.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

H. Mei, M. Bansal, and M. R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of Human Language Technologies: The 2016 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

A. Oh and A. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL Workshop on Conversational Systems*, pages 27–32.

A. Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, 16(3):435–455.

E. Reiter, R. Dale, and Z. Feng. 2000. *Building natural language generation systems*, volume 33. MIT Press.

E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1):137–169.

L. Shang, Z. Lu, and H. Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proc. ACL*, pages 1105–1112.

R. Turner, S. Sripada, and E. Reiter. 2010. Generating approximate geographic descriptions. In *Empirical methods in natural language generation*, pages 121–140. Springer.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

T. Wen, M. Gasic, N. Mrkšić, P. Su, D. Vandyke, and S. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September. Association for Computational Linguistics.

Y. W. Wong and R. J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of The 32nd International Conference on Machine Learning*, volume 37, July.

K. Yao, G. Zweig, and B. Peng. 2015. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*.

What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in Web argumentation

Ivan Habernal[†] and **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

This article tackles a new challenging task in computational argumentation. Given a pair of two arguments to a certain controversial topic, we aim to directly assess qualitative properties of the arguments in order to explain why one argument is more convincing than the other one. We approach this task in a fully empirical manner by annotating 26k explanations written in natural language. These explanations describe convincingness of arguments in the given argument pair, such as their strengths or flaws. We create a new crowd-sourced corpus containing 9,111 argument pairs, multi-labeled with 17 classes, which was cleaned and curated by employing several strict quality measures. We propose two tasks on this data set, namely (1) predicting the full label distribution and (2) classifying types of flaws in less convincing arguments. Our experiments with feature-rich SVM learners and Bidirectional LSTM neural networks with convolution and attention mechanism reveal that such a novel fine-grained analysis of Web argument convincingness is a very challenging task. We release the new corpus *UKPConvArg2* and the accompanying software under permissive licenses to the research community.

1 Introduction

People engage in argumentation in various contexts, both online and in the real life. Existing definitions of argumentation do not solely focus on giving reasons and laying out a logical framework of premises and conclusions, but also highlight its social purpose which is to convince or to persuade (O'Keefe, 2011; van Eemeren et al., 2014; Blair, 2011). Assessing the quality and strength of perceived arguments therefore plays an inherent role in argumentative discourse. Despite strong theoretical foundations and plethora of normative theories, such as Walton's schemes and their critical questions (Walton, 1989), an ideal model of critical discussion in the pragma-dialectic view (Van Eemeren and Grootendorst, 1987), or research into fallacies (Boudry et al., 2015), assessing qualitative criteria of everyday argumentation represents a challenge for argumentation scholars and practitioners (Weltzer-Ward et al., 2009; Swanson et al., 2015; Rosenfeld and Kraus, 2015).

Addressing qualitative aspects of arguments has recently started gaining attention in the field of computational argumentation. Scoring strength of persuasive essays (Farra et al., 2015; Persing and Ng, 2015), exploring interaction in persuasive dialogues on Reddit (Tan et al., 2016), or detecting convincing arguments (Habernal and Gurevych, 2016) are among recent attempts to tackle the quality of argumentation. However, these approaches are holistic and do not necessarily explain why a given argument is strong or convincing.

We asked the following research questions. First, can we assess what makes an argument convincing in a purely empirical fashion as opposite to theoretical normative approaches? Second, to what extent can the problem be tackled by computational models? To address these questions, we exploit our recently introduced *UKPConvArg1* corpus (Habernal and Gurevych, 2016). This data set consists of 11,650 *argument pairs* – two arguments with the

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1214–1223,
Austin, Texas, November 1-5, 2016.

Prompt: Should physical education be mandatory in schools? **Stance:** Yes!

Argument 1	Argument 2
PE should be compulsory because it keeps us constantly fit and healthy. If you really dislike sports, then you can quit it when you're an adult. But when you're a kid, the best thing for you to do is study, play and exercise. If you prefer to be lazy and lie on the couch all day then you are most likely to get sick and unfit. Besides, PE helps kids be better at teamwork.	physical education should be mandatory cuhz 112,000 people have died in the year 2011 so far and it's because of the lack of physical activity and people are becoming obese!!!!

A1 is more convincing than A2, because:

- "A1 is more intelligently written and makes some good points (teamwork, for example). A2 used 'cuhz' and I was done reading because that sounds stupid."
- "A1 gives more reasons and goes into detail, A2 only has one fact"
- "A1 makes several compelling points. A2 uses poor spelling and grammar."

Figure 1: An annotated argument pair from the *UKPConvArg* corpus with three *reasons* explaining the decision about convincingness (ID `arg54258_arg202285`).

same standpoint to the given topic, annotated with a binary relation describing which argument from the pair is more convincing. Each pair also contains several *reasons* written in natural language explaining which properties of the arguments influence their convincingness. An example of such an argument pair is shown in Figure 1.

We use these natural language reasons as a proxy to assess qualitative properties of the arguments in each argument pair. Our main contributions are: (1) We propose empirically inspired labels of quality properties of Web arguments and design a hierarchical annotation scheme. (2) We create a new large crowd-sourced benchmark data set containing 9,111 argument pairs multi-labeled with 17 categories which is improved by local and global filtering techniques. (3) We experiment with several computational models, both traditional and neural network-based, and evaluate their performance quantitatively and qualitatively.

The newly created data set *UKPConvArg2* is available under CC-BY-SA license along with the experimental software for full reproducibility at GitHub.[1]

2 Related Work

The growing field of computational argumentation has been traditionally devoted to structural tasks, such as argument component detection and classification (Habernal and Gurevych, 2017; Habernal and Gurevych, 2015), argument structure parsing (Peldszus and Stede, 2015; Stab and Gurevych, 2014), or argument schema classification (Lawrence and Reed, 2015), leaving the issues of argument evaluation or quality assessment as an open future work.

There are only few attempts to tackle the qualitative aspects of arguments, especially in the Web discourse. Park and Cardie (2014) classified propositions in Web arguments into four classes with respect to their level of verifiability. Focusing on convincingness of Web arguments, Habernal and Gurevych (2016) annotated 16k pairs of arguments with a binary relation "is more convincing" and also elicited explanation for the annotators' decisions.

Recently, research in persuasive essay scoring has started combining holistic approaches based on rubrics for several dimensions typical to this genre with explicit argument detection. Persing and Ng (2015) manually labeled 1,000 student persuasive essays with a single score on the 1–4 scale and trained a regression predictor with a rich feature set using LIBSVM. Among traditional features (such as POS or semantic frames), an argument structure parser by Stab and Gurevych (2014) was employed. Farra et al. (2015) also deal with essay scoring but rather then tackling the argument structure, they focus on methods for detecting opinion expressions. Persuasive essays however represent a genre with a rather strict qualitative and formal requirements (as taught in curricula) and substantially differ from online argumentation.

Argument evaluation belongs to the central research topics among argumentation scholars (Toul-

[1] `https://github.com/UKPLab/emnlp2016-empirical-convincingness`

min, 2003; Walton et al., 2008; Van Eemeren and Grootendorst, 1987). Yet treatment of assessing argumentation quality, persuasiveness, or convincingness is traditionally based on evaluating relevance, sufficiency or acceptability of premises (Govier, 2010; Johnson and Blair, 2006) or categorizing fallacies (Hamblin, 1970; Tindale, 2007). However, the nature of these normative approaches causes a gap between the 'ideal' models and empirically encountered real-world arguments, such as those on the Web (van Eemeren et al., 2014; Walton, 2012).

Regarding the methodology utilized later in this paper, deep (recursive) neural networks have gained extreme popularity in NLP in recent years. Long Short-Term Memory networks (LSTM) with Attention mechanism have been applied on textual entailment (Rocktäschel et al., 2016), Question-Answering (Golub and He, 2016), or source-code summarization (Allamanis et al., 2016).

3 Data

As our source data set, we took the publicly available *UKPConvArg1* corpus.[2] It is based on arguments originated from 16 debates from Web debate platforms `createdebate.com` and `convinceme.net`, each debate has two sides (usually pro and con). Arguments from each of the 32 debate sides are connected into a set of argument pairs, and each argument pair is annotated with a binary relation (argument A is more/less convincing than argument B), resulting in total into 11,650 argument pairs. Annotations performed by Habernal and Gurevych (2016) also contain several *reasons* written by crowd-workers that explain why a particular argument is more or less convincing; see an example in Figure 1.

As these *reasons* were written in an uncontrolled setting, they naturally reflect the main properties of argument quality in a downstream task, which is to decide which argument from a pair is more convincing. It differs from scoring arguments in isolation, which is inherently harder not only due to subjectivity in argument "strength" decision but also because of possible annotator's prior bias (Habernal and Gurevych, 2016). Assessing an argument in context helps to emphasize its main flaws or strengths. This approach is also known as *knowledge elicitation* – acquiring appropriate information from experts by asking "why?" (Reed and Rowe, 2004).

We therefore used the *reasons* as a proxy for developing a scheme for labeling argument quality attributes. This was done in a purely bottom-up empirical manner, as opposed to using 'standard' evaluation criteria known from argumentation literature (Johnson and Blair, 2006; Schiappa and Nordin, 2013). In particular, we split all *reasons* into several *reason units* by simple preprocessing (splitting using Stanford CoreNLP (Manning et al., 2014), segmentation into Elementary Discourse Units by RST tools (Surdeanu et al., 2015)) and identified the referenced arguments (A1 or A2) by pattern matching and dependency parsing. For example, each *reason* from Figure 1 would be transformed into two *reason units*.[3] Overall, we obtained about 70k *reason units* from the entire *UKPConvArg1* corpus.

3.1 Annotation scheme

In order to develop a code book for assigning a label to each *reason unit*, we ran several pilot expert annotation studies (each with 200-300 *reason units*). Having a set of $\approx$ 25 distinct labels, we ran two larger studies on Amazon Mechanical Turk (AMT), each with 500 reason units and 10 workers. The workers were split into two groups; we then estimated gold labels for each group using MACE (Hovy et al., 2013) and compared both groups' results in order to find systematic discrepancies. Finally, we ended up with a set of 19 distinct labels (classes). As the number of classes is too big for non-expert crowd workers, we developed a hierarchical annotation process guided by questions that narrow down the final class decision. The scheme is depicted in Figure 2.[4] Workers were shown only the *reason units* without seeing the original arguments.

[2] `https://github.com/UKPLab/acl2016-convincing-arguments`

[3] We picked this example for its simplicity, in reality the texts are much more fuzzy.

[4] It might seem that some labels are missing, such as C8-2 and C8-3; these belong to those removed during the pilot studies.

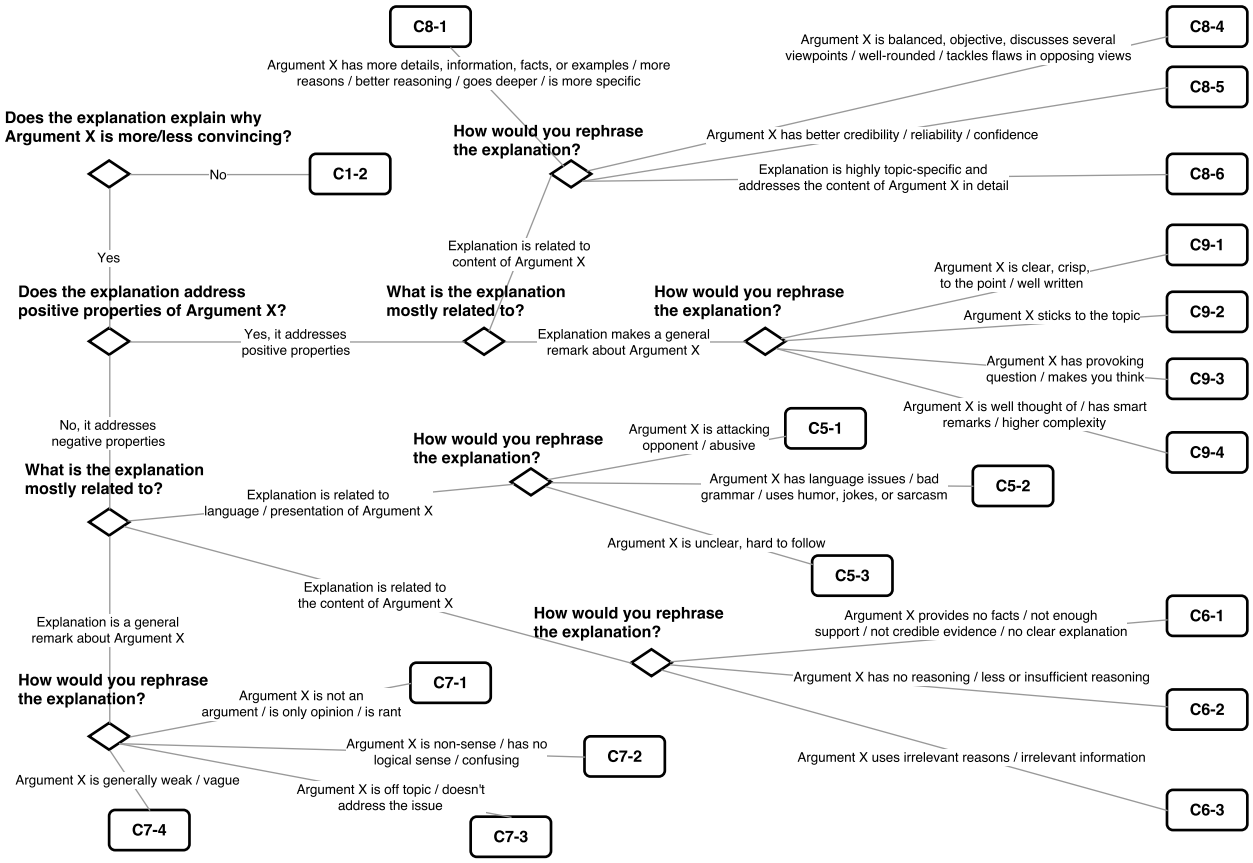


Figure 2: Decision tree-based annotation schema for labeling reason units using Mechanical Turk. *CX-Y* represent the final labels.

3.2 Annotation

We sampled 26,000 unique reason units ordered by the original author competence provided as part of the *UKPConvArg* corpus. We expected that workers with higher competence tend to write better reasons for their explanations. Using the previously introduced scheme, 776 AMT workers annotated the batch during two weeks; we required assignments from 5 workers for a single item. We employed MACE (Hovy et al., 2013) for gold label and worker competence estimation with 95% threshold to ignore the less confident labels. Several workers were rejected based on their low computed competence and other criteria, such as too short submission times.

3.3 Data cleaning

We performed several cleaning procedures to increase quality and consistency of the annotated data (apart from initial MACE filtering already explained above).

Local cleaning First, we removed 3,859 *reason units* annotated either with C1-2 (”not an explanation”) and C8-6 (”too topic-specific”, which usually paraphrases some details from the related argument and is not general enough). In the next step, we removed *reason units* with wrong polarity. In particular, all *reason units* labeled with C8-* or C9-* should refer to the more convincing argument in the argument pair (as they describe positive properties), whereas all reasons with labels C5-*, C6-*, and C7-* should refer to the less convincing argument. The target arguments for *reason units* were known from the heuristic preprocessing (see above); in this step 2,455 units were removed.

Global cleaning Since the argument pairs from one debate can be projected into an argument graph (Habernal and Gurevych, 2016), we utilized this ‘global’ context for further consistency cleaning.

Suppose we have two argument pairs, $P_1(A \rightarrow B)$ and $P_2(B \rightarrow C)$ (where $\rightarrow$ means “is more convincing than”). Let $P_1(R_B)$ be reason unit targeting

B in argument pair P_1 and similarly $P_2(R_B)$ reason unit targeting B in argument pair P_2. In other words, two reason units target the same argument in two different argument pairs (in one of them the argument is more convincing while in the other pair it is less convincing). There might then exist contradicting combination of classes for $P_1(R_B)$ and $P_2(R_B)$. For example classes C9-2 and C7-3 are contradicting, as the same argument cannot be both "on the topic" and "off-topic" at the same time.

When such a conflict between two reason units occurred, we selected the reason with a higher score using the following formula:

$$w_W * \sigma \left(\sum_{A=G} w_A - \lambda \sum_{A \neq G} w_A \right) \quad (1)$$

where w_W is the competence of the original author of the reason unit (originated from the *UKP-ConvArg* corpus), $A = G$ are crowdsourced assignments for a single reason unit that match the final predicted gold label, $A \neq G$ are assignments that differ from the final predicted gold label, w_A is the competence of worker for assignment A, λ is a penalty for non-gold labels, and σ is the sigmoid function to squeeze the score between 0 and 1.

We found 25 types of global contradictions between labels for reason units and used them for cleaning the data; in total 3,790 *reason units* were removed in this step. After all cleaning procedures, annotations from *reason units* were mapped back to *argument pairs*, resulting into a multi-label annotation of one or both arguments from the given pair. In total 9,111 pairs from the *UKPConvArg* corpus were annotated.

For example, the final annotations of argument pair shown in Figure 1 contain four labels – C8-1 (as the more convincing argument *"has more details, information, facts, or examples / more reasons / better reasoning / goes deeper / is more specific"*), C9-3 (as the more convincing argument *"has provoking question / makes you think"*), C5-2 (as the less convincing argument *"has language issues / bad grammar /..."*), and C6-1 (as the less convincing argument *"provides not enough support / ..."*). Only four of six *reason units* for this argument pair were annotated because of the competence score of their authors.

# of labels/pair	# of pairs
1	4,584
2	2,959
3	1,162
4	330
5	68
6	8
Total	9,111

Table 1: Number of annotated labels per argument pairs.

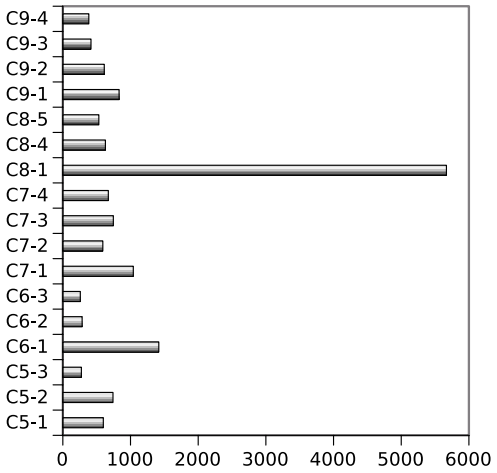


Figure 3: Distribution of labels in the annotated argument pairs. Consult Figure 2 for label descriptions.

Table 1 shows number of labels per argument pairs; about a half of the argument pairs have only one label. Figure 3 shows distribution of label in the entire data set which is heavily skewed towards C8-1 label. This is not surprising, as this label was used for reason units pointing out that the more convincing argument provided more reasons, details, information or better reasoning – a feature inherent to argumentation seen as *giving reasons* (Freeley and Steinberg, 2008).

3.4 Data validation

Since the qualitative attributes of arguments were annotated indirectly by labeling their corresponding reason units without seeing the original arguments, we wanted to validate correctness of this approach. We designed a validation study, in which workers were shown the original argument pair and two sets of labels. The first set contained the true labels as annotated previously, while we randomly replaced few labels in the second set. The goal was then to decide which set of labels better explains that argument A is

more convincing than argument B. For example, for the argument pair from Figure 1, one set of shown labels would be {C8-1, C9-3, C5-2, C6-1} (the correct set) while the other 'distracting' set would be {C8-1, C9-3, C5-1, C7-3} .

We randomly sampled 500 argument pairs and collected 9 assignments per pair on AMT; we again used MACE with 95% threshold. Accuracy of workers on 235 argument pairs achieved 82%. We can thus conclude that workers tend to prefer explanations based on labels from the *reason units* and using the annotation process presented in this section is reliable. Total costs of the annotations including pilot studies, bonuses, and data validation were USD 3,300.

4 Experiments

We propose two experiments, both performed in 16-fold cross-domain validation. In each fold, argument pairs from 15 debates are used and the remaining one is used for testing. In both experiments, it is assumed that the more convincing argument in a pair is known and we concatenate (using a particular delimiter) both arguments such that the more convincing argument comes first.

4.1 Predicting full multi-label distribution

This experiment is a multi-label classification. Given an argument pair annotated with several labels, the goal is to predict all these labels.

We use two deep learning models. Our first model, Bidirectional Long Short-Term Memory (BLSTM) network contains two LSTM blocks (forward and backward), each with 64 hidden units on the output. The output is concatenated into a single vector and pushed through sigmoid layer with 17 output units (corresponding to 17 labels). We use cross entropy loss function in order to minimize distance of label distributions in training and test data (Nam et al., 2014). In the input layer, we rely on pre-trained word embeddings from Glove (Pennington et al., 2014) whose weights are updated during training the network.

The second models is BLSTM extended with an attention mechanism (Rocktäschel et al., 2016; Golub and He, 2016) combined with convolution layers over the input. In particular, the input embedding layer is convoluted using 4 different convolution sizes (2, 3, 5, 7), each with 1,000 randomly initialized weight vectors. Then we perform max-over-time pooling and concatenate the output into a single vector. This vector is used as the attention module in BLSTM.

	BLSTM		BLSTM/CNN/ATT	
Debate	**H-loss**	**one-E**	**H-loss**	**one-E**
Ban plastic water bottles?	0.092	0.283	0.090	0.305
Christianity or Atheism	0.105	0.212	0.105	0.218
Evolution vs. Creation	0.093	0.196	0.094	0.234
Firefox vs. Internet Explorer	0.080	0.312	0.078	0.345
Gay marriage: right or wrong?	0.095	0.243	0.094	0.270
Should parents use spanking?	0.082	0.312	0.083	0.344
If your spouse committed murder...	0.094	0.297	0.094	0.272
India has the potential to lead the world	0.088	0.294	0.086	0.322
Is it better to have a lousy father or to be fatherless?	0.086	0.367	0.085	0.381
Is porn wrong?	0.098	0.278	0.100	0.270
Is the school uniform a good or bad idea?	0.081	0.279	0.077	0.406
Pro-choice vs. Pro-life	0.095	0.218	0.098	0.218
Should Physical Education be mandatory?	0.095	0.273	0.095	0.277
TV is better than books	0.091	0.265	0.087	0.300
Personal pursuit or common good?	0.095	0.328	0.094	0.343
W. Farquhar ought to be honored...	0.054	0.528	0.052	0.570
Average	0.089	0.293	0.088	0.317

Table 2: Results of multi-label classification from Experiment 1. Hamming-loss and One-Error are shown for two systems – Bidirectional LSTM and Bidirectional LSTM with Convolution and Attention.

We evaluate the system using two widely used metrics in multi-label classification. First, Hamming loss is the average per-item per-class total error; the smaller the better (Zhang and Zhou, 2007). Second, we report One-error (Sokolova and Lapalme, 2009) which corresponds to the error of the predicted label with highest probability; the smaller the better. We do not report other metrics (such as Area Under PRC-curves, MAP, or cover) as they require tuning a threshold parameter, see a survey by Zhang and Zhou (2014).

Results from Table 2 do not show significant differences between the two models. Putting the one-error numbers into human performance context can be done only indirectly, as the data validation pre-

sented in Section 3.4 had a different set-up. Here we can see that the error rate of the most confident predicted label is about 30%, while human performed similarly by choosing from a two different label sets in a binary settings, so their task was inherently harder.

Error analysis and discussion We examined outputs from the label distribution prediction for BLSTM/ATT/CNN. It turns out that the output layer leans toward predicting the dominant label *C8-1*, while prediction of other labels is seldom. We suspect two causes, first, the highly skewed distribution of labels (see Figure 3) and, second, insufficient training data sizes where 13 classes have less than 1k training examples (while Goodfellow et al. (2016) recommend at least 5k instances per class).

Although multi-label classification may be viewed as a set of binary classification tasks that decides for each label independently (and thus allows for employing other 'standard' classifiers such as SVM), this so-called binary relevance approach ignores dependencies between the labels. That is why we focused directly on deep-learning methods, as they are capable of learning and predicting a full label distribution (Nam et al., 2014).

4.2 Predicting flaws in less convincing arguments

In the second experiment, we focus on predicting flaws in arguments using coarse-grained labels. While this task makes several simplifications in the labeling, it still provides meaningful insights into argument quality assessment. For this purpose, we use only argument pairs where the less convincing argument is labeled with a single label (no multi-label classification). Second, we merged all labels from categories *C5-* C6-* C7-** into three classes corresponding to their parent nodes in the annotation decision schema from Figure 2. Table 3 shows distribution of the gold data for this task with explanation of the labels. It is worth noting that predicting flaws in the less convincing argument is still context-dependent and requires the entire argument pair because some of the quality labels are relative to the more convincing argument (such as "less reasoning" or "not enough support").

For this experiment, we modified the output layer of the neural models from the previous experiment. The non-linear output function is *softmax* and we train the networks using categorical cross-entropy loss. We also add another baseline model that employs SVM with RBF kernel[5] and a rich set of linguistically motivated features, similarly to (Habernal and Gurevych, 2016). The feature set includes *uni- and bi-gram presence*, ratio of *adjective and adverb endings* that may signalize neuroticism (Corney et al., 2002), *contextuality measure* (Heylighen and Dewaele, 2002), *dependency tree depth*, ratio of *exclamation* or *quotation* marks, ratio of *modal verbs*, counts of several *named entity types*, ratio of *past* vs. *future* tense verbs, *POS* n-grams, presence of dependency tree *production rules*, seven different *readability measures* (e.g., *Ari* (Senter and Smith, 1967), *Coleman-Liau* (Coleman and Liau, 1975), *Flesch* (Flesch, 1948), and others), five *sentiment scores* (from very negative to very positive) (Socher et al., 2013), *spell-checking* using standard Unix `words`, ratio of *superlatives*, and some *surface* features such as sentence lengths, longer words count, etc.[6] It results into a sparse 60k-dimensional feature vector space.

Label	Instances	Description
C5	856	Language and presentation issues
C6	1,203	Reasoning and factuality issues
C7	1,651	Off-topic, non-argument, nonsense
Total	3,710	

Table 3: Gold data distribution for the second experiment. Argument pairs with a single label for the less convincing argument.

Results in Table 4 suggest that the SVM-RBF baseline system performs poorly and its results are on par with a majority class baseline (not reported in detail). Both deep learning models significantly outperform the baseline, yielding Macro-F_1 score about 0.35. The attention-based model performs better than simple BLSTM in two classes (C5 and C6), but the overall Macro-F_1 score is not significantly better.

[5] We used LISBVM (Chang and Lin, 2011) with the default hyper-parameters. As Fernández-Delgado et al. (2014) show, SVM with gaussian kernels is a reasonable best choice on average.

[6] Detailed explanation of the features can be found directly in the attached source codes.

	Class C5			Class C6			Class C7				
Model	P	R	F_1	P	R	F_1	P	R	F_1	**M-F_1**	**C.I.**
SVM-RBF	0.351	0.023	0.044	0.394	0.083	0.137	0.446	0.918	0.600	0.260	0.014
BLSTM	0.265	0.600	0.368	0.376	0.229	0.285	0.479	0.301	0.370	0.341	0.015
BLSTM/ATT/CNN	0.270	0.625	0.378	0.421	0.247	0.311	0.484	0.301	0.371	**0.353**	0.015

Table 4: Results for experiment 2. P = precision, R = recall, M-F_1 = macro F_1, C.I. = confidence interval at 0.95. Both *BLSTM* and *BLSTM/ATT/CNN* are significantly better than *SVM-RBF* ($p < 0.05$, exact Liddell's test).

Error analysis We manually examined several dozens of predictions where the BLSTM model failed but the BLSTM/ATT/CNN model was correct in order to reveal some phenomena that the system is capable to cope with. First, the BLSTM/ATT/CNN model started catching some purely abusive, sarcastic, and attacking arguments. Also, the language/grammar issues were revealed in many cases, as well as using slang in arguments.

Examining predictions in which both systems failed reveal some fundamental limitations of the current purely data-driven computational approach. While the problem of not catching off-topic arguments can be probably modeled by incorporating the debate description or some sort of debate topic model into the attention vector, the more common issue of non-sense arguments or fallacious arguments (which seem like actual arguments on the first view) needs much deeper understanding of real-world knowledge, logic, and reasoning.

5 Conclusion

This paper presented a novel task in the field of computational argumentation, namely empirical assessment of reasons for argument convincingness. We created a new large benchmark data set by utilizing a new annotation scheme and several filtering strategies for crowdsourced data. Then we tackled two challenging tasks, namely multi-label classification of argument pairs in order to reveal qualitative properties of the arguments, and predicting flaws in the less convincing argument from the given argument pair. We performed all evaluations in a cross-domain scenario and experimented with feature-rich SVM and two state-of-the-art neural network models. The results are promising but show that the task is inherently complex as it requires deep reasoning about the presented arguments that goes beyond capabilities of the current computational models. By releasing the *UKPConvArg2* data and code to the community, we believe more progress can be made in this direction in the near future.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant Nº I/82806, by the German Institute for Educational Research (DIPF), by the German Research Foundation (DFG) via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), by the GRK 1994/1 AIPHES (DFG), by the ArguAna Project GU 798/20-1 (DFG), and by Amazon Web Services in Education Grant award. Lastly, we would like to thank the anonymous reviewers for their valuable feedback.

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, pages 2091–2100, New York City, NY, June.

J. Anthony Blair. 2011. Argumentation as rational persuasion. *Argumentation*, 26(1):71–81.

Maarten Boudry, Fabio Paglieri, and Massimo Pigliucci. 2015. The Fake, the Flimsy, and the Fallacious: Demarcating Arguments in Real Life. *Argumentation*, 29(4):431–456.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.

Meri Coleman and T. L. Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60:283–284.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of the 18th An-*

nual Computer Security Applications Conference (ACSAC02), pages 282–289.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74, Denver, Colorado, June. Association for Computational Linguistics.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15:3133–3181.

Rudolf Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32:221–233.

Austin J. Freeley and David L. Steinberg. 2008. *Argumentation and Debate*. Cengage Learning, Stamford, CT, USA, 12th edition.

David Golub and Xiaodong He. 2016. Character-Level Question Answering with Attention. In *arXiv preprint*. http://arxiv.org/abs/1604.00727.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep learning. Book in preparation for MIT Press.

Trudy Govier. 2010. *A Practical Study of Argument*. Wadsworth, Cengage Learning, 7th edition.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137, Lisbon, Portugal, September. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics*, 43(1). In press. Preprint: http://arxiv.org/abs/1601.02403.

Charles L. Hamblin. 1970. *Fallacies*. Methuen, London, UK.

Francis Heylighen and Jean-Marc Dewaele. 2002. Variation in the contextuality of language: An empirical measure. *Foundations of Science*, 7(3):293–340.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of NAACL-HLT 2013*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Ralph H. Johnson and Anthony J. Blair. 2006. *Logical Self-Defense*. International Debate Education Association.

John Lawrence and Chris Reed. 2015. Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 127–136, Denver, CO, June. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-Scale Multi-label Text Classification – Revisiting Neural Networks. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, volume 8725 LNCS, pages 437–452, Nancy, France. Springer Berlin / Heidelberg.

Daniel J. O'Keefe. 2011. Conviction, persuasion, and argumentation: Untangling the ends and means of influence. *Argumentation*, 26(1):19–32.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal, September. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552, Beijing, China. Association for Computational Linguistics.

Chris Reed and Glenn Rowe. 2004. Araucaria: software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, 13(04):961–979, dec.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 2016 International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1509.06664.

Ariel Rosenfeld and Sarit Kraus. 2015. Providing arguments in discussions based on the prediction of human argumentative behavior. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1320–1327.

Edward Schiappa and John P. Nordin. 2013. *Argumentation: Keeping Faith with Reason*. Pearson UK, 1st edition.

J. R. Senter and E. A. Smith. 1967. Automated readability index. Technical report AMRL-TR-66-220, Aerospace Medical Research Laboratories, Ohio.

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

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, Denver, Colorado, June. Association for Computational Linguistics.

Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument Mining: Extracting Arguments from Online Dialogue. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 217–226, Prague, Czech Republic. Association for Computational Linguistics.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions. In *Proceedings of the 25th International Conference on World Wide Web*, pages 613–624, Montreal, CA, Februar. International World Wide Web Conferences Steering Committee.

Christopher W. Tindale. 2007. *Fallacies and Argument Appraisal*. Cambridge University Press, New York, NY, USA, critical reasoning and argumentation edition.

Stephen E. Toulmin. 2003. *The Uses of Argument, Updated Edition*. Cambridge University Press, New York.

Frans H. Van Eemeren and Rob Grootendorst. 1987. Fallacies in pragma-dialectical perspective. *Argumentation*, 1(3):283–301.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer, Berlin/Heidelberg.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Douglas N. Walton. 1989. *Informal Logic: A Handbook for Critical Argument*. Cambridge University Press.

Douglas Walton. 2012. Using argumentation schemes for argument extraction: A bottom-up method. *International Journal of Cognitive Informatics and Natural Intelligence*, 6(3):33–61.

Lisa Weltzer-Ward, Beate Baltes, and Laura Knight Lynn. 2009. Assessing quality of critical thought in online discussion. *Campus-Wide Information Systems*, 26(3):168–177.

Min Ling Zhang and Zhi Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Recognizing Implicit Discourse Relations via Repeated Reading: Neural Networks with Multi-Level Attention

Yang Liu[1,2], **Sujian Li**[1]
[1] Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2] ILCC, School of Informatics, University of Edinburgh, United Kingdom
{cs-ly, lisujian}@pku.edu.cn

Abstract

Recognizing implicit discourse relations is a challenging but important task in the field of Natural Language Processing. For such a complex text processing task, different from previous studies, we argue that it is necessary to repeatedly read the arguments and dynamically exploit the efficient features useful for recognizing discourse relations. To mimic the repeated reading strategy, we propose the neural networks with multi-level attention (NNMA), combining the attention mechanism and external memories to gradually fix the attention on some specific words helpful to judging the discourse relations. Experiments on the PDTB dataset show that our proposed method achieves the state-of-art results. The visualization of the attention weights also illustrates the progress that our model observes the arguments on each level and progressively locates the important words.

1 Introduction

Discourse relations (e.g., contrast and causality) support a set of sentences to form a coherent text. Automatically recognizing discourse relations can help many downstream tasks such as question answering and automatic summarization. Despite great progress in classifying explicit discourse relations where the discourse connectives (e.g., "because", "but") explicitly exist in the text, implicit discourse relation recognition remains a challenge due to the absence of discourse connectives. Previous research mainly focus on exploring various kinds of efficient features and machine learning models to classify the implicit discourse relations (Soricut and Marcu, 2003; Baldridge and Lascarides, 2005; Subba and Di Eugenio, 2009; Hernault et al., 2010; Pitler et al., 2009; Joty et al., 2012). To some extent, these methods simulate the single-pass reading process that a person quickly skim the text through one-pass reading and directly collect important clues for understanding the text. Although single-pass reading plays a crucial role when we just want the general meaning and do not necessarily need to understand every single point of the text, it is not enough for tackling tasks that need a deep analysis of the text. In contrast with single-pass reading, repeated reading involves the process where learners repeatedly read the text in detail with specific learning aims, and has the potential to improve readers' reading fluency and comprehension of the text (National Institute of Child Health and Human Development, 2000; LaBerge and Samuels, 1974). Therefore, for the task of discourse parsing, repeated reading is necessary, as it is difficult to generalize which words are really useful on the first try and efficient features should be dynamically exploited through several passes of reading .

Now, let us check one real example to elaborate the necessity of using repeated reading in discourse parsing.

Arg-1 : the use of 900 toll numbers has been *expanding rapidly* in recent years

Arg-2 : for a while, high-cost pornography lines and services that tempt children to dial (and redial) movie or music information *earned the service a somewhat sleazy image*

(Comparison - wsj_2100)

To identify the "**Comparison**" relation between

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1224–1233,
Austin, Texas, November 1-5, 2016.

the two arguments *Arg-1* and *Arg-2*, the most crucial clues mainly lie in some content, like *"expanding rapidly"* in *Arg-1* and *"earned the service a somewhat sleazy image"* in *Arg-2*, since there exists a contrast between the semantic meanings of these two text spans. However, it is difficult to obtain sufficient information for pinpointing these words through scanning the argument pair left to right in one pass. In such case, we follow the repeated reading strategy, where we obtain the general meaning through reading the arguments for the first time, re-read them later and gradually pay close attention to the key content.

Recently, some approaches simulating repeated reading have witnessed their success in different tasks. These models mostly combine the attention mechanism that has been originally designed to solve the alignment problem in machine translation (Bahdanau et al., 2014) and the external memory which can be read and written when processing the objects (Sukhbaatar et al., 2015). For example, Kumar et al. (2015) drew attention to specific facts of the input sequence and processed the sequence via multiple hops to generate an answer. In computation vision, Yang et al. (2015) pointed out that repeatedly giving attention to different regions of an image could gradually lead to more precise image representations.

Inspired by these recent work, for discourse parsing, we propose a model that aims to repeatedly read an argument pair and gradually focus on more fine-grained parts after grasping the global information. Specifically, we design the Neural Networks with Multi-Level Attention (NNMA) consisting of one general level and several attention levels. In the general level, we capture the general representations of each argument based on two bidirectional long short-term memory (LSTM) models. For each attention level, NNMA generates a weight vector over the argument pair to locate the important parts related to the discourse relation. And an external short-term memory is designed to store the information exploited in previous levels and help update the argument representations. We stack this structure in a recurrent manner, mimicking the process of reading the arguments multiple times. Finally, we use the representation output from the highest attention level to identify the discourse relation. Experiments on the PDTB dataset show that our proposed model achieves the state-of-art results.

2 Repeated Reading Neural Network with Multi-Level Attention

In this section, we describe how we use the neural networks with multi-level attention to repeatedly read the argument pairs and recognize implicit discourse relations.

First, we get the general understanding of the arguments through skimming them. To implement this, we adopt the bidirectional Long-Short Term Memory Neural Network (bi-LSTM) to model each argument, as bi-LSTM is good at modeling over a sequence of words and can represent each word with consideration of more contextual information. Then, several attention levels are designed to simulate the subsequent multiple passes of reading. On each attention level, an external short-term memory is used to store what has been learned from previous passes and guide which words should be focused on. To pinpoint the useful parts of the arguments, the attention mechanism is used to predict a probability distribution over each word, indicating to what degree each word should be concerned. The overall architecture of our model is shown in Figure 1. For clarity, we only illustrate two attention levels in the figure. It is noted that we can easily extend our model to more attention levels.

2.1 Representing Arguments with LSTM

The Long-Short Term Memory (LSTM) Neural Network is a variant of the Recurrent Neural Network which is usually used for modeling a sequence. In our model, we adopt two LSTM neural networks to respectively model the two arguments: the left argument *Arg-1* and the right argument *Arg-2*.

First of all, we associate each word w in our vocabulary with a vector representation $\boldsymbol{x}_w \in \mathbb{R}^{D_e}$. Here we adopt the pre-trained vectors provided by GloVe (Pennington et al., 2014). Since an argument can be viewed as a sequence of word vectors, let $\boldsymbol{x}_i^1$ ($\boldsymbol{x}_i^2$) be the i-th word vector in argument *Arg-1* (*Arg-*

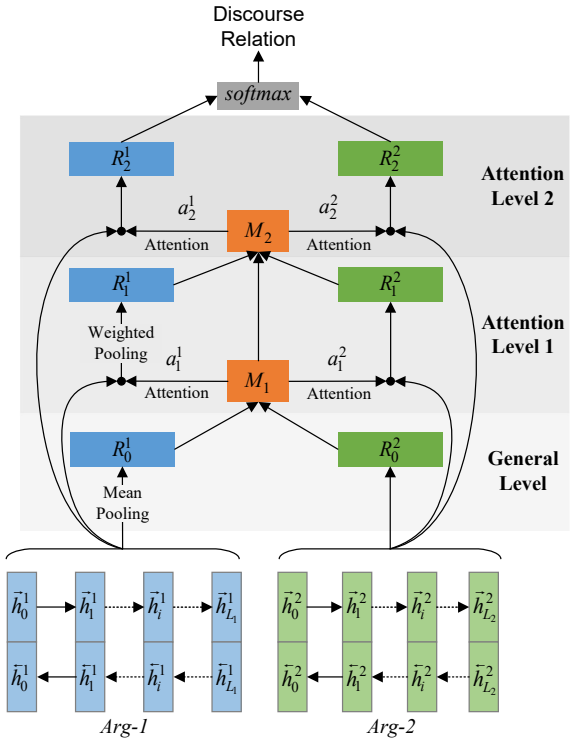


Figure 1: Neural Network with Multi-Level Attention. (Two attention levels are given here.)

2) and the two arguments can be represented as,

$$Arg\text{-}1 : [\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, \cdots, \boldsymbol{x}_{L_1}^1]$$
$$Arg\text{-}2 : [\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \cdots, \boldsymbol{x}_{L_2}^2]$$

where *Arg-1* has L_1 words and *Arg-2* has L_2 words.

To model the two arguments, we briefly introduce the working process how the LSTM neural networks model a sequence of words. For the i-th time step, the model reads the i-th word $\boldsymbol{x}_i$ as the input and updates the output vector $\boldsymbol{h}_i$ as follows (Zaremba and Sutskever, 2014).

$$\boldsymbol{i}_i = sigmoid(\boldsymbol{W}_i[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_i) \quad (1)$$
$$\boldsymbol{f}_i = sigmoid(\boldsymbol{W}_f[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_f) \quad (2)$$
$$\boldsymbol{o}_i = sigmoid(\boldsymbol{W}_o[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_o) \quad (3)$$
$$\tilde{\boldsymbol{c}}_i = tanh(\boldsymbol{W}_c[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_c) \quad (4)$$
$$\boldsymbol{c}_i = \boldsymbol{i}_i * \tilde{\boldsymbol{c}}_i + \boldsymbol{f}_i * \boldsymbol{c}_{i-1} \quad (5)$$
$$\boldsymbol{h}_i = \boldsymbol{o}_i * tanh(\boldsymbol{c}_i) \quad (6)$$

where $[\ ]$ means the concatenation operation of several vectors. $\boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}$ and $\boldsymbol{c}$ denote the input gate, forget gate, output gate and memory cell respectively in the LSTM architecture. The input gate $\boldsymbol{i}$ determines how much the input $\boldsymbol{x}_i$ updates the memory cell. The output gate $\boldsymbol{o}$ controls how much the memory cell influences the output. The forget gate $\boldsymbol{f}$ controls how the past memory $\boldsymbol{c}_{i-1}$ affects the current state. $\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_o, \boldsymbol{W}_c, \boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o, \boldsymbol{b}_c$ are the network parameters.

Referring to the work of Wang and Nyberg (2015), we implement the bidirectional version of LSTM neural network to model the argument sequence. Besides processing the sequence in the forward direction, the bidirectional LSTM (bi-LSTM) neural network also processes it in the reverse direction. As shown in Figure 1, using two bi-LSTM neural networks, we can obtain $\boldsymbol{h}_i^1 = [\overrightarrow{\boldsymbol{h}}_i^1, \overleftarrow{\boldsymbol{h}}_i^1]$ for the i-th word in *Arg-1* and $\boldsymbol{h}_i^2 = [\overrightarrow{\boldsymbol{h}}_i^2, \overleftarrow{\boldsymbol{h}}_i^2]$ for the i-th word in *Arg-2*, where $\overrightarrow{\boldsymbol{h}}_i^1, \overrightarrow{\boldsymbol{h}}_i^2 \in \mathbb{R}^d$ and $\overleftarrow{\boldsymbol{h}}_i^1, \overleftarrow{\boldsymbol{h}}_i^2 \in \mathbb{R}^d$ are the output vectors from two directions.

Next, to get the general-level representations of the arguments, we apply a mean pooling operation over the bi-LSTM outputs, and obtain two vectors $\boldsymbol{R}_0^1$ and $\boldsymbol{R}_0^2$, which can reflect the global information of the argument pair.

$$\boldsymbol{R}_0^1 = \frac{1}{L_1}\sum_{i=0}^{L_1} \boldsymbol{h}_i^1 \quad (7)$$
$$\boldsymbol{R}_0^2 = \frac{1}{L_2}\sum_{i=0}^{L_2} \boldsymbol{h}_i^2 \quad (8)$$

2.2 Tuning Attention via Repeated Reading

After obtaining the general-level representations by treating each word equally, we simulate the repeated reading and design multiple attention levels to gradually pinpoint those words particularly useful for discourse relation recognition. In each attention level, we adopt the attention mechanism to determine which words should be focused on. An external short-term memory is designed to remember what has seen in the prior levels and guide the attention tuning process in current level.

Specifically, in the first attention level, we concatenate $\boldsymbol{R}_0^1$, $\boldsymbol{R}_0^2$ and $\boldsymbol{R}_0^1$–$\boldsymbol{R}_0^2$ and apply a non-linear transformation over the concatenation to catch the general understanding of the argument pair. The use of $\boldsymbol{R}_0^1$–$\boldsymbol{R}_0^2$ takes a cue from the difference between two vector representations which

has been found explainable and meaningful in many applications (Mikolov et al., 2013). Then, we get the memory vector $\boldsymbol{M}_1 \in \mathbb{R}^{d_m}$ of the first attention level as

$$\boldsymbol{M}_1 = tanh(\boldsymbol{W}_{m,1}[\boldsymbol{R}_0^1, \boldsymbol{R}_0^2, \boldsymbol{R}_0^1 - \boldsymbol{R}_0^2]) \quad (9)$$

where $\boldsymbol{W}_{m,1} \in \mathbb{R}^{d_m \times 6d}$ is the weight matrix.

With $\boldsymbol{M}_1$ recording the general meaning of the argument pair, our model re-calculates the importance of each word. We assign each word a weight measuring to what degree our model should pay attention to it. The weights are so-called "attention" in our paper. This process is designed to simulate the process that we re-read the arguments and pay more attention to some specific words with an overall understanding derived from the first-pass reading. Formally, for *Arg-1*, we use the memory vector $\boldsymbol{M}_1$ to update the representation of each word with a non-linear transformation. According to the updated word representations $\boldsymbol{o}_1^1$, we get the attention vector $\boldsymbol{a}_1^1$.

$$\boldsymbol{h}^1 = [\boldsymbol{h}_0^1, \boldsymbol{h}_1^1, \cdots, \boldsymbol{h}_{L_1}^1] \quad (10)$$

$$\boldsymbol{o}_1^1 = tanh(\boldsymbol{W}_{a,1}^1 \boldsymbol{h}^1 + \boldsymbol{W}_{b,1}^1 (\boldsymbol{M}_1 \otimes \boldsymbol{e})) \quad (11)$$

$$\boldsymbol{a}_1^1 = softmax(\boldsymbol{W}_{s,1}^1 \boldsymbol{o}_1^1) \quad (12)$$

where $\boldsymbol{h}^1 \in \mathbb{R}^{2d \times L_1}$ is the concatenation of all LSTM output vectors of *Arg-1*. $\boldsymbol{e} \in \mathbb{R}^{L_1}$ is a vector of 1s and the $M_1 \otimes \boldsymbol{e}$ operation denotes that we repeat the vector M_1 L_1 times and generate a $d_m \times L_1$ matrix. The attention vector $\boldsymbol{a}_1^1 \in \mathbb{R}^{L_1}$ is obtained through applying a $softmax$ operation over $\boldsymbol{o}_1^1$. ${\boldsymbol{W}_{a,1}}^1 \in \mathbb{R}^{2d \times 2d}$, ${\boldsymbol{W}_{b,1}}^1 \in \mathbb{R}^{2d \times d_m}$ and ${\boldsymbol{W}_{s,1}}^1 \in \mathbb{R}^{1 \times 2d}$ are the transformation weights. It is noted that the subscripts denote the current attention level and the superscripts denote the corresponding argument. In the same way, we can get the attention vector $\boldsymbol{a}_1^2$ for *Arg-2*.

Then, according to $\boldsymbol{a}_1^1$ and $\boldsymbol{a}_1^2$, our model re-reads the arguments and get the new representations $\boldsymbol{R}_1^1$ and $\boldsymbol{R}_1^2$ for the first attention level.

$$\boldsymbol{R}_1^1 = \boldsymbol{h}^1 (\boldsymbol{a}_1^1)^T \quad (13)$$

$$\boldsymbol{R}_1^2 = \boldsymbol{h}^2 (\boldsymbol{a}_1^2)^T \quad (14)$$

Next, we iterate the "memory-attention-representation" process and design more attention levels, giving NNMA the ability to gradually infer more precise attention vectors. The processing of the second or above attention levels is slightly different from that of the first level, as we update the memory vector in a recurrent way. To formalize, for the k-th attention level ($k \geq 2$), we use the following formulae for *Arg-1*.

$$\boldsymbol{M}_k = tanh(\boldsymbol{W}_{m,k}[\boldsymbol{R}_{k-1}^1, \boldsymbol{R}_{k-1}^2, \boldsymbol{R}_{k-1}^1 - \boldsymbol{R}_{k-1}^2, \boldsymbol{M}_{k-1}]) \quad (15)$$

$$\boldsymbol{o}_k^1 = tanh(\boldsymbol{W}_{a,k}^1 \boldsymbol{h}^1 + \boldsymbol{W}_{b,k}^1 (\boldsymbol{M}_k \otimes \boldsymbol{e})) \quad (16)$$

$$\boldsymbol{a}_k^1 = softmax(\boldsymbol{W}_{s,k}^1 \boldsymbol{o}_k^1) \quad (17)$$

$$\boldsymbol{R}_k^1 = \boldsymbol{h}^1 (\boldsymbol{a}_k^1)^T \quad (18)$$

In the same way, we can computer $\boldsymbol{o}_k^2, \boldsymbol{a}_k^2$ and $\boldsymbol{R}_k^2$ for *Arg-2*.

Finally, we use the newest representation derived from the top attention level to recognize the discourse relations. Suppose there are totally K attention levels and n relation types, the predicted discourse relation distribution $\boldsymbol{P} \in \mathbb{R}^n$ is calculated as

$$\boldsymbol{P} = softmax(\boldsymbol{W}_p[\boldsymbol{R}_K^1, \boldsymbol{R}_K^2, \boldsymbol{R}_K^1 - \boldsymbol{R}_K^2] + \boldsymbol{b}_p) \quad (19)$$

where $\boldsymbol{W}_p \in \mathbb{R}^{n \times 6d}$ and $\boldsymbol{b}_p \in \mathbb{R}^n$ are the transformation weights.

2.3 Model Training

To train our model, the training objective is defined as the cross-entropy loss between the outputs of the softmax layer and the ground-truth class labels. We use stochastic gradient descent (SGD) with momentum to train the neural networks.

To avoid over-fitting, dropout operation is applied on the top feature vector before the softmax layer. Also, we use different learning rates λ and λ_e to train the neural network parameters Θ and the word embeddings Θ_e referring to (Ji and Eisenstein, 2015). λ_e is set to a small value for preventing over-fitting on this task. In the experimental part, we will introduce the setting of the hyper-parameters.

3 Experiments

3.1 Preparation

We evaluate our model on the Penn Discourse Treebank (PDTB) (Prasad et al., 2008). In our work,

we experiment on the four top-level classes in this corpus as in previous work (Rutherford and Xue, 2015). We extract all the implicit relations of PDTB, and follow the setup of (Rutherford and Xue, 2015). We split the data into a training set (Sections 2-20), development set (Sections 0-1), and test set (Section 21-22). Table 1 summarizes the statistics of the four PDTB discourse relations, i.e., Comparison, Contingency, Expansion and Temporal.

Relation	Train	Dev	Test
Comparison	1855	189	145
Contingency	3235	281	273
Expansion	6673	638	538
Temporal	582	48	55
Total	12345	1156	1011

Table 1: Statistics of Implicit Discourse Relations in PDTB.

We first convert the tokens in PDTB to lowercase. The word embeddings used for initializing the word representations are provided by GloVe (Pennington et al., 2014), and the dimension of the embeddings D_e is 50. The hyper-parameters, including the momentum δ, the two learning rates λ and λ_e, the dropout rate q, the dimension of LSTM output vector d, the dimension of memory vector d_m are all set according to the performance on the development set Due to space limitation, we do not present the details of tuning the hyper-parameters and only give their final settings as shown in Table 2.

δ	λ	λ_e	q	d	d_m
0.9	0.01	0.002	0.1	50	200

Table 2: Hyper-parameters for Neural Network with Multi-Level Attention.

To evaluate our model, we adopt two kinds of experiment settings. The first one is the four-way classification task, and the second one is the binary classification task, where we build a one-vs-other classifier for each class. For the second setting, to solve the problem of unbalanced classes in the training data, we follow the reweighting method of (Rutherford and Xue, 2015) to reweigh the training instances according to the size of each relation class. We also use visualization methods to analyze how multi-level attention helps our model.

3.2 Results

First, we design experiments to evaluate the effectiveness of attention levels and how many attention levels are appropriate. To this end, we implement a baseline model (LSTM with no attention) which directly applies the mean pooling operation over LSTM output vectors of two arguments without any attention mechanism. Then we consider different attention levels including one-level, two-level and three-level. The detailed results are shown in Table 3. For four-way classification, macro-averaged $F1$ and Accuracy are used as evaluation metrics. For binary classification, $F1$ is adopted to evaluate the performance on each class.

System	Four-way		Binary			
	$F1$	Acc.	Comp.	Cont.	Expa.	Temp.
LSTM	39.40	54.50	33.72	44.79	68.74	33.14
NNMA (one-level)	43.48	55.59	34.72	49.47	68.52	36.70
NNMA (two-level)	**46.29**	57.17	36.70	**54.48**	**70.43**	**38.84**
NNMA (three-level)	44.95	**57.57**	**39.86**	53.69	69.71	37.61

Table 3: Performances of NNMA with Different Attention Levels.

From Table 3, we can see that the basic LSTM model performs the worst. With attention levels added, our NNMA model performs much better. This confirms the observation above that one-pass reading is not enough for identifying the discourse relations. With respect to the four-way F_1 measure, using NNMA with one-level attention produces a 4% improvement over the baseline system with no attention. Adding the second attention level gives another 2.8% improvement. We perform significance test for these two improvements, and they are both significant under one-tailed t-test ($p < 0.05$). However, when adding the third attention level, the performance does not promote much and almost reaches its plateau. We can see that three-level NNMA experiences a decease in F_1 and a slight increase in Accuracy compared to two-level NNMA. The results imply that with more attention levels considered, our model may perform slightly better, but it may incur the over-fitting problem due to adding more parameters. With respect to the binary classification F_1 measures, we can see

System	Four-way		Binary				
	F_1	Acc.	Comp.	Cont.	Expa.	Expa.+EntRel	Temp.
P&C2012	-	-	31.32	49.82	-	79.22	26.57
J&E2015	-	-	35.93	52.78	-	80.02	27.63
Zhang2015	38.80	55.39	32.03	47.08	68.96	80.22	20.29
R&X2014	38.40	55.50	39.70	54.40	70.20	80.44	28.70
R&X2015	40.50	57.10	**41.00**	53.80	69.40	-	33.30
B&D2015	-	-	36.36	55.76	61.76	-	27.30
Liu2016	44.98	57.27	37.91	**55.88**	69.97	-	37.17
Ji2016	42.30	**59.50**	-	-	-	-	-
NNMA(two-level)	**46.29**	57.17	36.70	54.48	**70.43**	80.73	**38.84**
NNMA(three-level)	44.95	57.57	39.86	53.69	69.71	**80.86**	37.61

Table 4: Comparison with the State-of-the-art Approaches.

that the "Comparison" relation needs more passes of reading compared to the other three relations. The reason may be that the identification of the "Comparison" depends more on some deep analysis such as semantic parsing, according to (Zhou et al., 2010).

Next, we compare our models with six state-of-the-art baseline approaches, as shown in Table 4. The six baselines are introduced as follows.

- **P&C2012**: Park and Cardie (2012) designed a feature-based method and promoted the performance through optimizing the feature set.
- **J&E2015**: Ji and Eisenstein (2015) used two recursive neural networks on the syntactic parse tree to induce the representation of the arguments and the entity spans.
- **Zhang2015**: Zhang et al. (2015) proposed to use shallow convolutional neural networks to model two arguments respectively. We replicated their model since they used a different setting in preprocessing PDTB.
- **R&X2014, R&X2015**: Rutherford and Xue (2014) selected lexical features, production rules, and Brown cluster pairs, and fed them into a maximum entropy classifier. Rutherford and Xue (2015) further proposed to gather extra weakly labeled data based on the discourse connectives for the classifier.
- **B&D2015**: Braud and Denis (2015) combined several hand-crafted lexical features and word embeddings to train a max-entropy classifier.
- **Liu2016**: Liu et al. (2016) proposed to better classify the discourse relations by learning from other discourse-related tasks with a multi-task neural network.
- **Ji2016**: Ji et al. (2016) proposed a neural language model over sequences of words and used the discourse relations as latent variables to connect the adjacent sequences.

It is noted that P&C2012 and J&E2015 merged the "EntRel" relation into the "Expansion" relation[1]. For a comprehensive comparison, we also experiment our model by adding a *Expa.+EntRel vs Other* classification. Our NNMA model with two attention levels exhibits obvious advantages over the six baseline methods on the whole. It is worth noting that NNMA is even better than the R&X2015 approach which employs extra data.

As for the performance on each discourse relation, with respect to the F_1 measure, we can see that our NNMA model can achieve the best results on the "Expansion", "Expansion+EntRel" and "Temporal" relations and competitive results on the "Contingency" relation . The performance of recognizing the "Comparison" relation is only worse than R&X2014 and R&X2015. As (Rutherford and Xue, 2014) stated, the "Comparison" relation is closely related to the constituent parse feature of the text, like production rules. How to represent and

[1]EntRel is the entity-based coherence relation which is independent of implicit and explicit relations in PDTB. However some research merges it into the implicit Expansion relation.

exploit these information in our model will be our next research focus.

3.3 Analysis of Attention Levels

The multiple attention levels in our model greatly boost the performance of classifying implicit discourse relations. In this subsection, we perform both qualitative and quantitative analysis on the attention levels.

First, we take a three-level NNMA model for example and analyze its attention distributions on different attention levels by calculating the mean Kullback-Leibler (KL) Divergence between any two levels on the training set. In Figure 3, we use kl_{ij} to denote the KL Divergence between the i^{th} and the j^{th}attention level and use kl_{ui} to denote the KL Divergence between the uniform distribution and the i^{th} attention level. We can see that each attention level forms different attention distributions and the difference increases in the higher levels. It can be inferred that the 2^{nd} and 3^{rd} levels in NNMA gradually neglect some words and pay more attention to some other words in the arguments. One point worth mentioning is that *Arg-2* tends to have more non-uniform attention weights, since kl_{u2} and kl_{u3} of *Arg-2* are much larger than those of *Arg-1*. And also, the changes between attention levels are more obvious for *Arg-2* through observing the values of kl_{12}, kl_{13} and kl_{23}. The reason may be that *Arg-2* contains more information related with discourse relation and some words in it tend to require focused attention, as *Arg-2* is syntactically bound to the implicit connective.

At the same time, we visualize the attention levels of some example argument pairs which are analyzed by the three-level NNMA. To illustrate the k^{th} attention level, we get its attention weights $\boldsymbol{a}_k^1$ and $\boldsymbol{a}_k^2$ which reflect the contribution of each word and then depict them by a row of color-shaded grids in Figure 2.

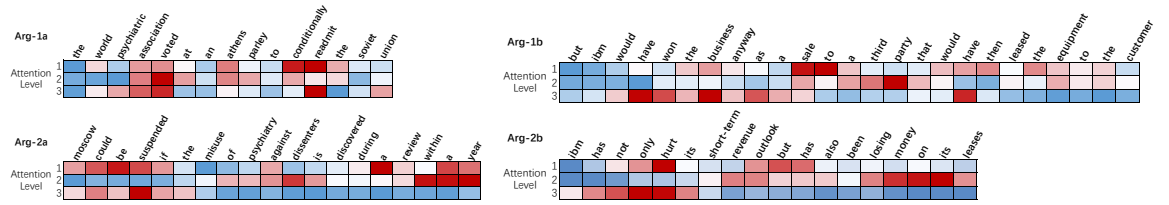
We can see that the NNMA model focuses on different words on different attention levels. Interestingly, from Figure 2, we find that the 1^{st} and 3^{rd} attention levels focus on some similar words, while the 2^{nd} level is relatively different from them. It seems that NNMA tries to find some clues (e.g. "moscow could be suspended" in *Arg-2a*; "won the business" in *Arg-1b*; "with great aplomb he considers not only" in *Arg-2c*) for recognizing the discourse relation on the 1^{st} level, looking closely at other words (e.g. "misuse of psychiatry against dissenters" in *Arg-2a*; "a third party that" in *Arg-1b*; "and support of hitler" in *Arg-2c*) on the 2^{nd} level, and then reconsider the arguments, focus on some specific words (e.g. "moscow could be suspended" in *Arg-2a*; "has not only hurt" in *Arg-2b*) and make the final decision on the last level.

4 Related Work

4.1 Implicit Discourse Relation Classification

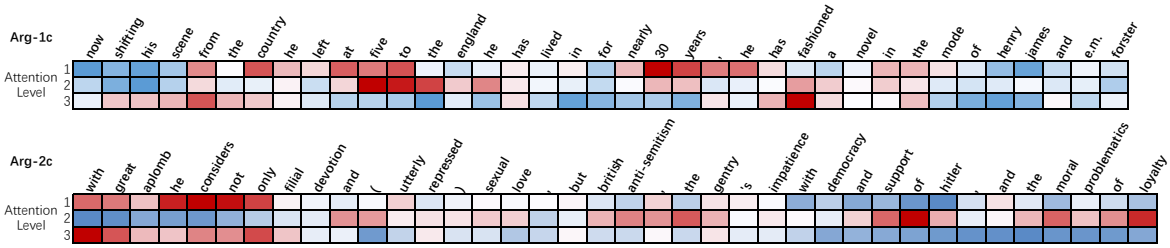
The Penn Discourse Treebank (PDTB) (Prasad et al., 2008), known as the largest discourse corpus, is composed of 2159 Wall Street Journal articles. Each document is annotated with the predicate-argument structure, where the predicate is the discourse connective (e.g. while) and the arguments are two text spans around the connective. The discourse connective can be either explicit or implicit. In PDTB, a hierarchy of relation tags is provided for annotation. In our study, we use the four top-level tags, including Temporal, Contingency, Comparison and Expansion. These four core relations allow us to be theory-neutral, since they are almost included in all discourse theories, sometimes with different names (Wang et al., 2012).

Implicit discourse relation recognition is often treated as a classification problem. The first work to tackle this task on PDTB is (Pitler et al., 2009). They selected several surface features to train four binary classifiers, each for one of the top-level PDTB relation classes. Extending from this work, Lin et al. (2009) further identified four different feature types representing the context, the constituent parse trees, the dependency parse trees and the raw text respectively. Rutherford and Xue (2014) used brown cluster to replace the word pair features for solving the sparsity problem. Ji and Eisenstein (2015) adopted two recursive neural networks to exploit the representation of arguments and entity spans. Very recently, Liu et al. (2016) proposed a two-dimensional convolutional neural network (CNN) to model the argument pairs and employed a multi-task learning framework to boost the performance by learning from other discourse-related tasks. Ji et al. (2016) considered discourse relations as



(a) Example with *Comparison* relation

(b) Example with *Contingency* relation



(c) Example with *Expansion* relation

Figure 2: Visualization Examples: Illustrating Attentions Learned by NNMA. (The blue grid means the the attention on this word is lower than the value of a uniform distribution and the red red grid means the attention is higher than that.)

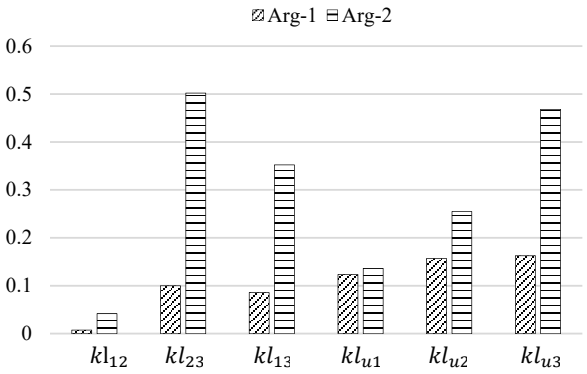


Figure 3: KL-divergences between attention levels

latent variables connecting two token sequences and trained a discourse informed language model.

4.2 Neural Networks and Attention Mechanism

Recently, neural network-based methods have gained prominence in the field of natural language processing (Kim, 2014). Such methods are primarily based on learning a distributed representation for each word, which is also called a word embedding (Collobert et al., 2011).

Attention mechanism was first introduced into neural models to solve the alignment problem between different modalities. Graves (2013) designed a neural network to generate handwriting based on a text. It assigned a window on the input text at each step and generate characters based on the content within the window. Bahdanau et al. (2014) introduced this idea into machine translation, where their model computed a probabilistic distribution over the input sequence when generating each target word. Tan et al. (2015) proposed an attention-based neural network to model both questions and sentences for selecting the appropriate non-factoid answers.

In parallel, the idea of equipping the neural model with an external memory has gained increasing attention recently. A memory can remember what the model has learned and guide its subsequent actions. Weston et al. (2015) presented a neural network to read and update the external memory in a recurrent manner with the guidance of a question embedding. Kumar et al. (2015) proposed a similar model where a memory was designed to change the gate of the gated recurrent unit for each iteration.

5 Conclusion

As a complex text processing task, implicit discourse relation recognition needs a deep analysis

of the arguments. To this end, we for the first time propose to imitate the repeated reading strategy and dynamically exploit efficient features through several passes of reading. Following this idea, we design neural networks with multiple levels of attention (NNMA), where the general level and the attention levels represent the first and subsequent passes of reading. With the help of external short-term memories, NNMA can gradually update the arguments representations on each attention level and fix attention on some specific words which provide effective clues to discourse relation recognition. We conducted experiments on PDTB and the evaluation results show that our model can achieve the state-of-the-art performance on recognizing the implicit discourse relations.

Acknowledgments

We thank all the anonymous reviewers for their insightful comments on this paper. This work was partially supported by National Key Basic Research Program of China (2014CB340504), and National Natural Science Foundation of China (61273278 and 61572049). The correspondence author of this paper is Sujian Li.

References

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

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of CoNLL*.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of EMNLP*.

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hugo Hernault, Helmut Prendinger, David A duVerle, Mitsuru Ishizuka, et al. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. *arXiv preprint arXiv:1603.01913*.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of EMNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of EMNLP*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural network. In *Proceedings of AAAI*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of SigDial*.

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

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of ACL*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K. Joshi, and Bonnie L. Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC*.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of EACL*.

Attapol T Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of NAACL*.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL*.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of NAACL*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of EMNLP*.

Xun Wang, Sujian Li, Jiwei Li, and Wenjie Li. 2012. Implicit Discourse Relation Recognition by Selecting Typical Training Examples. In *Proceedings of COLING*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *Proceedings of ICLR*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of EMNLP*.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the ICCL*, pages 1507–1514.

Antecedent Selection for Sluicing: Structure and Content

Pranav Anand
Linguistics
UC Santa Cruz
panand@ucsc.edu

Daniel Hardt
IT Management
Copenhagen Business School
dh.itm@cbs.dk

Abstract

Sluicing is an elliptical process where the majority of a question can go unpronounced as long as there is a salient antecedent in previous discourse. This paper considers the task of antecedent selection: finding the correct antecedent for a given case of sluicing. We argue that both syntactic and discourse relationships are important in antecedent selection, and we construct linguistically sophisticated features that describe the relevant relationships. We also define features that describe the relation of the content of the antecedent and the sluice type. We develop a linear model which achieves accuracy of 72.4%, a substantial improvement over a strong manually constructed baseline. Feature analysis confirms that both syntactic and discourse features are important in antecedent selection.

1 Introduction

Ellipsis involves sentences with missing subparts, where those subparts must be interpretatively filled in by the hearer. How this is possible has been a major topic in linguistic theory for decades (Sag, 1976; Chung et al., 1995; Merchant, 2001). One widely studied example is *verb phrase ellipsis* (VPE), exemplified by (1).

(1) Harry traveled to southern Denmark to study botany . Tom did too .

In the second sentence (*Tom did too*) the verb phrase is entirely missing, yet the hearer effortlessly 'resolves' (understands) its content to be *traveled to southern Denmark to study botany*.

Another widely studied case of ellipsis is *sluicing*, in which the majority of a question is unpronounced, as in (2).

(2) Harry traveled to southern Denmark to study botany . I want to know **why** .

Here the content of the question, introduced by the WH-phrase *why*, is missing, yet it is understood by the hearer to be *why did Harry travel to southern Denmark to study botany?*. In both of these cases, ellipsis resolution is made possible by the presence of an *antecedent*, material in prior discourse that, informally speaking, is equivalent to what is missing.

Ellipsis poses an important challenge for many applications in language technology, as various forms of ellipsis are known to be frequent in a variety of languages and text types. This is perhaps most evident in the case of question-answering systems, since elliptical questions and elliptical answers are both very common in discourse. A computational system that can effectively deal with ellipsis involves three subtasks (Nielsen, 2005): *ellipsis detection*, in which a case of ellipsis is identified, *antecedent selection*, in which the antecedent for a case of ellipsis is found, and *ellipsis resolution*, where the content of the ellipsis is filled in with reference to the antecedent and the context of the ellipsis. Here, we focus on antecedent selection for sluicing. In addressing this problem of antecedent selection, we make use of a newly available annotated corpus of sluice occurrences (Anand and McCloskey, 2015). This corpus consists of 4100 automatically parsed and annotated examples from the New York Times subset of the Gigaword Corpus, of which 2185 are

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1234–1243,
Austin, Texas, November 1-5, 2016.

publicly available.

Sluicing antecedent selection might appear simple – after all, it typically involves a sentential expression in the nearby context. However, analysis of the annotated corpus data reveals surprising ambiguity in the identification of the antecedent for sluicing.

In what follows, we describe a series of algorithms and models for antecedent selection in sluicing. Following section 2 on background, we describe our dataset in section 3. Then in section 4, we describe the structural factors that we have identified as relevant for antecedent selection. In section 5, we look at ways in which the content of the sluice and the content of the antecedent tend to be related to each other: we address lexical overlap, as well as the probabilistic relation of head verbs to WH-phrase types, and the relation of correlate expressions to sluice types. In section 6 we present two manually constructed baseline classifiers, and then we describe an approach to automatically tuning weights for the complete set of features. In section 7 we present the results of these algorithms and models, including results involving various subsets of features, to better understand their contributions to the overall results. Finally in section 8 we discuss the results in light of plans for future work.

2 Background

2.1 Sluicing and ellipsis

Sluicing is formally defined in theoretical linguistics as ellipsis of a question, leaving only a WH-phrase *remnant*. While VPE is licensed only by a small series of auxiliaries (e.g., modals, *do*, see Lobeck (1995)), sluicing can occur wherever questions can, both in unembedded 'root' environments (e.g., *Why?*) or governed by the range of expressions that embed questions, like *know* in (2). Sluicing is argued to be possible principally in contexts where there is uncertainty or vagueness about an issue (Ginzburg and Sag, 2000). In some cases, this manifests as a *correlate*, an overt indefinite expression whose value is not further specified, like *one of the candidates* in (3). But in many others, like that in (2) or (4), there is no correlate, and the uncertainty is implicit.

(3) They 've made an offer to [$_{cor}$ one of the candidates] , but I 'm not sure **which one**

(4) They were firing , but **at what** was unclear

The existence of correlate-sluices suggests an obvious potential feature type for antecedent detection. However, the annotated sluices in (Anand and McCloskey, 2015) have correlates only 22% of the time, making this process considerably harder. We return to the question of correlates in section 5.1.

2.2 Related Work

The first large-scale study of ellipsis is due to Hardt (1997), which addresses VPE. Examining 644 cases of VPE in the Penn Treebank, Hardt presents a manually constructed algorithm for locating the antecedent for VPE, and reports accuracy of 75% to 94.8%, depending on whether the metric used requires exact match or more liberal overlap or containment. Several preference factors for choosing VPE antecedents are identified (Recency, Clausal Relations, Parallelism, and Quotation). One of the central components of the analysis is the identification of structural constraints which rule out antecedents that improperly contain the ellipsis site, an issue we also address here for sluicing. Drawing on 1510 instances of VPE in both the British National Corpus (BNC) and the Penn Treebank, Nielsen (2005) shows that a maxent classifier using refinements of Hardt's features can achieve roughly similar results to Hardt's, but that additional lexical features do not help appreciably.

Nielsen chooses to optimize for Hardt's Head Overlap metric, which assigns success to any candidate containing/contained in the correct antecedent. There are thus many "correct" antecedents for a given instance of VPE, which mitigates the class imbalance problem. However, the approach does not provide a way to discriminate between these containing candidates, an important step in the eventual goal of resolving the ellipsis.

There is no similar work on antecedent selection for sluicing, though there have been small-scale corpora gathered for sluices (Nykiel, 2010; Beecher, 2008). In addition, Fernandez et al. (2005) build rule-based and memory-based classifiers for the pragmatic import of root (unembedded) sluices in the BNC, based on the typology of Ginzburg and Sag (2000). Using features for the type of WH-

phrase, markers of mood (declarative/interrogative) and polarity (positive/negative) as well as the presence of correlate-like material (e.g., quantifiers, definites, etc.), they can diagnose the purpose of a sluice in a dataset of 300 root sluices with 79% average F-score, a 5% improvement over the MLE. Fernandez et al. (2007) address the problem of identifying sluices and other non-sentential utterances. We don't address that problem in the current work. Furthermore, Fernandez et al. (2007) and Fernandez et al. (2008) address the general problem of non-sentential utterances or fragments in dialogue, including sluices. Sluicing in dialogue differs from sluicing in written text in various ways: there is a high proportion of root sluices, and antecedent selection is likely mitigated by the length of utterances and the order of conversation. As we discuss, many of our newswire sluices evince difficult patterns of containment inside the antecedent (particularly what we call interpolated and cataphoric sluices), and it does not appear from inspection that root sluices ever participate in such processes.

Looking more generally, there is an obvious potential connection between antecedent selection for ellipsis and the problem of coreference resolution (see Hardt (1999) for an explicit theoretical link between the two). However, entity coreference resolution is a problem with two major differences from ellipsis antecedent detection: a) the antecedent and anaphor often share a variety of syntactic, semantic, and morphological characteristics that can be featurally exploited; b) entity expressions in a text are often densely coreferent, which can help provide proxies for discourse salience of an entity.

In contrast, abstract anaphora, particularly discourse anaphora (*this/that* anaphora to something sentential), may offer a more parallel case to ours. Here, Kolhatkar et al. (2013) use a combination of syntactic type, syntactic/word context, length, and lexical features to identify the antecedents of anaphoric shell nouns (*this fact*) with precision from 0.35-0.72. Because of the sparsity of these cases, Kolhatkar et al. use Denis and Baldridge's (2008) candidate ranking model (versus a standard mention-pair model (Soon et al., 2001)), in which all potential candidates for an anaphor receive a relative rank in the overall candidate pool. In this paper, we will pursue a hillclimbing approach to antecedent selection, inspired by the candidate ranking scheme.

3 Data

3.1 The Annotated Dataset

Our dataset, described in Anand and McCloskey (2015), consists of 4100 sluicing examples from the New York Times subset of the Gigaword Corpus, 2nd edition. This dataset is the first systematic, exhaustive corpus of sluicing.[1] Each example is annotated with four main tags, given in terms of token sequence offsets: the *sluice remnant*, the *antecedent*, and then inside the antecedent the main *predicate* and the *correlate*, if any. The annotations also provide a free-text resolution. Of the 4100 annotated, 2185 sluices have been made publicly available; we use that smaller dataset here. We make use of the annotation of the antecedent and remnant tags. See Anand and McCloskey (2015) for additional information on the dataset and the annotation scheme. For the feature extraction in section 4, we rely on the the token, parsetree, and dependency parse information in Annotated Gigaword (extracted from Stanford CoreNLP).

3.2 Defining the Correct Antecedent

Because of disagreements with the automatic parses of their data, Anand and McCloskey (2015) had annotators tag token sequences, not parsetree constituents. As a result, 10% of the annotations are not sentence-level (i.e., S, SBAR, SBARQ) constituents, such as the VP antecedent in (5), and 15% are not constituents at all, such as the case of (6), where the parse lacks an S node excluding the initial temporal clause. We describe two different ways to define what will count as the correct antecedent in building and assessing our models.

3.2.1 Constituent-Based Accuracy

Linguists generally agree that the antecedent for sluicing is a sentential constituent (see Merchant (2001) and references therein). Thus, it is straightforward to define the antecedent as the minimal

[1] 4100 sluices works out to roughly 0.14% of WH-phrases in the NYT portion of Gigaword. However, note that this includes all uses of WH-phrases (e.g., clefts and relative clauses), whereas sluicing is only possible for WH-questions. It's not clear how many questions there are in the dataset (distinguishing questions and other WH-phrases is non-trivial).

sentence-level constituent containing the token sequence marked as the antecedent. Then we define CONACCURACY as the percentage of cases in which the system selects the correct antecedent, as defined here.

While it is linguistically appealing to uniformly define candidates as sentential constituents, the annotator choices are sometimes not parsed that way, as in the following examples:

(5) " I do n't know how , " said Mrs. Kitayeva , " but [$_S$ we want [$_{VP}$ **to bring Lydia home**] , in any condition] . "

(6) [$_S$ [$_S$$BAR$ When Brown , an all-America tight end , was selected in the first round in 1992] **he was one of the highest rated players on the Giants ' draft board**]

In such cases, there is a risk that we will not accurately assess the performance of our systems, since the system choice and annotator choice will only partially overlap.

3.2.2 Token-Based Precision and Recall

Here we define a metric which calculates the precision and recall of individual token occurrences, following Bos and Spenader (2011) (see also Kolhatkar and Hirst (2012)). This will accurately reflect the discrepancy in examples like (5) – according to ConAccuracy, a system choice of *we want to bring Lydia home in any condition* is simply considered correct, as it is the smallest sentential consituent containing the annotator choice. According to the Token-Based metric, we see that the system achieves recall of 1; however, since the system includes six extraneous tokens, precision is .4. We define TOKF as the harmonic mean of Token-Based Precision and Recall; for (5), TokF is .57.

3.3 Development and Test Data

The dataset consists of 2185 sluices extracted from the New York Times between July 1994 and December 2000. For feature development, we segmented the data into a development set (DS) of the 453 sluices from July 1994 to December 1995. The experiments in section 6 were carried out on a test set (TS) of the 1732 sluices in the remainder of the dataset, January 1996 to December 2000.

4 Structure

Under our assumptions, the candidate antecedent set for a given sluice is the set of all sentence-level parsetree constituents within a n-sentence radius around the sluice sentence (based on DS, we set $n = 2$). Because sentence-level constituents embed, in DS there are on average 6.4 candidate antecedents per sluice. However, because ellipsis resolution involves identification of an antecedent, we assume that it, like anaphora resolution, should be sensitive to the overall salience of the antecedent. This means that there should be, in principle, proxies for salience that we can exploit to diagnose the plausibility of a candidate for sluicing in general. We consider four principle kinds of proxies: measures of candidate-sluice distance, measures of candidate-sluice containment, measures of candidate 'main point', and candidate-sluice discourse relation markers.

4.1 Distance

Within DS, 63% of antecedents are within the same sentence as the sluice site, and 33% are in the immediately preceding sentence. In terms of candidates, the antecedent is on average the 5th candidate from the end of the n-sentence window. The positive integer-valued feature DISTANCE tracks these notions of recency, where DISTANCE is 1 if the candidate is the candidate immediately preceding or following the sluice site (DISTANCE is defined to be 0 only for infinitival Ss like S0 in (7) below). The feature FOLLOWS marks whether a candidate follows the sluice.

4.2 Containment

As two-thirds of the antecedents are in the same sentence as the sluice, we need measures to distinguish the candidates internal to the sentence containing the sluice. In general, we want to exclude any candidate that 'contains' (i.e., dominates) the sluice, such as S0 and S-1 in (7). One might have thought that we want to always exclude the entire sentence (here, S-4) as well, but there are several cases where the smallest sentence-level constituent containing the annotated antecedent dominates the sluice, including: parenthetical sluices inside the antecedent (8), sluices in subordinating clauses (9), or

sluice VPs coordinated with the antecedent VP (10). We thus need features to mark when such candidates are 'non-containers'.

(7) [$_{S-4}$ [$_{S-3}$ I have concluded that [$_{S-2}$ I can not support the nomination] , and [$_{S-1}$ I need [$_{S0}$ to explain **why**]].]

(8) [$_{S-2}$ A major part of the increase in coverage , [$_{S-1}$ though Mitchell 's aides could not say just **how much** ,] would come from a provision providing insurance for children and pregnant women .]

(9) [$_{S-3}$ Weltlich still plans [$_{S-2}$ to go , [$_{S-1}$ although he does n't know **where**]]]

(10) [$_{S-2}$ State regulators have ordered 20th Century Industries Inc. [$_{S-1}$ to begin paying $ 119 million in Proposition 103 rebates or explain **why not** by Nov. 14 .]]

Conceptually, what renders S-3 in (9), S-2 in (8), and S-1 in (10) non-containers is that in all three cases the sluice is semantically dissociable from the rest of the sentence. We provide three features to mark this. First, the boolean feature SLUICEINPARENTHETICAL marks when the sluice is dominated by a parenthetical (a PRN node in the parse or an *(al)though* SBAR delimited by punctuation). Second, SLUICEINCOORDVP marks the configuration exemplified (10).

We also compute a less structure-specific measure of whether the candidate is meaningful once the sluice (and material dependent on it) is removed. This means determining, for example, that S-4 in (7) is meaningful once *to explain why* . is removed but S-1 is not. But the latter result follows from the fact that the main predicate of S-1, *need* takes the sluice govering verb *explain* as an argument, and hence removing that argument renders it semantically incomplete. We operationalize this in terms of complement dependency relations. We first locate the largest subgraph containing the sluice in a chain of *ccomp* and *xcomp* relations. This gives us gov_{max}, the highest such governor (i.e., *explain*) in Fig. 1. The subgraph dependent on gov_{max} is then removed, as indicated by the grayed boxes in Fig 1. If the resulting subgraph contains a verbal governor, the candidate is meaningful and CONTAINSSLUICE is false. By this logic, S-4 in (7) is meaningful because it contains *concluded*, but S-1 is not, because there is no verbal material remaining.

4.3 Discourse Structure

It has often been suggested (Asher, 1993; Hardt, 1997; Hardt and Romero, 2004) that the antecedent selection process is very closely tied to discourse relations, in the sense that there is a strong preference or even a requirement for a discourse relation between the antecedent and ellipsis.

Here we define several features that indicate either that a discourse relation is present or is not present.

We begin with features indicating that a discourse relation is not present: the theoretical linguistics literature on sluicing has noted that antecedents not in the 'main point' of an assertion (e.g., ones in appositives (AnderBois, 2014) or relative clauses (Cantor, 2013)) are very poor antecedents for sluices, presumably because their content is not very salient. The boolean features CANDINPARENTHETICAL (determined as for the sluice above) and CANDINRELCLAUSE mark these patterns.[2]

We also define features that would tend to indicate the presence of a discourse relation. These have to do with antecedents that occur after the sluice. Although antecedents overwhelmingly occur prior to sluices, we observe one prominent cataphoric pattern in DS, where the sentence containing the sluice is coordinated with a contrastive discourse relation; this is exemplified in (11).

(11) " I do n't know **why** , but I like Jimmy Carter . "

Three features are designed to capture this pattern: COORDWITHSLUICE indicates whether the sluice and candidate are connected by a coordination dependency, AFTERINITIALSLUICE marks the conjunctive condition where the candidate follows a sluice initial in its sentence, and IMMEDAFTERINITIALSLUICE marks a candidate that is the closest following candidate to an initial sluice.

[2]This feature might be seen as an analog to the apposition features used in nominal coreference resolution (Bengtson and Roth, 2008), but there it is used to link appositives, whereas here it is to exclude candidates.

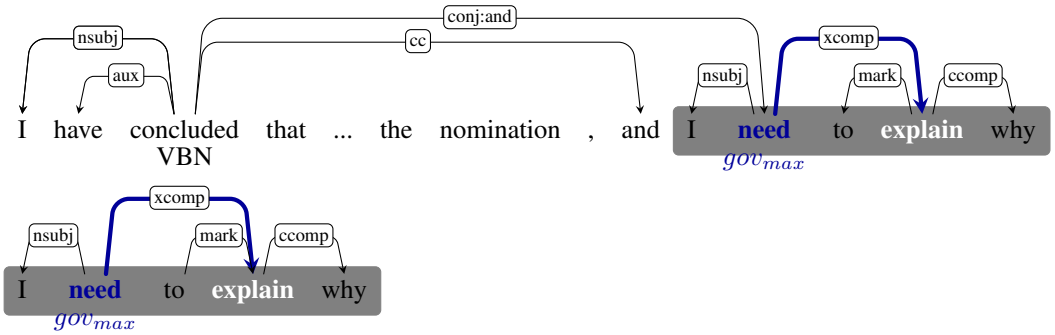


Figure 1: Sluice containment for S-4 and S-1 in (7). Starting at the governor of the sluice, *explain*, find gov_{max} *need* and delete its transitive dependents. The candidate does not contain the sluice if the remaining graph contains verbal governors.

5 Content

In addition to the structural features above, we also compute several features relating the content of the sluice site and the antecedent. The intuition behind these relational features is the following: each sluice type (*why*, *who*, *how much*, etc.) represents a certain type of question, and each candidate represents a particular type of predication. For a given a sluice type, some predications might fit more naturally than others. More generally, it is a common view that an elliptical expression and its antecedent contain matching "parallel elements".[3]

Below we describe three approaches to this: one simply looks for lexical overlap – words that occur both in the sluice expression and in the candidate. The second involves a more general notion of how a predication fits with a sluice type. To capture this, we gather co-occurrence counts of main verb and sluice types. The third approach compares potential correlates in candidates with the type of sluice.

5.1 Overlap

One potential candidate for overlap information is the presence of a correlate in the antecedent. However, 75% of of sluices involve WH-phrases that typically involve no correlate (e.g., *how, when, why*). The pertinent exception to this are extent sluices (ones where the remnant is *how (much|many|JJ)*), which have been argued to heavily favor a correlate (Merchant, 2001), such as (12) below (though see (13) for a counterexample).

[3]This term is from Dalrymple et al. (1991); a similar general view about parallelism in ellipsis arises in many different theories, such as Prüst et al. (1994) and Asher (1993).

(12) The 49ers are [$_{corr}$ very good] .
It 's hard to know **how good** because the Cowboys were the only team in the league who could test them .

We thus compute the number of tokens of OVERLAP between the content terms in the WH-phrase sluice (non-WH, non prepositional) and the entire antecedent.

5.2 Wh-Predicate

Even for correlate-less sluices, the WH-phrase must semantically cohere with the main predicate of the antecedent. Thus, in (13), S-3 is a more likely antecedent than S-2 because *increase* is more likely to take an implicit extent than *predict*. Although we could have consulted a lexically rich resource (e.g, VerbNet, FrameNet), our hope was that this general approach could carry over to less argument-specific combinations such as *how* with *complete* and *raise* in (14).

(13) [$_{S-3}$ Deliveries would increase as a result of the acquisition] , [$_{S-2}$ he predicted] , but [$_{S-1}$ he would not say by how much]

(14) [$_{S-4}$ [$_{S-3}$ Once the city and team complete a contract] , the Firebirds will begin to raise $ 9 million] , [$_{S-2}$ team president Yount said] , [$_{S-1}$ but he would not say how] .

Our assumption is that some main predicates are more likely than others for a given sluice type, and we wish to gather data that reveals these probabilities. This is somewhat similar to the approach of Hindle and Rooth (1993), who gather probabilities

that reflect the association of verbal and nominal heads with prepositions to disambiguiate prepositional phrase attachment.

One way to collect these would be to use our sluicing data, which consists of a total of 2185 annotated examples. However, the probabilities of interest are not about sluicing *per se*. Rather, they are about how well a given predication fits with a given type of question. Thus instead of using our comparatively small set of annotated sluicing examples, we used overt WH-constructions in Gigaword to observe cooccurrences between question types and main predicates. To find overt WH-constructions, we extracted all instances where a WH-phrase is: a) a dependent (to exclude cases like *Who?*) and b) not at the right edge of a VP (to exclude sluices like *know who*, per Anand and McCloskey (2015)). To further ensure that we were not overlapping with our dataset, we did this only for the non=NYT subsets of Gigaword (i.e., AFP, APW, CNA, and LTW). This procedure generated 687,000 WH-phrase instances, and 79,753 WH-phrase-governor bigram types. From these bigrams, we calculated WH-PREDICATE, the normalized pmi of WH-phrase type and governor lemma in Annotated Gigaword.

5.3 Correlate Overlap

Twenty-two percent of our data has correlates, and these correlates should be discriminative for particular sluice types. For example, temporal (*when*) sluices have timespan correlates (e.g., *tomorrow*, *later*), while entity (*who/what*) sluices have individuals as correlates (e.g., *someone*, *a book*). We extracted four potential integer-valued correlate features from each candidate: LOCATIVECORR is the number of primarily locative prepositions (those with a locative MLE in The Preposition Project (Litowski and Hargraves, 2005)). ENTITYCORR is the number of nominals in the candidate that are indefinite (bare nominals or ones with a determiner relation to *a, an* and weak quantifiers (*some*, *many*, *much*, *few*, *several*).TEMPORALCORR is the number of lexical patterns in the candidate for TIMEX3 annotations in Timebank 1.2 (Pustejovesky et al., 2016). WHICHCORR is the pattern for entities plus *or*.

distance	DISTANCE, FOLLOWS
containment	CONTAINSSLUICE ISDOMINATED-BYSLUICE
discourse structure	COORDWITHSLUICE, AFTERINITIALSLUICE, IMMEDAFTERINITIALSLUICE, CANDINPARENTHETICAL, CANDINRELCLAUSE
content	OVERLAP, WH-PREDICATE
correlate	LOCATIVECORR, ENTITYCORR, TEMPORALCORR, WHICHCORR

Table 1: Summary of features used in experiments.

6 Algorithms

Mention-pair coreference models reduce coreference resolution to two steps: a local binary classification, and a global resolution of coreference chains. We may see antecedent selection as a similar two-stage process: classification on the probability a given candidate is an antecedent, and then selection of the most likely candidate for a given sluice. As Denis and Baldridge (2008) note, one limitation of this approach is that the overall rank of the candidates is never directly learned. They instead propose to learn the *rank* of a candidate c for antecedent a, modeled as the log-linear score of a candidate across a set of coreference models m, $(exp \sum_j w_j m_j(c, a))$, normalized by the sum of candidate scores. We apply the same approach to our problem, viewing each feature in Table 1 as a model, and estimating weights for the features by hill-climbing. We begin by defining constructed baselines which are implemented by manually assigning weights. We then consider the results of a maxent classifier over the features. Finally, we determine the weights directly by hill-climbing with random restarts.

6.1 Manual Baselines

Random simply selects candidates at random. Clst chooses the closest candidate that starts before the sluice. This is done by assigning a weight of -1 to DISTANCE and -10 to FOLLOWING (to exclude

the following candidate), and 0 to all other features. ClstBef chooses the closest candidate that entirely precedes the sluice (i.e., starts before and does not contain the sluice site). To construct ClstBef, we change the weight of CONTAINSSLUICE to -10, which means that candidates containing the sluice will never be chosen.

6.2 A maxent model

We trained a maxent classifier on the features in Table 1 for the binary antecedent-not antecedent task. With 10-fold cross-validation on the test set, the maxent model achieved an average accuracy on the binary antecedent task of 87.1 and an F-score of 53.8 (P=63.9, R=46.5). We then constructed an antecedent selector that chose the candidate with the highest classifier score.

6.3 Hill-Climbing

We define a procedure to hill-climb over weights in order to maximize ConAccuracy over the entire training set (maximizing TokF yielded similar results, and is not reported here). Weights are initialized with random values in the interval [-10,10]. At iteration i, the current weight vector is compared to alternatives differing from it by the current step size on one weight, and the best new vector is selected. For the results reported here, we performed 13 random restarts and exponential step size $10 * i^{.5}$ (values that maximized performance on the DS).

7 Results

We performed 10-fold cross-validation over TS on the hill-climbed and maxent models above, producing average ConAccuracy and TokF as shown in Table 2, which also gives results of the three baselines on the entire dataset. The hill-climbed approach with all features substantially outperformed the baselines, achieving a ConAccuracy of 72.4%.

We investigated the performance of our hill-climbing procedure with ablation of several feature subsets. We ablated features by group, as in Table 1. Table 2 shows the results for using four groups and only one group, as well as the top two three group and two group combinations.

Features fall in three tiers. Distance features are the most predictive: all the top systems use them, and they alone perform reasonably well (like Clst).

	A:Tr	F:Tr	A:Tes	F:Tes
HC-DCSNR	73.8	72.4	72.4	71.5
HC-CSNR	41.8	51.8	40.3	51.6
HC-DCSR	72.9	71.6	72.1	71.0
HC-DSNR	53.5	59.1	52.7	58.3
HC-DCNR	65.8	67.1	64.6	65.9
HC-DCSN	73.3	72.1	72.7	71.8
HC-DCS	72.7	71.5	72.5	71.3
HC-DCN	65.6	67.8	64.3	66.8
HC-DC	63.3	65.4	63.0	65.4
HC-DS	51.2	57.2	50.9	57.1
HC-D	41.6	51.6	41.5	51.6
HC-C	30.6	45.1	28.9	45.3
HC-S	30.7	43.0	27.0	42.0
HC-N	30.5	38.6	30.7	38.2
HC-R	23.6	35.9	22.2	33.1
Maxent	65.3	70.2	64.2	68.0
Random	19.4	44.1	19.5	46.3
Clst	41.2	52.1	na	na
ClstBef	56.5	67.9	na	na

Table 2: Average (Con)A(ccuracy) and (Tok)F(-Score) for Tr(ain) and Tes(t) splits on 10-fold cross-validation of data. Feature groups: **D**istance, **C**ontainment, Discourse **S**tructure, co**N**tent, co**R**relate. (Red marks results not significantly different (via paired t-test) from HC-DCSNR.)

Containment and then Discourse Structure features are the next most helpful. The full system has a ConAccuracy of 72.4 on the TS, not reliably different from several systems without Content and/or Correlate features. At the same time, the scores for these feature types on their own show that they are predictive of the antecedent: The Correlate feature **R** has a score of 22.2, which is a rather modest, but statistically significant, improvement over Random. The Content feature **N** improves quite substantially, up to 30.7. This suggests that there is some redundancy with the other features, so that the contributions of Content and Correlate are not observed in combination with them. (HC-N and HC-R's lower than Random TokF is a result of precision: Random more often selects very small candidates inside the correct antecedent, leading to a higher precision.)

The Content and Correlate features concern relations between the type of sluice and the content of the antecedent; since other features do not capture this, it is puzzling that these provide no further improvement. To better understand why this is, we investigated the performance of our feature

sets by sluice type. For the top performing systems, we found that antecedent selection for sluices over extents (e.g, *how much*, *how tall*) performed 11% better than average and those over reasons (*why*) and manner (*how*) performed 13% worse than average; no other WH-phrase types differed significantly from average. Importantly, this finding was consistent even for the systems without Content or Correlate features, which we extracted in large part to help highlight possible correlate material for extent sluices as well as entity (*who/what*) and temporal (*when*) sluices.

We also examined systems knocking out our best performing features, Distance, Containment, and Discourse Structure. When Distance features were omitted, we saw a bimodal distribution: reason and manner sluice antecedent selection was 31% better than expected (based on the full system differences discussed above), and the other sluices performed 22% worse. When Containment features were omitted, reason sluices performed 10% better than expected, while extent ones were 10% worse. Finally, when Discourse Structure features were removed, entity and temporal sluices had half the error rate we would expect. While it is hard to provide a clear takeaway from these differences, they do point to the relative difficulty in locating sluice antecedents based on WH-phrase type, and they also suggest that different sluice types present quite different challenges. This suggests that one promising line might be to learn different featural weights for each sluice type.

8 Conclusion

We have addressed the problem of sluicing antecedent selection by defining linguistically sophisticated features describing the structure and content of candidates. We described a hill-climbed model which achieves accuracy of 72.4%, a substantial improvement over a strong manually constructed baseline. We have shown that both syntactic and discourse relationships are important in antecedent selection. In future work, we hope to improve the performance of several of our features. Notable among these are the discourse structural proxies we found to make a contribution to the model. These features constitute a quite limited view of discourse structure, and we suspect that a better representation of discourse structure might well lead to further improvements. One potential path would be to leverage data where discourse relations are explicitly annotated, such as that in the Penn Discourse Treebank (Prasad et al., 2008). In addition, although our Content and Correlate features were not useful alongside the others, we hope that more refined versions of those could provide some assistance. We also noted that our performance was impacted by WH-types, and therefore it might be helpful to learn different featural weights per type.

In closing, we would like to return to the larger question of effectively handling ellipsis. The solution to antecedent selection that we have presented here provides a starting point for addressing the problem of *resolution*, in which the content of the sluice is filled in. However, even if the correct antecedent is selected, the missing content is not always an exact copy of the antecedent – often substantial modifications will be required – and an effective resolution system will have to negotiate such mismatches. As it turns out, many incorrect antecedents differ from the correct antecedent in ways highly reminiscent of these mismatches. Thus, some of the errors of our selection algorithm may be most naturally addressed by the resolution system, and it may be that the relative priority of the specific challenges we identified here will become clearer as we address the next step down in the overall pipeline.

Acknowledgments

We gratefully acknowledge the work of Jim McCloskey in helping to create the dataset investigated here, as well as the principal annotators on the project. We thank Jordan Boyd-Graber, Ellen Riloff, and three incisive reviewers for helpful comments. This research has been sponsored by NSF grant number 1451819.

References

Pranav Anand and Jim McCloskey. 2015. Annotating the implicit content of sluices. In *The 9th Linguistic Annotation Workshop held in conjuncion with NAACL 2015*, page 178.

Scott AnderBois. 2014. The semantics of sluicing: Beyond truth conditions. *Language*, 90(4):887–926.

Nicholas Asher. 1993. *Reference to Abstract Objects in English*. Dordrecht.

Henry Beecher. 2008. Pramatic inference in the interpretation of sluiced Prepositional Phrases. In *San Diego Linguistic Papers*, volume 3, pages 2–10. Department of Linguistics, UCSD, La Jolla, California.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of vp ellipsis. *Language Resources and Evaluation*, 45(4):463–494.

Sara Cantor. 2013. Ungrammatical double-island sluicing as a diagnostic of left-branch positioning.

S. Chung, W. Ladusaw, and J. McCloskey. 1995. Sluicing and logical form. *Natural Language Semantics*, 3:1–44.

Mary Dalrymple, Stuart Shieber, and Fernando Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4), August.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2005. Automatic bare sluice disambiguation in dialogue. In *Proceedings of the IWCS-6 (Sixth International Workshop on Computational Semantics)*, pages 115–127, Tilburg, the Netherlands, January. Available at: `http://www.dcs.kcl.ac.uk/staff/lappin/recent_papers_index.html`.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2007. Classifying non-sentential utterances in dialogue: A machine learning approach. *Computational Linguistics*, 33(3):397–427.

Raquel Fernández, Jonathan Ginzburg, Howard Gregory, and Shalom Lappin. 2008. Shards: Fragment resolution in dialogue. In *Computing Meaning*, pages 125–144. Springer.

Jonathan Ginzburg and Ivan Sag. 2000. *Interrogative Investigations: The Form, Meaning and Use of English Interrogatives*. CSLI Publications, Stanford, Calif.

Daniel Hardt and Maribel Romero. 2004. Ellipsis and the structure of discourse. *Journal of Semantics*, 24(5):375–414.

Daniel Hardt. 1997. An empirical approach to vp ellipsis. *Computational Lingusitics*, 23(4):525–541.

Daniel Hardt. 1999. Dynamic interpretation of verb phrase ellipsis. *Linguistics & Philosophy*, 22(2):187–221.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

Varada Kolhatkar and Graeme Hirst. 2012. Resolving "this-issue" anaphora. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1255–65.

Varada Kolhatkar, Heike Zinmeister, and Graeme Hirst. 2013. Interpreting anaphoric shell nouns using antecedents of ctaphoric shell nouns as training data. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Ken Litowski and Orin Hargraves. 2005. The preposition project. In *ACL-SIGSEM Workshop on "The Linguistic Dimension of Prepositions and Their Use in Computational Linguistic Formalisms and Applications*, pages 171–179.

Anne Lobeck. 1995. *Ellipsis: Functional heads, licensing and identification*. Oxford University Press.

Jason Merchant. 2001. *The syntax of silence: Sluicing, islands, and identity in ellipsis*. Oxford.

Leif Nielsen. 2005. *A Corpus-Based Study of Verb Phrase Ellipsis Identification and Resolution*. Ph.D. thesis, King's College London.

Johanna Nykiel. 2010. Whatever happened to Old English sluicing. In Robert A. Cloutier, Anne Marie Hamilton-Brehm, and Jr. William A. Kretzschmar, editors, *Studies in the History of the English Language V: Variation and Change in English Grammar and Lexicon: Contemporary Approaches*, pages 37–59. Walter de Gruyter.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Hub Prüst, Remko Scha, and Martin van den Berg. 1994. A discourse perspective on verb phrase anaphora. *Linguistics and Philosophy*, 17(3):261–327.

James Pustejovesky, Marc Verhagen, Roser Sauri, Jessica Littman, Robert Gaizauskas, Graham Katz, Inderjeet Mani, Robert Knippen, and Andrea Setzer. 2016. Timebank 1.2. LDC2006T08, April.

Ivan A. Sag. 1976. *Deletion and Logical Form*. Ph.D. thesis, Massachusetts Institute of Technology. (Published 1980 by Garland Publishing, New York).

W. M. Soon, H.T. Ng, and D. C. Y Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–44.

Intra-Sentential Subject Zero Anaphora Resolution using Multi-Column Convolutional Neural Network

Ryu Iida **Kentaro Torisawa** **Jong-Hoon Oh**
Canasai Kruengkrai **Julien Kloetzer**
National Institute of Information and Communications Technology
Kyoto 619-0289, Japan
{ryu.iida,torisawa,rovellia,canasai,julien}@nict.go.jp

Abstract

This paper proposes a method for intra-sentential subject zero anaphora resolution in Japanese. Our proposed method utilizes a Multi-column Convolutional Neural Network (MCNN) for predicting zero anaphoric relations. Motivated by Centering Theory and other previous works, we exploit as clues both the surface word sequence and the dependency tree of a target sentence in our MCNN. Even though the F-score of our method was lower than that of the state-of-the-art method, which achieved relatively high recall and low precision, our method achieved much higher precision (>0.8) in a wide range of recall levels. We believe such high precision is crucial for real-world NLP applications and thus our method is preferable to the state-of-the-art method.

1 Introduction

In such pro-drop languages as Japanese, Chinese and Italian, pronouns are frequently omitted in text. For example, the subject of *uketa* (suffered) is unrealized in the following Japanese example (1):

(1) *sono-houkokusho-wa* *seifu$_i$-ga*
the report-TOP government$_i$-SUBJ
jouyaku-o *teiketsushi* *(ϕ_i-ga)* *keizaitekini*
treaty-OBJ make it$_i$-SUBJ economically
higai-o *uke-ta* *koto-o* *shitekishi-ta*
damage-OBJ suffer-PAST COMP point out-PAST
The report pointed out that the government$_i$ agreed to a treaty and (it$_i$) suffered economically.

The omitted argument is called a *zero anaphor*, which is represented using ϕ. In example (1), zero anaphor ϕ_i refers to its antecedent, *seifu$_i$* (government). Such a reference phenomenon is called *zero anaphora*. Identifying zero anaphoric relations is an essential task in developing such accurate NLP applications as information extraction and machine translation for pro-drop languages. For example, in Japanese, 60% of subjects in newspaper articles are unrealized as zero anaphors (Iida et al., 2007).

This paper proposes a method for *intra-sentential subject zero anaphora resolution*, in which a zero anaphor and its antecedent appear in the same sentence and the zero anaphor must be a *subject* of a predicate, for Japanese. We target *subject* zero anaphors because they represent 85% of the intra-sentential zero anaphora in our data set (example (1) is such a case). Furthermore, this work focuses on intra-sentential zero anaphora because inter-sentential cases, in which a zero anaphor and its antecedent do not appear in the same sentence, are extremely difficult. The accuracy of the state-of-the-art method for resolving inter-sentential anaphora is low (Sasano and Kurohashi, 2011), and we believe the current technologies are not mature enough to deal with inter-sentential cases.

Our method *locally* predicts the likelihood of a zero anaphoric relation between every possible combination of potential zero anaphor and potential antecedent without considering the other (potential) zero anaphoric relations in the same sentence. The final determination of zero anaphoric relations for each zero anaphor in a given sentence is done in a *greedy way*; only the most likely candidate antecedent for each zero anaphor is selected as its antecedent as far as the likelihood score exceeds a

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1244–1254,
Austin, Texas, November 1-5, 2016.

given threshold. This approach contrasts with *global* optimization methods (Yoshikawa et al., 2011; Iida and Poesio, 2011; Ouchi et al., 2015), which have recently become popular. These methods use the constraints among possible zero anaphoric relations, such as "if a candidate antecedent is identified as the antecedent of a subject zero anaphor of a predicate, the candidate cannot be referred to by the object zero anaphor of the same predicate", and determine an optimal set of zero anaphoric relations in an entire sentence while satisfying such constraints, using such optimization techniques as sentence-wise global learning (Ouchi et al., 2015) and integer linear programming (Iida and Poesio, 2011).

Although the global optimization methods have outperformed the previous greedy-style methods, our contention is that greedy-style methods can still, in a certain sense, outperform the state-of-the-art global optimization methods. Ouchi et al. (2015)'s global optimization method achieved the state-of-the-art F-score for Japanese intra-sentential subject zero anaphora resolution, but its performance has not yet reached a level of practical use. In our setting, for example, it actually obtained a precision of only 0.61, and even after attempting to obtain more reliable zero anaphoric relations by several modifications, we could only achieve 0.80 precision at extremely low recall levels ($<$0.01). On the other hand, while our proposed greedy-style method obtained a lower F-score than Ouchi et al.'s method, it achieved much higher precision in a wide range of recall levels (e.g., around 0.8 precision at 0.25 in recall and around 0.7 precision at 0.4 in recall). We believe such high precision is crucial to real-world applications, even though the recall remains low, and thus our method is preferable to Ouchi et al.'s method in that sense.

In our proposed method, we use a Multi-column Convolutional Neural Network (MCNN) (Cireşan et al., 2012), which is a variant of a Convolutional Neural Network (CNN) (LeCun et al., 1998). An MCNN has several independent columns, each of which has its own convolutional and pooling layers. The outputs of all the columns are combined in the final layer to provide a final prediction. In this work, motivated by Centering Theory (Grosz et al., 1995) and other previous works, we exploit as distinct columns the word sequences obtained from the surface word sequence and the dependency tree of a target sentence in our MCNN. Although the existing works also exploited such word sequences, they used only particular types of information from them as features based on the researchers' linguistic insights. In contrast, we minimized such feature engineering due to using an MCNN.

The rest of this paper is organized as follows. In Section 2, we briefly overview previous work on zero anaphora resolution. In Section 3, we present the procedure of our zero anaphora resolution method and explain the column sets used in our MCNN architecture. We evaluate how effectively our method recognizes intra-sentential subject zero anaphora in Section 4 and summarize this work and discuss future directions in Section 5.

2 Related work

The typical zero anaphora resolution algorithms proposed so far have exploited the information of a predicate that potentially has a zero anaphor and its candidate antecedent in a supervised manner (Seki et al., 2002; Iida et al., 2003; Isozaki and Hirao, 2003; Iida et al., 2006; Taira et al., 2008; Sasano et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011; Iida and Poesio, 2011; Sasano and Kurohashi, 2011; Yoshikawa et al., 2011). In addition, existing works have exploited the dependency path between a predicate and a candidate antecedent either by encoding such paths to the set of binary features of the words that appear in the path (Iida and Poesio, 2011) or by mining from the paths the sub-trees that effectively discriminate zero anaphoric relations (Iida et al., 2006). However, both methods just focus on the dependency paths between a predicate and a candidate antecedent without exploiting other structural fragments in the dependency tree representing a target sentence, whereas our method uses the text fragments that cover the entire dependency tree.

Another important clue was derived from discourse theories, such as Centering Theory (Grosz et al., 1995). In this theory, (zero) anaphoric phenomenon is explained based on the rules and principles regarding the recency and saliency of candidate antecedents. Okumura and Tamura (1996) developed a rule-based method based on the idea of Centering Theory. Iida et al. (2003) and Imamura et

al. (2009) used as features for machine learning the results of rule-based antecedent identification based on a variant of Centering Theory (Nariyama, 2002). However, we observed that actual anaphoric phenomena often do not obey Centering Theory. To robustly resolve zero anaphora, we need to explore additional clues that are represented in a target sentence (or text).

Recent work by Iida et al. (2015) newly introduced a sub-problem of zero anaphora resolution, *subject sharing recognition*, which is the task that judges whether two predicates have the same subject. In their method, a network of subject sharing predicates is created by their subject sharing recognizer, and then zero anaphora resolution is performed by propagating a subject to the unrealized subject positions through the path in the network. Even though the accuracy of subject sharing recognition exceeds that of zero anaphora resolution, the zero anaphoric relations identified using the results of subject sharing recognition are limited to those that can be reached by subject sharing relations. The recall of this method is not high.

Although most zero anaphora resolution methods independently identify a zero anaphoric relation for each predicate, some previous works optimized the global assignment of zero anaphoric relations in an entire sentence (or an entire text) while satisfying several constraints among zero anaphoric relations. For example, Iida and Poesio (2011) found the best assignment of subject zero anaphoric relations using integer linear programming. As mentioned in the Introduction, Ouchi et al. (2015) estimated the global score of all of the predicate-argument assignments in a sentence, which include the assignments of intra-sentential zero anaphoric relations, to find the best assignment using a hill-climbing technique. Their method has an advantage: it can exploit complicated relations (e.g., the combination of two potential zero anaphoric relations) as features to directly decide more than one predicate-argument relation simultaneously. We adopted Ouchi et al. (2015)'s method as a baseline in Section 4 because it achieved the state-of-the-art performance for intra-sentential zero anaphora resolution.

Collobert et al. (2011) proposed CNN architecture that can be applied to various NLP tasks, such as PoS tagging, chunking, named entity recognition and semantic role labeling. Following this work, CNNs have been utilized in such NLP tasks as document classification (Kalchbrenner et al., 2014; Kim, 2014; Johnson and Zhang, 2015), paraphrase (Hu et al., 2014; Yin and Schütze, 2015) and relation extraction (Liu et al., 2013; Zeng et al., 2014; dos Santos et al., 2015; Nguyen and Grishman, 2015). MCNNs were first introduced for image classification (Cireşan et al., 2012). In NLP tasks, they have been utilized for question-answering (Dong et al., 2015) and relation extraction (Zeng et al., 2015). Our MCNN architecture was inspired by a Siamese architecture (Chopra et al., 2005), which we extend to a multi-column network and replace its similarity measure with a softmax function at its top.

3 Proposed method

Our proposed method consists of the following four steps:

Step 1 Extract every pair of a predicate and a candidate antecedent, $\langle pred_i, cand_i \rangle$, that appears in a target sentence.

Step 2 Predict the probability of each pair using our MCNN.

Step 3 Rank in descending order all the pairs by their probabilities obtained in Step 2.

Step 4 Choose the top pair $\langle pred_i, cand_i \rangle$ in the ranked list and fill the zero anaphor position of predicate $pred_i$ by $cand_i$ if the position has not already been filled by another candidate. Remove $\langle pred_i, cand_i \rangle$ from the list and repeat this step as long as the score of the chosen pair exceeds a given threshold.

In Step 1, we extract set of pairs $\langle pred_i, cand_i \rangle$ in which candidate antecedent $cand_i$ is paired with predicate $pred_i$. Note that we extracted predicate $pred_i$, instead of a zero anaphor that is an unrealized subject of $pred_i$, because the (potential) zero anaphor of $pred_i$ is omitted in the text and cannot be extracted directly.

In Step 2, our MCNN gives a probability that indicates the likelihood of a zero anaphoric relation to judge for each pair whether $cand_i$ fills the blank subject position of $pred_i$ through zero anaphora and ranks all of the pairs by the probabilities in Step 3.

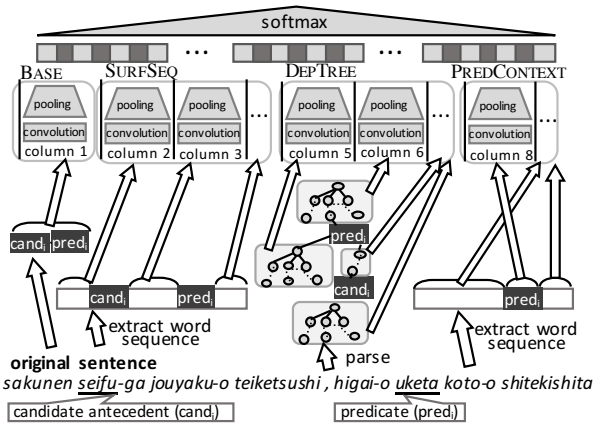


Figure 1: Our multi-column CNN architecture

Finally, in Step 4 we actually fill $cand_i$ in the blank subject positions of $pred_i$ in a greedy style in the order of the ranked list in Step 3, i.e., the zero anaphora resolution with a higher probability is done before that with a lower probability. If the subject position is already occupied by another candidate antecedent, candidate antecedents are no longer filled at that position.

3.1 Design of columns used in MCNN

In Step 2 of our method, we use a Multi-column Convolutional Neural Network (MCNN). Note that zero anaphoric phenomena can be divided into two different referential phenomena: *anaphoric* (i.e., an antecedent precedes its zero anaphor) and *cataphoric* (i.e., a zero anaphor precedes its antecedent) cases. To capture this difference, we divided the set of training instances into two subsets by the relative occurrence positions of a predicate and a candidate antecedent and respectively trained two independent MCNNs using each set.

Our MCNN simultaneously uses four column sets, as illustrated in Figure 1. In the following explanation for each column set, we assume that candidate antecedent $cand_i$ precedes predicate $pred_i$ in the surface order (for the opposite case, i.e., the cataphoric case, the positions of $cand_i$ and $pred_i$ are switched).

BASE The first column set consists of one column, which stores the word vectors of the *bunsetsu* phrases[1] including either $cand_i$ or $pred_i$. We call this column set the BASE column set.

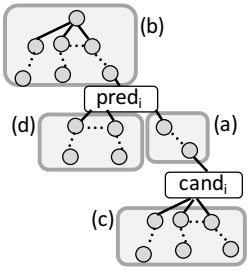


Figure 2: Columns (a, b, c, d) in DEPTREE column set

SURFSEQ The second column set consists of three columns, which store the word vectors of (a) the surface word sequence spanning from the beginning of the sentence to $cand_i$, (b) the sequence between $cand_i$ and $pred_i$, and (c) the remainder, i.e., from $pred_i$ to the end of the sentence. Note that $cand_i$ and $pred_i$ are not included in any column of this column set. We call this column set the SURFSEQ column set.

DEPTREE The third set consists of four columns. We extracted four partial dependency trees from the entire dependency tree of a target sentence: (a) the dependency path between $pred_i$ and $cand_i$, (b) the sub-trees that depend on $pred_i$, (c) the sub-trees on which $cand_i$ depends and (d) the remaining sub-trees, which are illustrated in Figure 2. Note that $cand_i$ and $pred_i$ are not included in the partial trees. Each column stores the word vectors of the word sequence in which the words in (the set of) the partial trees are ordered by their surface order. We call this set the DEPTREE column set.

PREDCONTEXT The fourth set consists of three columns, which store the word vectors of (a) the bunsetsu phrase including $pred_i$, (b) the surface word sequence that appears before (a) (from the beginning of the sentence) and (c) the sequence that appears after (a) (until the end of the sentence). We call this column set the PREDCONTEXT column set.

Among the four column sets, the SURFSEQ column set was designed to introduce the clues based

[1] A bunsetsu phrase is a Japanese base phrase that consists of at least one content word optionally followed by function words.

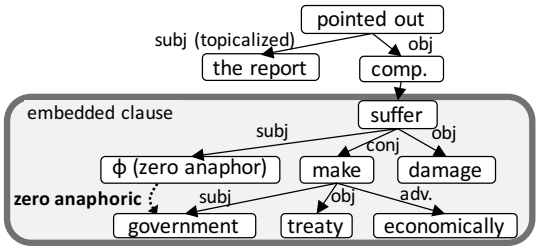


Figure 3: Dependency tree of example (1)

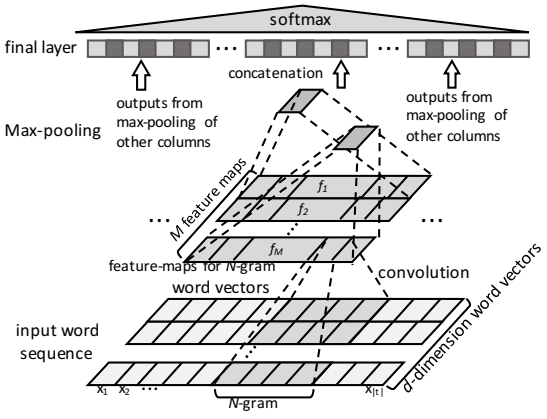


Figure 4: Column of our MCNN

on Centering Theory, in which the antecedent for a given zero anaphor can basically be identified by the *recency* and *saliency* properties of a candidate antecedent. More precisely, in the set of the most salient candidate antecedents, the most recent one is preferred. For example, suppose example (2) in which the predicate *increase* has a subject zero anaphor and its antecedent is *France*:

(2) *nihon-wa* *shoshikataisaku-ni*
Japan-TOP countermeasures to falling birth rate-IOBJ
shippaishi-taga, *furansu-wa* *sore-ni* *seikoushi*
fail-PAST/BUT France-TOP it-IOBJ succeed
(ϕ_i-ga) *shusseiritsu-o* *fuyashiteiru*
(it$_i$-SUBJ) birth rate-OBJ increase
Japan failed to develop countermeasures to its falling birth rate, but France$_i$ succeeded and (ϕ_i) increased its birth rate.

In this situation, there are two most salient candidate antecedents, *Japan* and *France*, because they are marked with topic marker *wa*, which basically indicates the highest degree of candidate saliency. In this case, *France* is selected as the antecedent because it appears more recently than *Japan*, and such recency can be estimated by consulting the surface word sequence between *France* and *increase*: no other salient candidates are included in the word sequence. Also, the other two types of word sequences (i.e., the sequence that spans from the beginning of the sentence to $cand_i$ and that spans from $pred_i$ to its end) are important for confirming whether a more salient candidate than $cand_i$ appears in each word sequence. If such a more salient candidate is found, it should be a stronger candidate of the antecedent.

The DEPTREE column set is introduced for capturing a different aspect of intra-sentential zero anaphora. In the explanation based on Centering Theory, the most salient candidate (e.g., the candidate marked with *wa* (topic marker)) is selected as an antecedent, but example (1) in Section 1 cannot be interpreted based on saliency and recency. In example (1), the *report* is the most salient candidate in the sentence because it is marked with topic marker *wa*, but the less salient candidate *government* becomes the antecedent of zero anaphor ϕ. Such a problem is often solved by introducing the dependency tree of a sentence. Figure 3 represents the dependency tree of example (1) in which the antecedent of ϕ_i appears in the embedded clause. In such a case, an antecedent probably exists among the most salient candidates in the embedded clause. To introduce such structural clues, we used the partial dependency trees as columns in the DEPTREE column set.

Anaphoricity determination, which is the task of judging whether a candidate anaphor has an antecedent, was established as a subtask of coreference resolution. This problem was basically solved by exploring the possible candidate antecedents for a given anaphor candidate in its search space, and the results were used for improving the overall performance of coreference resolution, especially in English (Ng, 2004; Wiseman et al., 2015). Inspired by such previous works, we designed the PRED-CONTEXT set to determine the anaphoricity of zero anaphors, i.e., to judge whether a zero anaphor candidate has its antecedent in a sentence, by consulting the surface word sequences before and after $pred_i$.

3.2 MCNN architecture

In our MCNN (Figure 4), we represent each word in text fragment t by d-dimensional embedding vec-

tor x_i and t by matrix $\mathbf{T} = [\mathrm{x}_1, \ldots, \mathrm{x}_{|t|}]$.[2] $\mathbf{T}$ is then wired to a set of M feature maps where each feature map is a vector. Each element O in the feature map is computed by a filter denoted by f_j ($1 \leq j \leq M$) from the N-gram word sequences in t for a fixed integer N, as $O = \mathrm{ReLU}(\mathbf{W}_{f_j} \bullet \mathrm{x}_{i:i+N-1} + b_{f_j})$, where $\bullet$ denotes element-wise multiplication followed by the summation of the resulting elements (i.e., a Frobenious inner product of $\mathbf{W}_{f_j}$ and $\mathrm{x}_{i:i+N-1}$) and $\mathrm{ReLU}(x) = \max(0, x)$. In other words, we construct a feature map by convolving a text fragment with a filter, which is parameterized by weight $\mathbf{W}_{f_j} \in \mathbb{R}^{d \times N}$ and bias $b_{f_j} \in \mathbb{R}$. Note that there can be several sets of feature maps where each set covers N-grams for different N. Note that the weight of the feature maps for each N-gram in each column set is shared.

As a whole, these feature maps are referred to as a *convolution layer*. The next layer is called a *pooling layer*. Here we use max-pooling (Scherer et al., 2010; Collobert et al., 2011), which simply selects the maximum value among the elements in the same feature map. Our assumption is that the maximum value indicates the existence of a strong clue, i.e., N-gram, for our final judgment. The selected maximum values from all the M feature maps are simply concatenated, and the resulting M-dimensional vector is given to our final layer.

The final layer has vectors coming from multiple feature maps in multiple columns. They are again simply concatenated and constitute a high dimensional feature vector. The final layer applies a linear softmax function to produce the class probabilities of the zero anaphoric labels: *true* and *false*. We use a mini-batch stochastic gradient descent (SGD) with the Adadelta update rule (Zeiler, 2012), apply random initialization within (-0.01, 0.01) for $\mathbf{W}_{f_j}$, and initialize the remaining parameters at zero.

4 Experiments

4.1 Revising annotation results

In our preliminary investigation of the intra-sentential zero anaphoric relations in the NAIST Text Corpus (Iida et al., 2007), since we found more annotation errors than we expected, we decided to revise the annotation results. In this revision, we additionally annotated the subject sharing relations, where two predicates have the same subject regardless whether the subject is realized or omitted, between pairs of predicates in our data set. Note that two predicates can have a subject sharing relation even if neither has a realized subject as far as a subject exists that can naturally fill the subject position of the two predicates. We used the annotated results of subject sharing relations to efficiently detect the annotation errors of intra-sentential zero anaphoric relations, as shown below.

Twenty-six human annotators directly annotated the subject sharing relations for pairs of predicates in a sentence. For this annotation, we automatically extracted from the NAIST Text Corpus all the pairs of predicates that appear in the same sentence and obtained 227,517 predicate pairs. For making the annotation results more reliable, each subject sharing relation was individually judged by three annotators, and the final label was decided by a majority vote. After that, further revisions of the subject sharing relations and the zero anaphoric relations were performed by focusing on the inconsistent annotations between the newly annotated subject sharing relations and the original predicate-argument relations in the NAIST Text Corpus. More precisely, we scrutinized the suspicious annotations such that a subject, which was determined through the annotated subject sharing relations, is not the same as a subject that was directly annotated in the NAIST Text Corpus. In this revision phase, both the subject sharing and zero anaphora relations for such suspicious instances were independently re-annotated by three annotators, and their final labels of both relations were determined by a majority of the their decisions.[3] As a result, 2,120 zero anaphoric instances were newly added to the corpus and 1,184 instances were removed from it for a total of 19,049 instances of intra-sentential subject zero anaphoric relations.[4]

[2] We use zero padding for dealing with text fragments of variable length (Kim, 2014).

[3] We are planning to release the annotated results and information on the data separation used in our evaluation from https://alaginrc.nict.go.jp/.

[4] After this revision, a small number of inconsistent annotated results have both a syntactically dependent subject and a subject zero anaphor because the revision was performed locally. There were 30 inconsistent instances in the testing set and 100 in the training and development sets. We only removed such instances from the testing set without changing the other

Type	#docs	#sentences	#zero anaphors (intra-sentential)
train	1,757	23,152	11,453
dev	586	7,526	3,691
test	586	7,705	3,875

Table 1: Statistics of our data set

4.2 Experimental settings

The documents in the corpus were divided into five subsets, three of which were used as a training data set, one as a development data set, and one as a testing data set. The statistics of our data set are summarized in Table 1. We evaluated the performance of our intra-sentential subject zero anaphora resolution method and three baseline methods described below using the revised annotated results in our data set.

We implemented our MCNN using Theano (Bastien et al., 2012). We pre-trained 300-dimensional word embedding vectors for 1,658,487 words[5] using Skip-gram with a negative-sampling algorithm (Mikolov et al., 2013)[6] on a set of all the sentences extracted from Wikipedia articles[7] (35,975,219 sentences). We removed from the training data all the words that only appeared once before training. In training, we treated them as unknown words and assigned them a random vector. To avoid overfitting, we applied early-stopping and dropout (Hinton et al., 2012) of 0.5 to the final layer. We used an SGD with mini-batches of 100 and a learning rate decay of 0.95. We ran ten epochs through all of the training data, where each epoch consisted of many mini-batch updates. We utilized 3-, 4- and 5-grams with 100 filters each and used the F-score of positive instances as our evaluation metric. The total number of the nodes in the final layers of our MCNN was 3,300: 11 columns $\times$ 3 N-gram $\times$ 100 filters.

Word segmentation, PoS tagging and dependency parsing of the sentences in the NAIST Text Corpus were performed by a Japanese morphological analyzer, MeCab[8] (Kudo et al., 2004), and a dependency parser, J.DepP[9] (Yoshinaga and Kitsuregawa, 2009).

4.3 Baselines

We compared our method with three baseline methods. The first baseline is a single-column convolutional neural network in which the column includes the entire surface word sequence of a sentence. To give the positions of $pred_i$ and $cand_i$ to the network, we concatenated to each word vector an additional 2-dimensional vector, where the first element is set to one if the corresponding word is $pred_i$, the second element is set to 1 if the corresponding word is $cand_i$, and otherwise they are set to 0. This baseline was adopted for estimating the impact of a multi-column network compared to a single-column one.

The remaining two baselines are Ouchi et al. (2015)'s global optimization method and Iida et al. (2015)'s method based on subject sharing recognition. Note that Ouchi's method outputs predicate-argument relations for three grammatical roles (subj, obj, iobj), but for this evaluation we used only the outputs related to intra-sentential subject zero anaphora resolution. As done in Ouchi et al. (2015), we averaged their performances across ten independent runs because the initial random assignment of the predicate-argument relations that was employed in their method changes the performance. Ouchi's method does not require any development data set, so we used both the development and training data sets for training their joint model. For training the subject sharing recognizer used in Iida's method, we used the annotated subject sharing relations in the training and development data sets. In these two baselines, we used the same morphological analyzer and dependency analyzer as for our method.

4.4 Results

Table 2 shows the results for each method. Their performances were evaluated by measuring recall, precision, F-score and average precision (Avg.P). To assess the effectiveness of each column set introduced in Section 3.1, we evaluated the performance of our method using every possible combination of column sets that includes at least the BASE column set. We also gave the precision-recall (PR)

two sets.

[5]Words occurring less than five times in all the sentences were ignored to train the word embedding vectors.

[6]We set the skip distance to 5 and the number of negative samples to 10.

[7]https://archive.org/details/jawiki-20150118

[8]http://taku910.github.io/mecab/

[9]http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/

Method		#cols.	Recall	Precision	F-score	Avg.P
Ouchi et al. (ACL2015)		—	**0.539**	0.612	**0.573**	0.670
Iida et al. (EMNLP2015)		—	0.484	0.357	0.411	—
single column CNN (w/ position vec.)		1	0.365	0.524	0.430	0.540
MCNN	BASE	1	0.446	0.394	0.419	0.448
	BASE+SURFSEQ	4	0.458	0.597	0.518	0.679
	BASE+DEPTREE	5	0.339	0.688	0.454	0.690
	BASE+SURFSEQ+DEPTREE	8	0.417	0.695	0.521	0.730
	BASE+SURFSEQ+PREDCONTEXT	7	0.459	0.631	0.531	0.702
	BASE+DEPTREE+PREDCONTEXT	8	0.298	**0.728**	0.422	0.702
	BASE+SURFSEQ+DEPTREE+PREDCONTEXT (**Proposed**)	11	0.418	0.704	0.525	**0.732**

#cols. stands for the number of columns used in each MCNN.

Table 2: Results of intra-sentential subject zero anaphora resolution

curves of our method using the four column sets (BASE+SURFSEQ+DEPTREE+PREDCONTEXT), the single column baseline, and Ouchi's method in Figure 5 to investigate the behavior of each method at a high precision level.[10] The PR-curves of our method and the single-column baseline were plotted just by altering the threshold parameters in Step 4 of our method (See Section 3). In contrast, the PR-curve of Ouchi's method cannot be easily plotted because it gives a score to each sentence, not to each zero anaphoric relation. For plotting the PR-curve, we used the normalized global score of a sentence as the score of any zero anaphoric relations in the sentence.[11] Note that the recall of their PR-curve reached just 0.539, shown in Table 2, because we could not estimate the scores of the zero anaphoric relations that were not outputted by their method. The PR-curves of the other methods also fail to reach 1.0 in recall. This is because the zero anaphoric relations are exclusive; a zero anaphor does not refer to more than one antecedent. If a method provides an incorrect zero anaphoric relation, a correct relation for the same zero anaphor will never be provided in its output. Also, note that the average precision of each method was calculated by averaging the precisions at the available recall levels for each method.

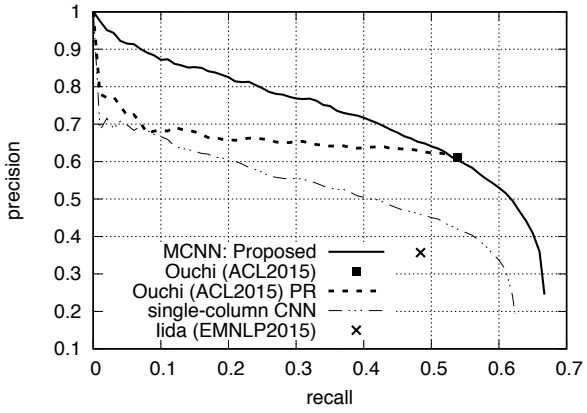


Figure 5: PR-curves of each method

The results in Table 2 show that our method using all the column sets achieved the best average precision among the combination of column sets that include at least the BASE column set. This suggests that all of the clues introduced by our four column sets are effective for performance improvement. Table 2 also demonstrates that our method using all the column sets obtained better average precision than the strongest baseline, Ouchi's method, in spite of an unfavorable condition for it.[12] The results also show that our method with all of the column sets achieved a better F-score than Iida's method and the single-column baseline. However, it achieved a lower F-score than Ouchi's method. This was caused by the choice of different recall levels for computing the F-score. In contrast, the PR-

[10] The PR-curve of Iida et al. (2015)'s method was not plotted because it does not provide the score of each zero anaphoric relation.

[11] The global score provided by Ouchi's method becomes greater based on the number of predicate-argument pairs in a sentence. To control this, we normalized the original global score by the sum of the frequencies of the single or double predicate-argument pairs because the feature functions were applied to such pairs in their method. This achieved the best performance among the normalization schemes we have tried so far.

[12] When calculating the average precision of each method, the relatively low values in precision at high recall levels (i.e., from 0.54 to 0.67) were used in our method but not in Ouchi's method, as seen in Figure 5.

Set	Method	Recall	Precision	F-score	Avg.P
Anaphoric	single-column CNN (w/ position vec.)	0.445	0.525	0.481	0.341
	MCNN (BASE)	**0.591**	0.330	0.424	0.367
	MCNN (BASE+SURFSEQ)	0.555	0.566	0.560	0.565
	MCNN (BASE+DEPTREE)	0.389	0.615	0.476	0.518
	MCNN (BASE+SURFSEQ+DEPTREE)	0.503	0.660	**0.571**	0.599
	MCNN (BASE+SURFSEQ+PREDCONTEXT)	0.535	0.611	0.570	0.581
	MCNN (BASE+DEPTREE+PREDCONTEXT)	0.330	**0.699**	0.449	0.528
	MCNN (**Proposed**)	0.492	0.673	0.569	**0.602**
Cataphoric	single-column CNN (w/ position vec.)	0.163	0.293	0.209	0.163
	MCNN (BASE)	0.171	0.130	0.148	0.099
	MCNN (BASE+SURFSEQ)	0.202	0.417	0.272	0.257
	MCNN (BASE+DEPTREE)	**0.268**	0.438	0.332	0.329
	MCNN (BASE+SURFSEQ+DEPTREE)	0.195	**0.525**	0.285	0.330
	MCNN (BASE+SURFSEQ+PREDCONTEXT)	0.258	0.406	0.316	0.276
	MCNN (BASE+DEPTREE+PREDCONTEXT)	0.240	0.488	0.322	0.341
	MCNN (**Proposed**)	0.251	0.522	**0.339**	**0.337**

Table 3: Results of instance-wise evaluation for anaphoric and cataphoric sets

curves for these two methods in Figure 5 show that our method obtained higher precision than Ouchi's method at all recall levels. Particularly, it got high precision in a wide range of recall levels (e.g., around 0.8 in precision at 0.25 in recall and around 0.7 in precision at 0.4 in recall), while the precision obtained by Ouchi's method at 0.25 in recall was just around 0.65. We believe this difference becomes crucial when using the outputs of each method for developing accurate real-world NLP applications.

In addition to an evaluation that used all of the test instances, we also investigated how our method performed differently for anaphoric and cataphoric cases. In this evaluation, we first divided our data set into anaphoric and cataphoric sets by the relative position of the candidate antecedent and evaluated the performance by measuring the recall, precision, F-score and average precision for each set. This evaluation was done instance-wise, where we took into account each pair of a predicate and its candidate antecedent as a classification target, while in the previous evaluation the performance was measured for the set of zero anaphors in the test set. Thus, the figures in Table 2 and Table 3 are not comparable. Note that we only compared our method with the baseline using a single-column convolutional neural network because the other baselines are not able to output the score of each instance for measuring their average precision.

The results in Table 3 show that our MCNN-based method achieved better average precision than the single-column CNN baseline except the method that uses only the BASE column set for the cataphoric case. The results also demonstrate that each column set consistently contributes to improving the average precision for both the anaphoric and cataphoric cases. However, Table 3 shows that the average precision for the cataphoric set remains low. As one future direction for further improvement, we need to explore clues for identifying cataphoric relations more accurately.

5 Conclusion

This paper proposed an accurate method for intra-sentential subject zero anaphora resolution using a Multi-column Convolutional Neural Network (MCNN). As clues, our MCNN exploits both the surface word sequence and the dependency tree of a target sentence. Our experimental results show that the proposed method achieved better precision than the strong baselines in a wide range of recall levels.

As future work, we plan to use our MCNN architecture for inter-sentential zero anaphora resolution and develop highly accurate NLP applications using our intra-sentential subject zero anaphora resolution method.

Acknowledgement

We thank Hiroki Ouchi for providing his predicate-argument analyzer that was proposed in Ouchi et al. (2015).

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. In *Proceedings of the NIPS 2012 Workshop: Deep Learning and Unsupervised Feature Learning*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 539–546.

Dan Claudiu Cireşan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition*, pages 3642–3649.

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

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 260–269.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 626–634.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201–209.

Geoffrey E . Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs /1207.0580.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Processings of Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2042–2050.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 804–813.

Ryu Iida, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the 2003 EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Processings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 625–632.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the ACL Workshop: 'Linguistic Annotation Workshop'*, pages 132–139.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2179–2189.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, pages 85–88.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 184–191.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese

morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Chunyang Liu, Wenbo Sun, Wenhan Chao, and Wanxiang Che. 2013. Convolution neural network for relation extraction. In *Proceedings of the 9th International Conference of Advanced Data Mining and Applications*, pages 231–242.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119.

Shigeko Nariyama. 2002. Grammar for ellipsis resolution in Japanese. In *Proceedings of the 9th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 135–145.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 151–158.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Manabu Okumura and Kouji Tamura. 1996. Zero pronoun resolution in Japanese discourse based on centering theory. In *Proceedings of The 16th International Conference on Computational Linguistics*, pages 871–876.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 961–970.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 758–766.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for Japanese zero anaphora resolution. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 769–776.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks*, pages 92–101.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Processings of the 19th International Conference on Computational Linguistics*, pages 1–7.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 523–532.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1416–1426.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with Markov Logic. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1125–1133.

Naoki Yoshinaga and Masaru Kitsuregawa. 2009. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1542–1551.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. In *arXiv:1212.5701 (Dec 27, 2012)*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2335–2344.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

An Unsupervised Probability Model for Speech-to-Translation Alignment of Low-Resource Languages

Antonios Anastasopoulos and David Chiang
University of Notre Dame
{aanastas, dchiang}@nd.edu

Long Duong
University of Melbourne
lduong@student.unimelb.edu.au

Abstract

For many low-resource languages, spoken language resources are more likely to be annotated with translations than with transcriptions. Translated speech data is potentially valuable for documenting endangered languages or for training speech translation systems. A first step towards making use of such data would be to automatically align spoken words with their translations. We present a model that combines Dyer et al.'s reparameterization of IBM Model 2 (`fast_align`) and k-means clustering using Dynamic Time Warping as a distance measure. The two components are trained jointly using expectation-maximization. In an extremely low-resource scenario, our model performs significantly better than both a neural model and a strong baseline.

1 Introduction

For many low-resource languages, speech data is easier to obtain than textual data. And because speech transcription is a costly and slow process, speech is more likely to be annotated with translations than with transcriptions. This translated speech is a potentially valuable source of information – for example, for documenting endangered languages or for training speech translation systems.

In language documentation, data is usable only if it is interpretable. To make a collection of speech data usable for future studies of the language, something resembling interlinear glossed text (transcription, morphological analysis, word glosses, free translation) would be needed at minimum. New technologies are being developed to facilitate collection of translations (Bird et al., 2014), and there already exist recent examples of parallel speech collection efforts focused on endangered languages (Blachon et al., 2016; Adda et al., 2016). As for the other annotation layers, one might hope that a first pass could be done automatically. A first step towards this goal would be to automatically align spoken words with their translations, capturing information similar to that captured by word glosses.

In machine translation, statistical models have traditionally required alignments between the source and target languages as the first step of training. Therefore, producing alignments between speech and text would be a natural first step towards MT systems operating directly on speech.

We present a model that, in order to learn such alignments, adapts and combines two components: Dyer et al.'s reparameterization of IBM Model 2 (Dyer et al., 2013), more commonly known as `fast_align`, and k-means clustering using Dynamic Time Warping (Berndt and Clifford, 1994) as a distance measure. The two components are trained jointly using expectation-maximization.

We experiment on two language pairs. One is Spanish-English, using the CALLHOME and Fisher corpora. The other is Griko-Italian; Griko is an endangered language for which we created (and make freely available)[1] gold-standard translations and word alignments (Lekakou et al., 2013). In all cases, our model outperforms both a naive but strong baseline and a neural model (Duong et al., 2016).

[1] https://www3.nd.edu/~aanastas/griko/griko-data.tar.gz

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1255–1263,
Austin, Texas, November 1-5, 2016.

2 Background

In this section, we briefly describe the existing models that the two components of our model are based on. In the next section, we will describe how we adapt and combine them to the present task.

2.1 IBM Model 2 and `fast_align`

The IBM translation models (Brown et al., 1993) aim to model the distribution $p(\mathbf{e} \mid \mathbf{f})$ for an English sentence $\mathbf{e} = e_1 \cdots e_l$, given a French sentence $\mathbf{f} = f_1 \cdots e_m$. They all introduce a hidden variable $\mathbf{a} = a_1 \cdots a_l$ that gives the position of the French word to which each English word is aligned.

The general form of IBM Models 1, 2 and `fast_align` is

$$p(\mathbf{e}, \mathbf{a} \mid \mathbf{f}) = p(l) \prod_{i=1}^{l} t(e_i \mid f_{a_i})\, \delta(a_i \mid i, l, m)$$

where $t(e \mid f)$ is the probability of translating French word f to English word e, and $\delta(a_i = j \mid i, l, m)$ is the probability of aligning the i-th English word with the j-th French word.

In Model 1, δ is uniform; in Model 2, it is a categorical distribution. Dyer et al. (2013) propose a reparameterization of Model 2, known as `fast_align`:

$$h(i, j, l, m) = -\left|\frac{i}{l} - \frac{j}{m}\right|$$

$$\delta(a_i \mid i, l, m) = \begin{cases} p_0 & a_i = 0 \\ (1 - p_0)\frac{\exp \lambda h(i, a_i, l, m)}{Z_\lambda(i, l, m)} & a_i > 0 \end{cases}$$

where the null alignment probability p_0 and precision $\lambda \geq 0$ are hyperparameters optimized by grid search. As $\lambda \to 0$, the distribution gets closer to the distribution of IBM Model 1, and as λ gets larger, the model prefers monotone word alignments more strongly.

2.2 DTW and DBA

Dynamic Time Warping (DTW) (Berndt and Clifford, 1994) is a dynamic programming method for measuring distance between two temporal sequences of variable length, as well as computing an alignment based on this distance. Given two sequences ϕ, ϕ' of length m and m' respectively, DTW constructs an $m \times m'$ matrix w. The warping path can be found by evaluating the following recurrence:

$$w_{i,j} = d(\phi_i, \phi'_j) + \min\{w_{i-1,j}, w_{i-1,j-1}, w_{i,j-1}\}$$

where d is a distance measure. In this paper, we normalize the cost of the warping path:

$$\mathrm{DTW}(\phi, \phi') = \frac{w_{m,m'}}{m + m'}$$

which lies between zero and one.

DTW Barycenter Averaging (DBA) (Petitjean et al., 2011) is an iterative approximate method that attempts to find a centroid of a set of sequences, minimizing the sum of squared DTW distances.

In the original definition, given a set of sequences, DBA chooses one sequence randomly to be a "skeleton." Then, at each iteration, DBA computes the DTW between the skeleton and every sequence in the set, aligning each of the skeleton's points with points in all the sequences. The skeleton is then refined using the found alignments, by updating each frame in the skeleton to the mean of all the frames aligned to it. In our implementation, in order to avoid picking a skeleton that is too short or too long, we randomly choose one of the sequences with median length.

3 Model

We use a generative model from a source-language speech segment consisting of feature frames $\boldsymbol{\phi} = \phi_1 \cdots \phi_m$ to a target-language segment consisting of words $\mathbf{e} = e_1 \ldots e_l$. We chose to model $p(\mathbf{e} \mid \boldsymbol{\phi})$ rather than $p(\boldsymbol{\phi} \mid \mathbf{e})$ because it makes it easier to incorporate DTW. The other direction is also possible, and we plan to explore it in future work.

In addition to the target-language sentence $\mathbf{e}$, our model hypothesizes a sequence $\mathbf{f} = f_1 \cdots f_l$ of source-language clusters (intuitively, source-language words), and spans (a_i, b_i) of the source signal that each target word e_i is aligned to. Thus, the clusters $\mathbf{f} = f_1 \cdots f_l$ and the spans $\mathbf{a} = a_1, \ldots, a_l$ and $\mathbf{b} = b_1, \ldots, b_l$ are the hidden variables of the model:

$$p(\mathbf{e} \mid \boldsymbol{\phi}) = \sum_{\mathbf{a},\mathbf{b},\mathbf{f}} p(\mathbf{e}, \mathbf{a}, \mathbf{b}, \mathbf{f} \mid \boldsymbol{\phi}).$$

The model generates $\mathbf{e}, \mathbf{a}, \mathbf{b}$, and $\mathbf{f}$ from $\boldsymbol{\phi}$ as follows.

1. Choose l, the number of target words, with uniform probability. (Technically, this assumes a maximum target sentence length, which we can just set to be very high.)

2. For each target word position $i = 1, \ldots, l$:
   (a) Choose a cluster f_i.
   (b) Choose a span of source frames (a_i, b_i) for e_i to be aligned to.
   (c) Generate a target word e_i from f_i.

Accordingly, we decompose $p(\mathbf{e}, \mathbf{a}, \mathbf{b}, \mathbf{f} \mid \boldsymbol{\phi})$ into several submodels:

$$p(\mathbf{e}, \mathbf{a}, \mathbf{b}, \mathbf{f} \mid \boldsymbol{\phi}) = p(l) \prod_{i=1}^{l} u(f_i) \times s(a_i, b_i \mid f_i, \boldsymbol{\phi}) \times \delta(a_i, b_i \mid i, l, |\boldsymbol{\phi}|) \times t(e_i \mid f_i).$$

Note that submodels δ and s both generate spans (corresponding to step 2b), making the model deficient. We could make the model sum to one by replacing $u(f_i)s(a_i, b_i \mid f_i, \boldsymbol{\phi})$ with $s(f_i \mid a_i, b_i, \boldsymbol{\phi})$, and this was in fact our original idea, but the model as defined above works much better, as discussed in Section 7.4. We describe both δ and s in detail below.

Clustering model The probability over clusters, $u(f)$, is just a categorical distribution. The submodel s assumes that, for each cluster f, there is a "prototype" signal $\boldsymbol{\phi}^f$ (cf. Ristad and Yianilos, 1998). Technically, the $\boldsymbol{\phi}^f$ are parameters of the model, and will be recomputed during the M step. Then we can define:

$$s(a, b \mid f, \boldsymbol{\phi}) = \frac{\exp(-\mathrm{DTW}(\boldsymbol{\phi}^f, \phi_a \cdots \phi_b)^2)}{\sum_{a,b=1}^{m} \exp(-\mathrm{DTW}(\boldsymbol{\phi}^f, \phi_a \cdots \phi_b)^2)}$$

where DTW is the distance between the prototype and the segment computed using Dynamic Time Warping. Thus s assigns highest probability to spans of $\boldsymbol{\phi}$ that are most similar to the prototype $\boldsymbol{\phi}^f$.

Distortion model The submodel δ controls the reordering of the target words relative to the source frames. It is an adaptation of `fast_align` to our setting, where there is not a single source word position a_i, but a span (a_i, b_i). We want the model to prefer the middle of the word to be close to the diagonal, so we need the variable a to be somewhat to the left and b to be somewhat to the right. Therefore, we introduce an additional hyperparameter μ which is intuitively the number of frames in a word. Then we define

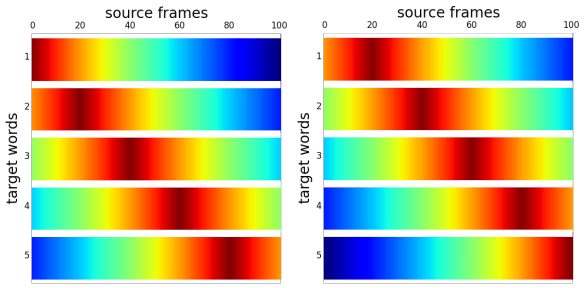


Figure 1: Sample distributions for the alignment variables a and b for $m = 100$, $l = 5$, $p_0 = 0$, $\lambda = 0.5$, and $\mu = 20$.

$$h_a(i, j, l, m, \mu) = -\left|\frac{i}{l} - \frac{j}{m - \mu}\right|$$

$$h_b(i, j, l, m, \mu) = -\left|\frac{i}{l} - \frac{j - \mu}{m - \mu}\right|$$

$$\delta_a(a_i \mid i, l, m) = \begin{cases} p_0 & a_i = 0 \\ (1 - p_0)\frac{\exp \lambda h_a(i, a_i, l, m)}{Z_\lambda(i, l, m)} & a_i > 0 \end{cases}$$

$$\delta_b(b_i \mid i, l, m) = \begin{cases} p_0 & b_i = 0 \\ (1 - p_0)\frac{\exp \lambda h_b(i, b_i, l, m)}{Z_\lambda(i, l, m)} & b_i > 0 \end{cases}$$

$$\delta(a_i, b_i \mid i, l, m) = \delta_a(a_i \mid i, l, m)\, \delta_b(b_i \mid i, l, m)$$

where the $Z_\lambda(i, l, m)$ are set so that all distributions sum to one. Figure 1 shows an example visualisation of the the resulting distributions for the two variables of our model.

We set μ differently for each word. For each i, we set μ_i to be proportional to the number of *characters* in e_i, such that $\sum_i \mu_i = m$.

Translation model The translation model $t(e \mid f)$ is just a categorical distribution, in principle allowing a many-to-many relation between source clusters and target words. To speed up training (with nearly no change in accuracy, in our experiments), we restrict this relation so that there are k source clusters for each target word, and a source cluster uniquely determines its target word. Thus, $t(e \mid f)$ is fixed to

either zero or one, and does not need to be reestimated. In our experiments, we set $k = 2$, allowing each target word to have up to two source-language translations/pronunciations. (If a source word has more than one target translation, they are treated as distinct clusters with distinct prototypes.)

4 Training

We use the hard (Viterbi) version of the Expectation-Maximization (EM) algorithm to estimate the parameters of our model, because calculating expected counts in full EM would be prohibitively expensive, requiring summations over all possible alignments.

Recall that the hidden variables of the model are the alignments (a_i, b_i) and the source words (f_i). The parameters are the translation probabilities $t(e_i \mid f)$ and the prototypes $(\boldsymbol{\phi}^f)$. The (hard) E step uses the current model and prototypes to find, for each target word, the best source segment to align it to and the best source word. The M step reestimates the probabilities $t(e \mid f)$ and the prototypes $\boldsymbol{\phi}^f$. We describe each of these steps in more detail below.

Initialization Initialization is especially important since we are using hard EM.

To initialize the parameters, we initialize the hidden variables and then perform an M step. We associate each target word type e with $k = 2$ source clusters, and for each occurrence of e, we randomly assign it one of the k source clusters.

The alignment variables a_i, b_i are initialized to

$$a_i, b_i = \arg\max_{a,b} \delta(a, b \mid i, l, m).$$

M step The M step reestimates the probabilities $t(e \mid f)$ using relative-frequency estimation.

The prototypes $\boldsymbol{\phi}^f$ are more complicated. Theoretically, the M step should recompute each $\boldsymbol{\phi}^f$ so as to maximize that part of the log-likelihood that depends on $\boldsymbol{\phi}^f$:

$$\begin{aligned} L_{\boldsymbol{\phi}^f} &= \sum_{\boldsymbol{\phi}} \sum_{i|f_i=f} \log s(a_i, b_i \mid f, \boldsymbol{\phi}) \\ &= \sum_{\boldsymbol{\phi}} \sum_{i|f_i=f} \log \frac{\exp(-\mathrm{DTW}(\phi^f, \phi_{a_i} \cdots \phi_{b_i})^2)}{Z(f, \boldsymbol{\phi})} \\ &= \sum_{\boldsymbol{\phi}} \sum_{i|f_i=f} -\mathrm{DTW}(\phi^f, \phi_{a_i} \cdots \phi_{b_i})^2 - \log Z(f, \boldsymbol{\phi}) \end{aligned}$$

where the summation over $\boldsymbol{\phi}$ is over all source signals in the training data. This is a hard problem, but note that the first term is just the sum-of-squares of the DTW distance between $\boldsymbol{\phi}^f$ and all source segments that are classified as f. This is what DBA is supposed to approximately minimize, so we simply set $\boldsymbol{\phi}^f$ using DBA, ignoring the denominator.

E step The (hard) E step uses the current model and prototypes to find, for each target word, the best source segment to align it to and the best source cluster.

In order to reduce the search space for **a** and **b**, we use the unsupervised phonetic boundary detection method of Khanagha et al. (2014). This method operates directly on the speech signal and provides us with candidate phone boundaries, on which we restrict the possible values for **a** and **b**, creating a list of candidate utterance spans.

Furthermore, we use a simple silence detection method. We pass the envelope of the signal through a low-pass filter, and then mark as "silence" time spans of 50ms or longer in which the magnitude is below a threshold of 5% relative to the maximum of the whole signal. This method is able to detect about 80% of the total pauses, with a 90% precision in a 50ms window around the correct silence boundary. We can then remove from the candidate list the utterance spans that include silence, on the assumption that a word should not include silences. Finally, in case one of the span's boundaries happens to be within a silence span, we also move it so as to not include the silence.

Hyperparameter tuning The hyperparameters p_0, λ, and μ are not learned. We simply set p_0 to zero (disallowing unaligned target words) and set μ as described above.

For λ we perform a grid search over candidate values to maximize the alignment F-score on the development set. We obtain the best scores with $\lambda = 0.5$.

5 Related Work

A first step towards modelling parallel speech can be performed by modelling phone-to-word alignment, instead of directly working on continuous speech. For example, Stahlberg et al. (2012) extend IBM Model 3 to align phones to words in order to build

cross-lingual pronunciation lexicons. Pialign (Neubig et al., 2012) aligns characters and can be applied equally well to phones. Duong et al. (2016) use an extension of the neural attentional model of Bahdanau et al. (2015) for aligning phones to words and speech to words; we discuss this model below in Section 6.2.

There exist several supervised approaches that attempt to integrate speech recognition and machine translation. However, they rely heavily on the abundance of training data, pronunciation lexicons, or language models, and therefore cannot be applied in a low- or zero-resource setting.

A task somewhat similar to ours, which operates at a monolingual level, is the task of zero-resource spoken term discovery, which aims to discover repeated words or phrases in continuous speech. Various approaches (Ten Bosch and Cranen, 2007; Park and Glass, 2008; Muscariello et al., 2009; Zhang and Glass, 2010; Jansen et al., 2010) have been tried, in order to spot keywords, using segmental DTW to identify repeated trajectories in the speech signal.

Kamper et al. (2016) try to discover word segmentation and a pronunciation lexicon in a zero-resource setting, combining DTW with acoustic embeddings; their methods operate in a very low-vocabulary setting. Bansal (2015) attempts to build a speech translation system in a low-resource setting, by using as source input the simulated output of an unsupervised term discovery system.

6 Experiments

We evaluate our method on two language pairs, Spanish-English and Griko-Italian, against two baseline methods, a naive baseline, and the model of Duong et al. (2016).

6.1 Data

For each language pair, we require a sentence-aligned parallel corpus of source-language speech and target-language text. A subset of these sentences should be annotated with span-to-word alignments for use as a gold standard.

6.1.1 Spanish-English

For Spanish-English, we use the Spanish CALLHOME corpus (LDC96S35) and the Fisher corpus (LDC2010T04), which consist of telephone conversations between Spanish native speakers based in the US and their relatives abroad, together with English translations produced by Post et al. (2013). Spanish is obviously not a low-resource language, but we pretend that it is low-resource by not making use of any Spanish ASR or resources like transcribed speech or pronunciation lexicons.

Since there do not exist gold standard alignments between the Spanish speech and English words, we use the "silver" standard alignments produced by Duong et al. (2016) for the CALLHOME corpus, and followed the same procedure for the Fisher corpus as well. In order to obtain them, they first used a forced aligner to align the speech to its transcription, and GIZA++ with the `gdfa` symmetrization heuristic to align the Spanish transcription to the English translation. They then combined the two alignments to produce "silver" standard alignments between the Spanish speech and the English words.

The CALLHOME dataset consists of 17532 Spanish utterances, based on the dialogue turns. We first use a sample of 2000 sentences, out of which we use 200 as a development set and the rest as a test set. We also run our experiments on the whole dataset, selecting 500 utterances for a development set, using the rest as a test set. The Fisher dataset consists of 143355 Spanish utterances. We use 1000 of them as a development set and the rest as a test set.

6.1.2 Griko-Italian

We also run our model on a corpus that consists of about 20 minutes of speech in Griko, an endangered minority dialect of Greek spoken in south Italy, along with text translations into Italian (Lekakou et al., 2013).[2] The corpus consists of 330 mostly prompted utterances by nine native speakers. Although the corpus is very small, we use it to showcase the effectiveness of our method in a hard setting with extremely low resources.

All utterances were manually annotated and transcribed by a trained linguist and bilingual speaker of both languages, who produced the Griko transcriptions and Italian glosses. We created full translations into Italian and manually aligned the translations with the Griko transcriptions. We then com-

[2] http://griko.project.uoi.gr

bined the two alignments (speech-to-transcription and transcription-to-translation) to produce speech-to-translation alignments. Therefore, our comparison is done against an accurate "gold" standard alignment. We split the data into a development set of just 30 instances, and a test set of the remaining 300 instances.

6.1.3 Preprocessing

In both data settings, we treat the speech data as a sequence of 39-dimensional Perceptual Linear Prediction (PLP) vectors encoding the power spectrum of the speech signal (Hermansky, 1990), computed at 10ms intervals. We also normalize the features at the utterance level, shifting and scaling them to have zero mean and unit variance.

6.2 Baselines

Our naive baseline assumes that there is no reordering between the source and target language, and aligns each target word e_i to a source span whose length in frames is proportional to the length of e_i in characters. This actually performs very well on language pairs that show minimal or no reordering, and language pairs that have shared or related vocabularies.

The other baseline that we compare against is the neural network attentional model of Duong et al. (2016), which extends the attentional model of Bahdanau et al. (2015) to be used for aligning and translating speech, and, along with several modifications, achieve good results on the phone-to-word alignment task, and almost match the baseline performance on the speech-to-word alignment task.

7 Results

To evaluate an automatic alignment between the speech and its translation against the gold/silver standard alignment, we compute alignment precision, recall, and F-score as usual, but on links between source-language frames and target-language words.

7.1 Overview

Table 1 shows the precision, recall, and balanced F-score of the three models on the Spanish-English CALLHOME corpus (both the 2000-sentence subset and the full set), the Spanish-English Fisher corpus, and the Griko-Italian corpus.

		method	precision	recall	F-score
CALLHOME spa-eng	2k sents	ours	**38.8**	38.9	**38.8**
		naive	31.9	**40.8**	35.8
		neural	23.8	29.8	26.4
	17k sents	ours	**38.4**	38.8	**38.6**
		naive	31.8	**40.7**	35.7
		neural	26.1	32.9	29.1
Fisher spa-eng	143k sents	ours	**33.3**	28.7	**30.8**
		naive	24.0	**33.2**	27.8
gri-ita	300 sents	ours	**56.6**	51.2	**53.8**
		naive	42.2	**52.2**	46.7
		neural	24.6	30.0	27.0

Table 1: Our model achieves higher precision and F-score than both the naive baseline and the neural model on all datasets.

In all cases, our model outperforms both the naive baseline and the neural attentional model. Our model, when compared to the baselines, improves greatly on precision, while slightly underperforming the naive baseline on recall. In certain applications, higher precision may be desirable: for example, in language documentation, it's probably better to err on the side of precision; in phrase-based translation, higher-precision alignments lead to more extracted phrases.

The rest of the section provides a further analysis of the results, focusing on the extremely low-resource Griko-Italian dataset.

7.2 Speaker robustness

Figure 2 shows the alignments produced by our model for three utterances of the same sentence from the Griko-Italian dataset by three different speakers. Our model's performance is roughly consistent across these utterances. In general, the model does not seem significantly affected by speaker-specific variations, as shown in Table 2.

We do find, however, that the performance on male speakers is slightly higher compared to the female speakers. This might be because the female speakers' utterances are, on average, longer by about 2 words than the ones uttered by males.

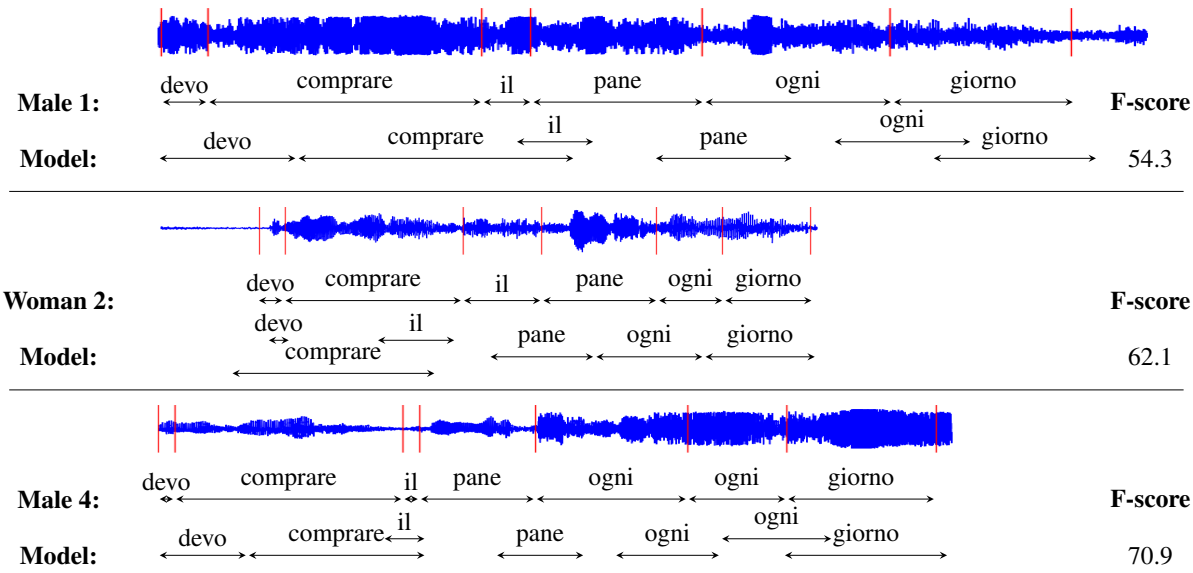


Figure 2: Alignments produced for the Italian sentence `devo comprare il pane ogni giorno` as uttered by three different Griko speakers.

speaker	utt	len	F-score
female 1	55	9.0	49.4
female 2	61	8.1	55.0
female 3	41	9.6	51.0
female 4	23	7.3	54.4
female 5	21	6.1	56.6
male 1	35	5.9	59.5
male 2	32	6.0	61.9
male 3	34	6.7	60.2
male 4	23	6.4	64.0

Table 2: Model performance (F-score) is generally consistent across speakers. The second column (utt) shows the number of utterances per speaker; the third (len), their average length in words.

7.3 Word level analysis

We also compute F-scores for each Italian word type. As shown in Figure 3, the longer the word's utterance, the easier it is for our model to correctly align it. Longer utterances seem to carry enough information for our DTW-based measure to function properly. On the other hand, shorter utterances are harder to align. The vast majority of Griko utterances that have less than 20 frames and are less accurately aligned correspond to monosyllabic determiners (`o`, `i`,`a`, `to`, `ta`) or conjunctions and prepositions (`ka`, `ce`, `en`, `na`, `an`). For such short utterances, there could be several parts of the signal that possibly match the prototype, leading the clustering component to prefer to align to wrong spans.

Furthermore, we note that rare word types tend to be correctly aligned. The average F-score for hapax legomena (on the Italian side) is 63.2, with 53% of them being aligned with an F-score higher than 70.0.

7.4 Comparison with proper model

As mentioned in Section 3, our model is deficient, but it performs much better than the model that sums to one (henceforth, the "proper" model): In the Spanish-English dataset (2000 sentences sample) the proper model yields an F-score of 32.1, performing worse than the naive baseline; in the Griko-

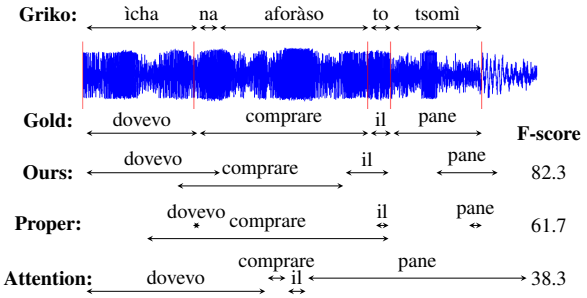


Figure 4: The deficient model performs very well, whereas the proper and the attentional model prefer extreme alignment spans. For example, the proper model's alignment for the words `dovevo` and `pane` are much too short.

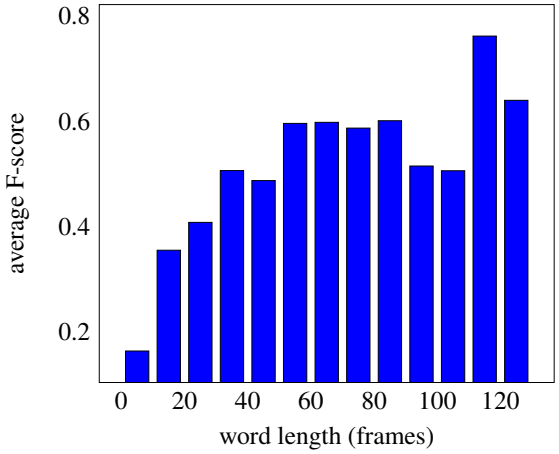


Figure 3: There is a positive correlation between average word-level F-score and average word utterance length (in frames).

Italian dataset, it achieves an F-score of 44.3, which is better than the baselines, but still worse than our model.

In order to further examine why this happens, we performed three EM iterations on the Griko-Italian dataset with our model (in our experience, three iterations are usually enough for convergence), and then computed one more E step with both our model and the proper model, so as to ensure that the two models would align the dataset using the exact same prototypes and that their outputs will be comparable.

In this case, the proper model achieved an overall F-score of 44.0, whereas our model achieved an F-score of 53.6. Figures 4 and 5 show the resulting alignments for two sentences. In both of these examples, it is clear that the proper model prefers extreme spans: the selected spans are either much too short or

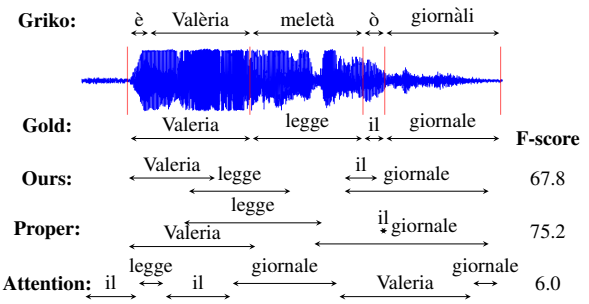


Figure 5: One of the rare examples where the proper model performs better than the deficient one. The hapax legomena `Valeria` and `giornali` are not properly handled by the attentional model.

(less frequently) much too long. This is further verified by examining the statistics of the alignments: the average span selected by the proper model has a length of about 30 ± 39 frames whereas the average span of the alignments produced by our deficient model is 37 ± 24 frames. This means that the alignments of the deficient model are much closer to the gold ones, whose average span is 42 ± 26 frames.

We think that this is analogous to the "garbage collection" problem in word alignment. In the IBM word alignment models, if a source word f occurs in only one sentence, then EM can align many target words to f and learn a very peaked distribution $t(e \mid f)$. This can happen in our model and the proper model as well, of course, since IBM Model 2 is embedded in them. But in the proper model, something similar can also happen with $s(f \mid a, b)$: EM can make the span (a, b) large or small, and evidently making the span small allows it to learn a very peaked distribution $s(f \mid a, b)$. By contrast, our model has $s(a, b \mid f)$, which seems less susceptible to this kind of effect.

8 Conclusion

Alignment of speech to text translations is a relatively new task, one with particular relevance for low-resource or endangered languages. The model we propose here, which combines `fast_align` and k-means clustering using DTW and DBA, outperforms both a very strong naive baseline and a neural attentional model, on three tasks of various sizes.

The language pairs used here do not have very much word reordering, and more divergent language

pairs should prove more challenging. In that case, the naive baseline should be much less competitive. Similarly, the `fast_align`-based distortion model may become less appopriate; we plan to try incorporating IBM Model 3 or the HMM alignment model (Vogel et al., 1996) instead. Finally, we will investigate downstream applications of our alignment methods, in the areas of both language documentation and speech translation.

Acknowledgements

We would like to thank Steven Bird, Eamonn Keogh, and the anonymous reviewers for their helpful feedback. This research was supported in part by NSF Award 1464553.

References

Gilles Adda, Sebastian Stüker, Martine Adda-Decker, Odette Ambouroue, Laurent Besacier, David Blachon, Hélène Bonneau-Maynard, Pierre Godard, Fatima Hamlaoui, Dmitry Idiatov, et al. 2016. Breaking the unwritten language barrier: The BULB project. *Procedia Computer Science*, 81:8–14.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Sameer Bansal. 2015. Speech translation without speech recognition. Master's thesis, University of Edinburgh.

Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proc. KDD*, pages 359–370.

Steven Bird, Lauren Gawne, Katie Gelbart, and Isaac McAlister. 2014. Collecting bilingual audio in remote indigenous communities. In *Proc. COLING*.

David Blachon, Elodie Gauthier, Laurent Besacier, Guy-Noël Kouarata, Martine Adda-Decker, and Annie Rialland. 2016. Parallel speech collection for underresourced language studies using the Lig-Aikuma mobile device app. *Procedia Computer Science*, 81:61–66.

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

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proc. NAACL HLT*, pages 949–959, June.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proc. NAACL HLT*.

Hynek Hermansky. 1990. Perceptual linear predictive (PLP) analysis of speech. *J. Acoustical Society of America*, 87(4):1738–1752.

Aren Jansen, Kenneth Church, and Hynek Hermansky. 2010. Towards spoken term discovery at scale with zero resources. In *Proc. INTERSPEECH*, pages 1676–1679.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2016. Unsupervised word segmentation and lexicon discovery using acoustic word embeddings. *IEEE Trans. Audio, Speech, and Language Processing*.

Vahid Khanagha, Khalid Daoudi, Oriol Pont, and Hussein Yahia. 2014. Phonetic segmentation of speech signal using local singularity analysis. *Digital Signal Processing*.

Marika Lekakou, Valeria Baldiserra, and Antonis Anastasopoulos. 2013. Documentation and analysis of an endangered language: aspects of the grammar of Griko.

Armando Muscariello, Guillaume Gravier, and Frédéric Bimbot. 2009. Audio keyword extraction by unsupervised word discovery. In *Proc. INTERSPEECH*.

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. 2012. Machine translation without words through substring alignment. In *Proc. ACL*.

Alex S. Park and James R. Glass. 2008. Unsupervised pattern discovery in speech. *IEEE Trans. Audio, Speech, and Language Processing*, 16(1):186–197.

François Petitjean, Alain Ketterlin, and Pierre Gançarski. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus. In *Proc. IWSLT*.

Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Felix Stahlberg, Tim Schlippe, Sue Vogel, and Tanja Schultz. 2012. Word segmentation through cross-lingual word-to-phoneme alignment. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*.

Louis Ten Bosch and Bert Cranen. 2007. A computational model for unsupervised word discovery. In *Proc. INTERSPEECH*, pages 1481–1484.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841.

Yaodong Zhang and James R Glass. 2010. Towards multi-speaker unsupervised speech pattern discovery. In *Proc. ICASSP*, pages 4366–4369.

HUME: Human UCCA-Based Evaluation of Machine Translation

Alexandra Birch[1]*, Omri Abend[2*], Ondřej Bojar[3*], Barry Haddow[1*]
[1]School of Informatics, University of Edinburgh
[2]School of Computer Science and Engineering, Hebrew University of Jerusalem
[3] Charles University in Prague, Faculty of Mathematics and Physics
a.birch@ed.ac.uk, oabend@cs.huji.ac.il
bojar@ufal.mff.cuni.cz, bhaddow@inf.ed.ac.uk

Abstract

Human evaluation of machine translation normally uses sentence-level measures such as relative ranking or adequacy scales. However, these provide no insight into possible errors, and do not scale well with sentence length. We argue for a semantics-based evaluation, which captures what meaning components are retained in the MT output, thus providing a more fine-grained analysis of translation quality, and enabling the construction and tuning of semantics-based MT. We present a novel human semantic evaluation measure, Human UCCA-based MT Evaluation (HUME), building on the UCCA semantic representation scheme. HUME covers a wider range of semantic phenomena than previous methods and does not rely on semantic annotation of the potentially garbled MT output. We experiment with four language pairs, demonstrating HUME's broad applicability, and report good inter-annotator agreement rates and correlation with human adequacy scores.

1 Introduction

Human judgement should be the ultimate test of the quality of an MT system. Nevertheless, common measures for human MT evaluation, such as adequacy and fluency judgements or the relative ranking of possible translations, are problematic in two ways. First, as the quality of translation is multi-faceted, it is difficult to quantify the quality of the entire sentence in a single number. This is indeed reflected in the diminishing inter-annotator agreement (IAA) rates of human ranking measures with the sentence length (Bojar et al., 2011). Second, a sentence-level quality score does not indicate what parts of the sentence are badly translated, and so cannot inform developers in repairing these errors.

These problems are partially addressed by measures that decompose over parts of the evaluated translation, often words or n-grams (see §2 for a brief survey of previous work). A promising line of research decomposes metrics over semantically defined units, quantifying the similarity of the output and the reference in terms of their verb argument structure; the most notable of these measures is HMEANT (Lo and Wu, 2011).

We propose the HUME metric, a human evaluation measure that decomposes over UCCA semantic units. UCCA (Abend and Rappoport, 2013) is an appealing candidate for semantic analysis, due to its cross-linguistic applicability, support for rapid annotation, and coverage of many fundamental semantic phenomena, such as verbal, nominal and adjectival argument structures and their inter-relations.

HUME operates by aggregating human assessments of the translation quality of individual semantic units in the source sentence. We are thus avoiding the semantic annotation of machine-generated text, which is often garbled or semantically unclear. This also allows the re-use of the source semantic annotation for measuring the quality of different translations of the same source sentence and avoids relying on reference translations, which have been shown to bias annotators (Fomicheva and Specia, 2016).

After a brief review (§2), we describe HUME in

* All authors contributed equally to this work.

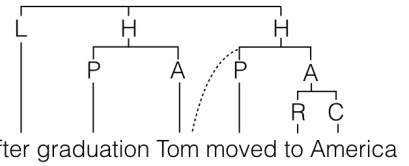
Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1264–1274,
Austin, Texas, November 1-5, 2016.

detail (§3). Our experiments with four language pairs: English to Czech, German, Polish and Romanian (§4) document HUME's inter-annotator agreement and efficiency (time of annotation). We further empirically compare HUME with direct assessment of human adequacy ratings (§5), and conclude by discussing the differences with HMEANT (§6).

2 Background

MT Evaluation. Human evaluation is generally done by ranking the outputs of multiple systems e.g., in the WMT tasks (Bojar et al., 2015), or by assigning adequacy/fluency scores to each translation, a procedure recently improved by Graham et al. (2015b) under the title Direct Assessment. We use this latter method to compare and contrast with HUME later in the paper. HTER (Snover et al., 2006) is another widely used human evaluation metric which uses edit distance metrics to compare a translation and its human post-edition. HTER suffers from the problem that small edits in the translation could in fact be serious flaws in accuracy, e.g., deleting a negation. Some manual measures ask annotators to explicitly mark errors, but this has been found to have even lower agreement than ranking (Lommel et al., 2014).

However, while providing the gold standard for MT evaluation, human evaluation is not a scalable solution. Scalability is addressed by employing automatic and semi-automatic approximations of human judgements. Commonly, such scores decompose over the sub-parts of the translation, and quantify how many of these sub-parts appear in a manually created reference translation. This decomposition allows system developers to localize the errors. The most commonly used measures decompose over n-grams or individual words, e.g., BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and METEOR (Banerjee and Lavie, 2005). Another common approach is to determine the similarity between the reference and translation in terms of string edits (Snover et al., 2006). While these measures stimulated much progress in MT research by allowing the evaluation of massive-scale experiments, the focus on words and n-grams does not provide a good estimate of semantic correctness, and may favour shallow string-based MT models.



L	Linker	A	Participant
H	Parallel Scene	R	Relater
P	Process	C	Centre

Figure 1: Sample UCCA annotation. Leaves correspond to words and nodes to units. The dashed edge indicates that "Tom" is also a participant in the "moved to America" Scene. Edge labels mark UCCA categories.

In order to address this shortcoming, more recent work quantified the similarity of the reference and translation in terms of their structure. Liu and Gildea (2005) took a syntactic approach, using dependency grammar, and Owczarzak et al. (2007) took a similar approach using Lexical Functional Grammar structures. Giménez and Màrquez (2007) proposed to combine multiple types of information, capturing the overlap between the translation and reference in terms of their semantic (predicate-argument structures), lexical and morphosyntactic features. Macháček and Bojar (2015) divided the source sentences into shorter segments, defined using a phrase structure parse, and applied human ranking to the resulting segments.

Perhaps the most notable attempt at semantic MT evaluation is MEANT and its human variant HMEANT (Lo and Wu, 2011), which quantifies the similarity between the reference and translation in terms of the overlap in their verbal argument structures and associated semantic roles. We discuss the differences between HMEANT and HUME in §6.

Semantic Representation. UCCA (Universal Conceptual Cognitive Annotation) (Abend and Rappoport, 2013) is a cross-linguistically applicable scheme for semantic annotation. Formally, an UCCA structure is a directed acyclic graph (DAG), whose leaves correspond to the words of the text. The graph's nodes, called *units*, are either terminals or several elements jointly viewed as a single entity according to some semantic or cognitive consideration. Edges bear a category, indicating the role of the sub-unit in the structure the unit represents.

UCCA's basic inventory of distinctions (its *foundational layer*) focuses on argument structures (ad-

jectival, nominal, verbal and others) and relations between them. The most basic notion is the *Scene*, which describes a movement, an action or a state which persists in time. Each Scene contains one main relation and zero or more participants. For example, the sentence "After graduation, Tom moved to America" contains two Scenes, whose main relations are "graduation" and "moved". The participant "Tom" is a part of both Scenes, while "America" only of the latter (Figure 1). Further categories account for inter-scene relations and the sub-structures of participants and relations.

The use of UCCA for semantic MT evaluation has several motivations. First, UCCA's foundational layer can be annotated by non-experts after a short training (Abend and Rappoport, 2013; Marinotti, 2014). Second, UCCA is cross-linguistically applicable, seeking to represent what is shared between languages by building on linguistic typological theory (Dixon, 2010b; Dixon, 2010a; Dixon, 2012). Its cross-linguistic applicability has so far been tested in annotations of English, French, German and Czech. Third, the scheme has been shown to be stable across translations: UCCA annotations of translated text usually contain the same set of relations (Sulem et al., 2015), indicating that UCCA reflects a layer of representation that in a correct translation is mostly shared between the translation and the source.

The Abstract Meaning Representation (AMR) (Banarescu et al., 2013) shares UCCA's motivation for defining a more complete semantic annotation. However, using AMR is not optimal for defining a decomposition of a sentence into semantic units as it does not anchor its semantic symbols in the text, and thus does not provide clear decomposition of the sentence into sub-spans. Also, AMR is more fine-grained than UCCA and consequently harder to annotate. Other approaches represent semantic structures as bi-lexical dependencies (Sgall et al., 1986; Hajič et al., 2012; Oepen and Lønning, 2006), which are indeed anchored in the text, but are less suitable for MT evaluation as they require linguistic expertise for their annotation.

3 The HUME Measure

3.1 Annotation Procedure

This section summarises the manual annotation procedure used to compute the HUME measure. We denote the source sentence as s and the translation as t. The procedure involves two manual steps: (1) UCCA-annotating s, (2) HUME-annotation: human judgements as to the translation quality of each semantic unit of s relative to t, where units are defined according to the UCCA annotation. UCCA annotation is performed once for every source sentence, irrespective of the number of its translations we wish to evaluate, and requires proficiency in the source language only. HUME annotation requires the employment of bilingual annotators.[1]

UCCA Annotation. We begin by creating UCCA annotations for the source sentence, following the UCCA guidelines.[2] A UCCA annotation for a sentence s is a labeled DAG G, whose leaves are the words of s. For every node in G, we define its *yield* to be its leaf descendants. The semantic units for s according to G are the yields of nodes in G.

Translation Evaluation. HUME annotation is done by traversing the semantic units of the source sentence, which correspond to the arguments and relations expressed in the text, and marking the extent to which they have been correctly translated. HUME aggregates the judgements of the users into a composite score, which reflects the overall extent to which the semantic content of s is preserved in t.

Annotation of the semantic units requires first deciding whether a unit is *structural*, i.e., has meaning-bearing sub-units in the target language, or *atomic*. In most cases, atomic units correspond to individual words, but they may also correspond to multi-word expressions that translate as one unit. For instance, the expression "took a shower" is translated into the German "duschte", while its individual words do not correspond to any sub-part of the German translation, motivating the labeling the entire expression as an atomic node. When a multi-word unit is labeled

[1] Where bilingual annotators are not available, the evaluation could be based on the UCCA structure for the *reference* translation. See discussion in §6.

[2] All UCCA-related resources can be found here: `http://www.cs.huji.ac.il/~oabend/ucca.html`

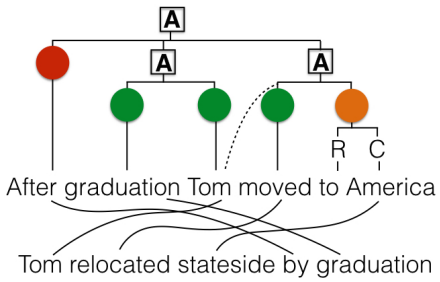


Figure 2: HUME annotation of an UCCA tree with a word-aligned example translation shown below. Atomic units are labelled using traffic lights (Red, Orange, Green) and structural units are marked A or B.

as atomic, its sub-units' annotations are ignored in the evaluation.

Atomic units can be labelled as "Green" (G, correct), "Orange" (O, partially correct) and "Red" (R, incorrect). Green means that the meaning of the word or phrase has been largely preserved. Orange means that the essential meaning of the unit has been preserved, but some part of the translation is wrong. This is often be due to the translated word having the wrong inflection, in a way that impacts little on the understandability of the sentence. Red means that the essential meaning of the unit has not been captured.

Structural units have sub-units (children in the UCCA graph), which are themselves atomic or structural. Structural units are labeled as "Adequate" (A) or "Bad" (B), meaning that the relation between the sub-units went wrong[3]. We will use the example "man bites dog" to illustrate typical examples of why a structural node should be labelled as "Bad": incorrect ordering ("dog bites man"), deletion ("man bites") and insertion ("man bites biscuit dog").

HUME labels reflect adequacy, rather than fluency judgements. Specifically, annotators are instructed to label a unit as Adequate if its translation is understandable and preserves the meaning of the source unit, even if its fluency is impaired.

Figure 2 presents an example of a HUME annotation, where the translation is in English for ease of comprehension. When evaluating "to America" the annotator looks at the translation and sees the word "stateside". This word captures the whole phrase and so we mark this non-leaf node with an atomic label. Here we choose Orange since it approximately captures the meaning in this context. The ability to mark non-leaves with atomic labels allows the annotator to account for translations which only correspond at the phrase level. Another feature highlighted in this example is that by separating structural and atomic units, we are able to define where an error occurs, and localise the error to its point of origin. The linker "After" is translated incorrectly as "by" which changes the meaning of the entire sentence. This error is captured at the atomic level, and it is labelled Red. The sentence still contains two Scenes and a Linker and therefore we mark the root node as structurally correct, Adequate.

3.2 Composite Score

We proceed to detailing how judgements on the semantic units of the source are aggregated into a composite score. We start by taking a very simple approach and compute an accuracy score. Let $\mathit{Green}(s,t)$, $\mathit{Adequate}(s,t)$ and $\mathit{Orange}(s,t)$ be the number of Green, Adequate and Orange units, respectively. Let $\mathit{Units}(s)$ be the number of units marked with any of the labels. Then HUME's composite score is:

$$\mathit{HUME}(s,t) = \frac{\mathit{Green}(s,t) + \mathit{Adequate}(s,t) + 0.5 \cdot \mathit{Orange}(s,t)}{\mathit{Units}(s)}$$

3.3 Annotation Interface

Figure 3 shows the HUME annotation interface[4]. One source sentence and one translation are presented at a time. The user is asked to select a label for each source semantic unit, by clicking the "A", "B", Green, Orange, or Red buttons to the right of the unit's box. Units with multiple parents (as with "Tom" in Figure 2) are displayed twice, once under each of their parents, but are only annotatable in one of their instances, to avoid double counting.

The interface presents, for each unit, the translation segment aligned with it. This allows the user, especially in long sentences, to focus her attention on the parts that are most likely to be relevant for her judgement. As the alignments are automatically derived, and therefore noisy, the annotator is instructed to treat the aligned text is a cue, but to ignore the alignment if it is misleading, and instead make a

[3]Three labels are used with atomic units, as opposed to two labels with structural units, as atomic units are more susceptible to slight errors.

[4]A demo of HUME can be found in `www.cs.huji.ac.il/~oabend/hume_demo.html`

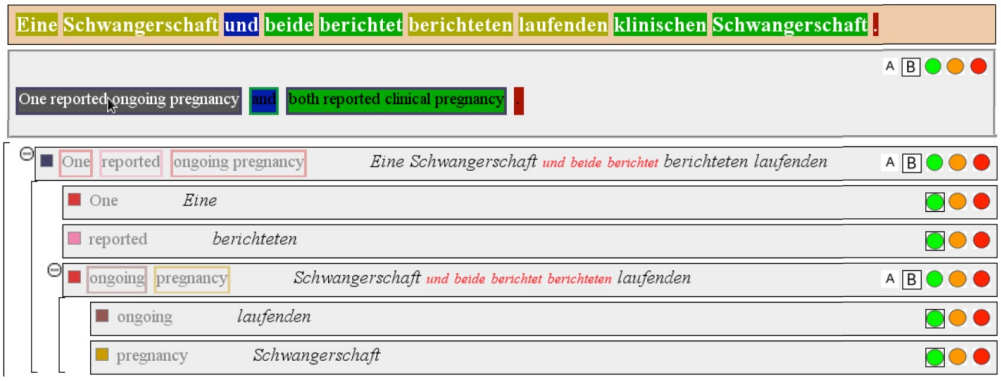


Figure 3: The HUME annotation tool. The top orange box contains the translation. The source sentence is directly below it, followed by the tree of the source semantic units. Alignments between the source and translation are in italics and unaligned intervening words are in red (see text).

judgement according to the full translation. Concretely, let s be a source sentence, t a translation, and $A \subset 2^s \times 2^t$ a many-to-many word alignment. If u is a semantic unit in s, whose yield is $yld(u)$, we define the aligned text in t to be $\bigcup_{(x_s,x_t)\in A \wedge x_s \cap yld(u) \neq \emptyset} x_t$.

Where the aligned text is discontinuous in t, words between the left and right boundaries which are not contained in it (intervening words) are presented in a smaller red font. Intervening words are likely to change the meaning of the translation of u, and thus should be attended to when considering whether the translation is correct or not.

For example, in Figure 3, "ongoing pregnancy" is translated to "Schwangerschaft ... laufenden" (lit. "pregnancy ... ongoing"). This alone seems acceptable but the interleaving words in red notify the annotator to check the whole translation, in which the meaning of the expression is not preserved[5]. The annotator should thus mark this structural node as Bad.

4 Experiments

In order to validate the HUME metric, we ran an annotation experiment with one source language (English), and four target languages (Czech, German, Polish and Romanian), using text from the public health domain. Semantically accurate translation is paramount in this domain, which makes it particularly suitable for semantic MT evaluation. HUME is evaluated in terms of its consistency (inter-annotator agreement), efficiency (time of annotation) and validity (by comparing it with crowd-sourced adequacy judgements).

4.1 Datasets and Translation Systems

For each of the four language pairs under consideration we built phrase-based MT systems using Moses (Koehn et al., 2007). These were trained on large parallel data sets extracted from OPUS (Tiedemann, 2009), and the data sets released for the WMT14 medical translation task (Bojar et al., 2014), giving between 45 and 85 million sentences of training data, depending on the language pair. These translation systems were used to translate texts derived from both NHS 24[6] and Cochrane[7] into the four languages. NHS 24 is a public body providing healthcare and health-service related information in Scotland; Cochrane is an international NGO which provides independent systematic reviews on health-related research. NHS 24 texts come from the "Health A-Z" section in the NHS Inform website, and Cochrane texts come from their plain language summaries and abstracts.

4.2 HUME Annotation Statistics

The source sentences are all in English, and their UCCA annotation was performed by four computational linguists and one linguist. For the annotation of the MT output, we recruited two annotators for each of German, Romanian and Polish and one main annotator for Czech. For computing Czech IAA, several further annotators worked on a small number of sentences each. We treat these further annotators

[5]The interleaving words are "... und beide berichtet berichteten ..." (lit. "... and both report reported ..."), which doesn't form any coherent relation with the rest of the sentence.

[6]http://www.nhs24.com/

[7]http://www.cochrane.org/

		cs	de	pl	ro
#Sentences	Annot. 1	324	339	351	230
	Annot. 2	205	104	340	337
#Units	Annot. 1	8794	9253	9557	6152
	Annot. 2	5553	2906	9303	9228

Table 1: HUME-annotated #sentences and #units.

	cs	de	pl	ro
Annot. 1	255	140	138	96
Annot. 2	*	162	229	207

Table 2: Median annotation times per sentence, in seconds. *: no timing information is available, as this was a collection of annotators, working in parallel.

as one annotator, resulting in two annotators for each language pair. The annotators were all native speakers of the respective target languages and fluent in English. They completed a three hour on-line training session which included a description of UCCA and the HUME task, followed by walking through a few examples.

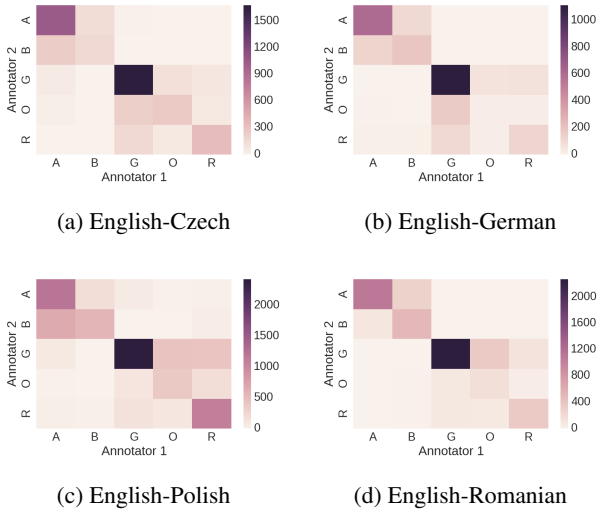
Table 1 shows the total number of sentences and units annotated by each annotator. Not all units in all sentences were annotated, often due to the annotator accidentally missing a node.

Efficiency. We estimate the annotation time using the timestamps provided by the annotation tool, which are recorded whenever an annotated sentence is submitted. Annotators are not able to re-open a sentence once submitted. To estimate the annotation time, we compute the time difference between successive sentences, and discard outlying times, assuming annotation was not continuous in these cases. From inspection of histograms of annotation times, we set the upper threshold at 500 seconds. Median annotation times are presented in Table 2, indicating that the annotation of a sentence takes around 2–4 minutes, with some variation between annotators.

Inter-Annotator Agreement. In order to assess the consistency of the annotation, we measure the Inter-Annotator Agreement (IAA) using Cohen's Kappa on the multiply-annotated units. Table 3 reports the number of units which have two annotations from different annotators and the corresponding Kappas. We report the overall Kappa, as well as separate Kappas on atomic units (annotated as Red, Orange or Green) and structural units (annotated as

	cs	de	pl	ro
Sentences	181	102	334	217
All units	4686	2793	8384	5604
Kappa	0.64	0.61	0.58	0.69
Atomic units	2982	1724	5386	3570
Kappa	0.54	0.29	0.54	0.50
Structural units	1602	1040	2655	1989
Kappa	0.31	0.44	0.33	0.58

Table 3: IAA for the multiply-annotated units, measured by Cohen's Kappa.



(a) English-Czech (b) English-German

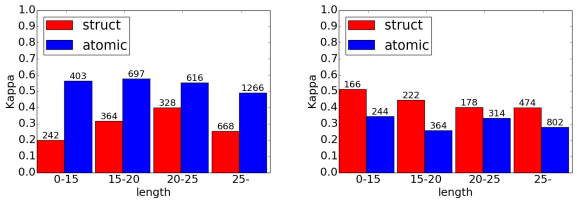
(c) English-Polish (d) English-Romanian

Figure 4: Confusion matrices for each language pair.

Adequate or Bad). As expected and confirmed by confusion matrices in Figure 4, there is generally little confusion between the two types of units. This results in the Kappa for all units being considerably higher than the Kappa over the atomic units or structural units, where there is more internal confusion.

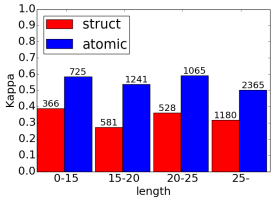
To assess HUME reliability for long sentences, we binned the sentences according to length and measured Kappa on each bin (Figure 5). We see no discernible reduction of IAA with sentence length. Table 3 also shows that the overall IAA is similar for all languages, presenting good agreement (0.6–0.7). However, there are differences observed when we break down by node type. Specifically, we see a contrast between Czech and Polish, where the IAA is higher for atomic than for structural units, and German and Romanian, where the reverse is true. We also observe low IAA (around 0.3) in the cases of German atomic units, and Polish and Czech structural units.

Looking more closely at the areas of disagree-

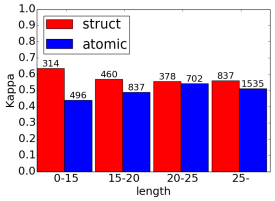


(a) English-Czech

(b) English-German



(c) English-Polish



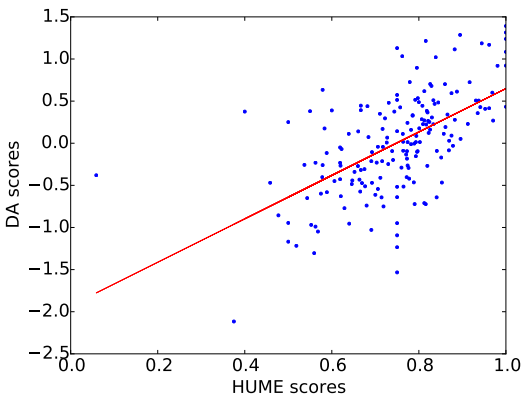
(d) English-Romanian

Figure 5: Kappa versus sentence length for structural and atomic units. (Node counts in bins on top of each bar.)

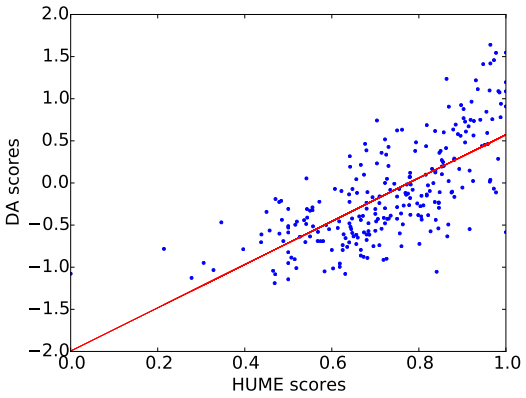
ment, we see that for the Polish structural units, the proportion of As was quite different between the two annotators (53% vs. 71%), whereas for other languages the annotators agree in the proportions. We believe that this was because one of the Polish annotators did not fully understand the guidelines for structural units, and percolated errors up the tree, creating more Bs. For German atomic and Czech structural units, where Kappa is also around 0.3, the proportion of such units being marked as "correct" is relatively high, meaning that the class distribution is more skewed, so the expected agreement used in the Kappa calculation is high, lowering Kappa. Finally we note some evidence of domain-specific disagreements, for instance the German MT system normally translated "review" (as in "systematic review" – a frequent term in the Cochrane texts) as "Überprüfung", which one annotator marked correct, and the other (a Cochrane employee) as incorrect.

5 Comparison with Direct Assessment

Recent research (Graham et al., 2015b; Graham et al., 2015a; Graham, 2015) has proposed a new approach for collecting accuracy ratings, direct assessment (DA). Statistical interpretation of a large number of crowd-sourced adequacy judgements for each candidate translation on a fine-grained scale of 0 to 100 results in reliable aggregate scores, that correlate very strongly with one another.



(a) English-German



(b) English-Romanian

Figure 6: HUME vs DA scores. DA score have been standardised for each crowdsourcing annotator and averaged across exactly 10 annotators. HUME scores are averaged where there were two annotations.

We attempted to follow Graham et al. (2015b) but struggled to get enough crowd-sourced judgements for our target languages. We ended up with 10 adequacy judgements on most of the HUME annotated translations for German and Romanian but insufficient data for Czech and Polish. We see this as a severe practical limitation of DA.

Figure 6 plots the HUME score for each sentence against its DA score. HUME and Direct Assessment scores correlate reasonably well. The Pearson correlation for en-ro (en-de) is 0.70 (0.58), or 0.78 (0.74) if only doubly HUME-annotated points are considered. This confirms that HUME is consistent with an accepted human evaluation method, despite their conceptual differences. While DA is a valuable tool, HUME has two advantages: it returns fine-grained

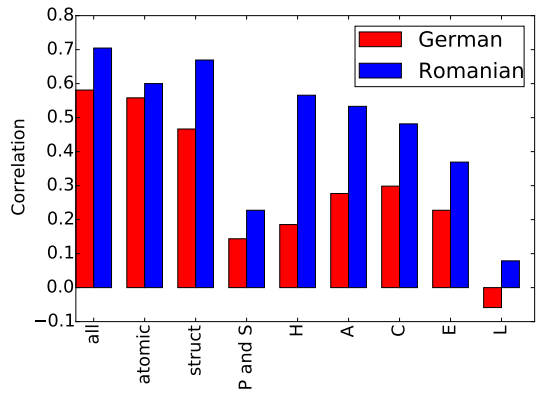


Figure 7: Pearson correlation of HUME vs. DA scores for en-ro and en-de. Each bar represents a correlation between DA and an aggregate HUME score based on a sub-set of the units (#nodes for the en-de/en-ro setting in brackets): all units ("all", 8624/10885), atomic ("atomic", 5417/6888) and structural units ("struct", 3207/3997), and units by UCCA categories: Scene main relations (i.e, Process and State units; "P and S", 954/1178), Parallel Scenes ("H", 656/784), Participants ("A", 1348/1746), Centres ("C", 1904/2474), elaborators ("E", 1608/2031) and linkers ("L", 261/315).

semantic information about the quality of translations and it only requires very few annotators. Direct assessment returns a single opaque score, and (as also noted by Graham et al.) requires a large crowd which may not be available or reliable.

Figure 7 presents an analysis of HUME's correlations with DA by HUME unit type, an analysis enabled by HUME's semantic decomposition. For both target languages, correlation is highest in the 'all' case, supporting our claim for the value of aggregating over a wide range of semantic phenomena. Some types of nodes predict the DA scores better than others. HUME scores on As correlate more strongly with DA than scores on Scene Main Relations (P+S). Center nodes (C) are also more correlated than elaborator nodes (E), which is expected given that Centers are defined to be more semantically dominant. Future work will construct an aggregate HUME score which weights the different node types according to their semantic prominence.

HUME and DA are conceptually very different metrics: while DA standardises and averages scores across annotators to denoise the crowd-sourced raw data, thus obtaining a single aggregate score, HUME decomposes over a combinatorial structure, thus allowing to localize the translation errors. We now turn to comparing HUME to a more conceptually-related measure, namely HMEANT.

6 Comparison with HMEANT

HMEANT is a human MT evaluation metric that measures the overlap between the translation a reference in terms of their SRL annotations. In this section we present a qualitative comparison between HUME and HMEANT, using examples from our experimental data.

Verbal Structures Only? HMEANT focuses on verbal argument structures, ignoring other pervasive phenomena such as non-verbal predicates and interclausal relations. Consider the following example:

Source	a coronary angioplasty may not be technically possible
Transl.	eine koronare Angioplastie kann nicht technisch möglich
Gloss	a coronary angioplasty can not technically possible

The German translation is largely correct, except that the main verb "sein" ("be") is omitted. While this may be interpreted as a minor error, HMEANT will assign the sentence a very low score, as it failed to translate the main verb.

It is also relatively common that verbal constructions are translated as non-verbal ones or vice versa. Consider the following example:

Source	... tend to be higher in saturated fats
Transl.	... in der Regel höher in gesättigte Fette
Gloss	... as a rule higher in saturated fats

The German translation is largely correct, despite the grammatical divergence, namely that the English verb "tend" is translated into the German prepositional phrase "in der Regel" ("as a rule"). HMEANT will consider the translation to be of poor quality as there is no German verb to align with the English one.

We conducted an analysis of the English UCCA Wikipedia corpus (5324 sentences) in order to assess the pervasiveness of three phenomena that are not well supported by HMEANT.[8] First, copula clauses

[8] Argument structures and linkers are explicitly marked in UCCA. Non-auxiliary instances of "be" and nouns are identi-

are treated in HMEANT simply as instances of the main verb "be", which generally does not convey the meaning of these clauses. They appear in 21.7% of the sentences, according to conservative estimates that only consider non-auxiliary instances of "be". Second, nominal argument structures, ignored by HMEANT, are in fact highly pervasive, appearing in 48.7% of the sentences. Third, linkers that express inter-relations between clauses (mainly discourse markers and conjunctions) appear in 56% of the sentences, but are again ignored by HMEANT. For instance, linkers are sometimes omitted in translation, but these omissions are not penalized by HMEANT. The following is such an example from our experimental dataset:

Source	However, this review was restricted to ...
Transl.	Diese Überprüfung bescränkte sich auf ...
Gloss	This review was restricted to ...

We note that some of these issues were already observed in previous applications of HMEANT to languages other than English. See Birch et al. (2013) for German, Bojar and Wu (2012) for Czech and Chuchunkov et al. (2014) for Russian.

One Structure or Two. HUME only annotates the source, while HMEANT relies on two independently constructed structural annotations, one for the reference and one for the translation. Not annotating the translation is appealing as it is often impossible to assign a semantic structure to a low quality translation. On the other hand, HUME may be artificially boosting the perceived understandability of the translation by allowing access to the source.

Alignment. In HMEANT, the alignment between the reference and translation structures is a key part of the manual annotation. If the alignment cannot be created, the translation is heavily penalized. Bojar and Wu (2012) and Chuchunkov et al. (2014) argue that the structures of the reference and of an accurate translation may still diverge, for instance due to a different interpretation of a PP-attachment, or the verb having an additional modifier in one of the structures. It would be desirable to allow modifications to the SRL annotations at the alignment stage, to avoid unduly penalizing such spurious divergences.

The same issue is noted by Lo and Wu (2014): the IAA on SRL dropped from 90% to 61% when the two aligned structures were from two different annotators. HUME uses automatic (word-level) alignment, which only serves as a cue for directing the attention of the annotators. The user is expected to mentally correct the alignment as needed, thus circumventing this difficulty.

Monolingual vs. Bilingual Evaluation. HUME diverges from HMEANT and from shallower measures like BLEU, in not requiring a reference. Instead, it directly compares the source and the translation. This requires the employment of bilingual annotators, but has the benefit of avoiding using a reference, which is never uniquely defined, and may thus lead to unjustly low scores where the translation is a paraphrase of the reference. If only monolingual annotators are available, the HUME evaluation could be performed with a reference sentence instead of with the source. This, however, would risk inaccurate judgements due to the naturally occurring differences between the source and its reference translations.

7 Conclusion

We have introduced HUME, a human semantic MT evaluation measure which addresses a wide range of semantic phenomena. We have shown that it can be reliably and efficiently annotated in multiple languages, and that annotation quality is robust to sentence length. Comparison to direct assessments further support HUME's validity. We believe that HUME, and a future automated version of HUME, allows for a finer-grained analysis of translation quality, and will be useful in informing the development of a more semantically aware approach to MT.

All annotation data gathered in this project, together with analysis scripts, is available online[9].

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 644402 (HimL).

fied using the NLTK standard tagger. Nominal argument structures are here Scenes whose Main Relation is headed by a noun.

[9] https://github.com/bhaddow/hume-emnlp16

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria, August. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, MI, USA. Association for Computational Linguistics.

Alexandra Birch, Barry Haddow, Ulrich Germann, Maria Nadejde, Christian Buck, and Philipp Koehn. 2013. The feasibility of HMEANT as a human MT evaluation metric. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 52–61, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar and Dekai Wu. 2012. Towards a Predicate-Argument Evaluation for MT. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 30–38, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar F. Zaidan. 2011. A grain of salt for the WMT manual evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11, Edinburgh, Scotland.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Alexander Chuchunkov, Alexander Tarelkin, and Irina Galinskaya. 2014. Applying HMEANT to English-Russian Translations. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 43–50, Doha, Qatar, October. Association for Computational Linguistics.

Robert M.W. Dixon. 2010a. *Basic Linguistic Theory: Grammatical Topics*, volume 2. Oxford University Press.

Robert M.W. Dixon. 2010b. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

Robert M.W. Dixon. 2012. *Basic Linguistic Theory: Further Grammatical Topics*, volume 3. Oxford University Press.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145, San Diego, CA, USA. Morgan Kaufmann Publishers Inc.

Marina Fomicheva and Lucia Specia. 2016. Reference bias in monolingual machine translation evaluation. In *54th Annual Meeting of the Association for Computational Linguistics*, ACL, Berlin, Germany.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic features for automatic evaluation of heterogenous mt systems. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 256–264.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2015a. Can machine translation systems be evaluated by the crowd alone? *Natural Language Engineering*, pages 1–28.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015b. Accurate evaluation of segment-level machine translation metrics. In *Proc. of NAACL-HLT*, pages 1183–1191.

Yvette Graham. 2015. Improving evaluation of machine translation quality estimation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1804–1813, Beijing, China, July. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In

Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12), pages 3153–3160, Istanbul, Turkey, May. ELRA, European Language Resources Association.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Chi-kiu Lo and Dekai Wu. 2011. Structured vs. flat semantic role representations for machine translation evaluation. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 10–20. Association for Computational Linguistics.

Chi-Kiu Lo and Dekai Wu. 2014. On the Reliability and Inter-Annotator Agreement of Human Semantic MT Evaluation via HMEANT. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Arle Richard Lommel, Maja Popovic, and Aljoscha Burchardt. 2014. Assessing Inter-Annotator Agreement for Translation Error Annotation. In *MTE: Workshop on Automatic and Manual Metrics for Operational Translation Evaluation*. LREC.

Matouš Macháček and Ondřej Bojar. 2015. Evaluating Machine Translation Quality Using Short Segments Annotations. *The Prague Bulletin of Mathematical Linguistics*, 103:85–110, April.

Pedro Marinotti. 2014. Measuring semantic preservation in machine translation with HCOMET: human cognitive metric for evaluating translation. Master's thesis, University of Edinburgh.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of LREC*, pages 1250–1255.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007. Evaluating machine translation with LFG dependencies. *Machine Translation*, 21(2):95–119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA. Association for Computational Linguistics.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *ACL 2015 Workshop on Semantics-Driven Statistical Machine Translation (S2MT)*, pages 11–22.

Jörg Tiedemann. 2009. News from OPUS – a collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248, Borovets, Bulgaria. John Benjamins.

Improving Multilingual Named Entity Recognition with Wikipedia Entity Type Mapping

Jian Ni and **Radu Florian**
IBM T. J. Watson Research Center
1101 Kitchawan Road, Yorktown Heights, NY 10598, USA
{nij, raduf}@us.ibm.com

Abstract

The state-of-the-art named entity recognition (NER) systems are statistical machine learning models that have strong generalization capability (i.e., can recognize unseen entities that do not appear in training data) based on lexical and contextual information. However, such a model could still make mistakes if its features favor a wrong entity type. In this paper, we utilize Wikipedia as an open knowledge base to improve multilingual NER systems. Central to our approach is the construction of high-accuracy, high-coverage multilingual Wikipedia entity type mappings. These mappings are built from weakly annotated data and can be extended to new languages with no human annotation or language-dependent knowledge involved. Based on these mappings, we develop several approaches to improve an NER system. We evaluate the performance of the approaches via experiments on NER systems trained for 6 languages. Experimental results show that the proposed approaches are effective in improving the accuracy of such systems on unseen entities, especially when a system is applied to a new domain or it is trained with little training data (up to 18.3 F_1 score improvement).

1 Introduction

Named entity recognition (NER) is an important NLP task that automatically detects entities in text and classifies them into pre-defined entity types such as persons, organizations, geopolitical entities, locations, events, etc. NER is a fundamental component of many information extraction and knowledge discovery applications, including relation extraction, entity linking, question answering and data mining.

The state-of-the-art NER systems are usually statistical machine learning models that are trained with human-annotated data. Popular models include maximum entropy Markov models (MEMM) (McCallum et al., 2000), conditional random fields (CRF) (Lafferty et al., 2001) and neural networks (Collobert et al., 2011; Lample et al., 2016). Such models have strong generalization capability to recognize *unseen entities*[1] based on lexical and contextual information (features). However, a model could still make mistakes if its features favor a wrong entity type, which happens more frequently for unseen entities as we have observed in our experiments.

Wikipedia is an open-access, free-content Internet encyclopedia, which has become the de facto on-line source for general reference. A Wikipedia page about an entity normally includes both structured information and unstructured text information, and such information can be used to help determine the entity type of the referred entity.

So far there are two classes of approaches that exploit Wikipedia to improve NER. The first class of approaches use Wikipedia to generate features for NER systems, e.g., (Kazama and Torisawa, 2007; Ratinov and Roth, 2009; Radford et al., 2015). Kazama and Torisawa (2007) try to find the Wikipedia entity for each candidate word sequence and then extract a category label from the first sentence of the Wikipedia entity page. A part-of-speech (POS) tagger is used to extract the category label

[1] An entity is an *unseen entity* if it does not appear in the training data used to train the NER model.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1275–1284,
Austin, Texas, November 1-5, 2016.

features in the training and decoding phase. Ratinov and Roth (2009) aggregate several Wikipedia categories into higher-level concept and build a gazetteer on top of it. The two approaches were shown to be able to improve an English NER system. Both approaches, however, are language-dependent because (Kazama and Torisawa, 2007) requires a POS tagger and (Ratinov and Roth, 2009) requires manual category aggregation by inspection of the annotation guidelines and the training set. Radford et al. (2015) assume that document-specific knowledge base (e.g., Wikipedia) tags for each document are provided, and they use those tags to build gazetteer type features for improving an English NER system.

The second class of approaches use Wikipedia to generate weakly annotated data for training multilingual NER systems, e.g., (Richman and Schone, 2008; Nothman et al., 2013). The motivation is that annotating multilingual NER data by human is both expensive and time-consuming. Richman and Schone (2008) utilize the category information of Wikipedia to determine the entity type of an entity based on manually constructed rules (e.g., category phrase "Living People" is mapped to entity type PERSON). Such a rule-based entity type mapping is limited both in accuracy and coverage, e.g., (Toral and Muoz, 2006). Nothman et al. (2013) train a Wikipedia entity type classifier using human-annotated Wikipedia pages. Such a supervised-learning based approach has better accuracy and coverage, e.g., (Dakka and Cucerzan, 2008). A number of heuristic rules are developed in both works to label the Wikipedia text to create weakly annotated NER training data. The NER systems trained with the weakly annotated data may achieve similar accuracy compared with systems trained with little human-annotated data (e.g., up to 40K tokens as in (Richman and Schone, 2008)), but they are still significantly worse than well-trained systems (e.g., a drop of 23.9 F_1 score on the CoNLL data and a drop of 19.6 F_1 score on the BBN data as in (Nothman et al., 2013)).

In this paper, we propose a new class of approaches that utilize Wikipedia to improve multilingual NER systems. Central to our approaches is the construction of high-accuracy, high-coverage multilingual Wikipedia entity type mappings. We use weakly annotated data to train an English Wikipedia entity type classifier, as opposed to using human-annotated data as in (Dakka and Cucerzan, 2008; Nothman et al., 2013). The accuracy of the classifier is further improved via self-training. We apply the classifier on all the English Wikipedia pages and construct an English Wikipedia entity type mapping that includes entities with high classification confidence scores. To build multilingual Wikipedia entity type mappings, we generate weakly annotated classifier training data for another language via projection using the inter-language links of Wikipedia. This approach requires no human annotation or language-dependent knowledge, and thus can be easily applied to new languages.

Our goal is to utilize the Wikipedia entity type mappings to improve NER systems. A natural approach is to use a mapping to create dictionary type features for training an NER system. In addition, we develop several other approaches. The first approach applies an entity type mapping as a *decoding constraint* for an NER system. The second approach uses a mapping to *post-process* the output of an NER system. We also design a robust *joint* approach that combines the decoding constraint approach and the post-processing approach in a smart way. We evaluate the performance of the Wikipedia-based approaches on NER systems trained for 6 languages. We find that when a system is well trained (e.g., with 200K to 300K tokens of human-annotated data), the dictionary feature approach achieves the best improvement over the baseline system; while when a system is trained with little human-annotated training data (e.g., 20K to 30K tokens), a more aggressive decoding constraint approach achieves the best improvement. In both scenarios, the Wikipedia-based approaches are effective in improving the accuracy on unseen entities, especially when a system is applied to a new domain (3.6 F_1 score improvement on political party articles/English NER) or it is trained with little training data (18.3 F_1 score improvement on Japanese NER).

We organize the paper as follows. We describe how to build English Wikipedia entity type mapping in Section 2 and extend it to multilingual mappings in Section 3. We present several Wikipedia-based approaches for improving NER systems in Section 4 and evaluate their performance in Section 5. We conclude the paper in Section 6.

2 English Wikipedia Entity Type Mapping

In this section, we focus on English Wikipedia. We divide Wikipedia pages into two types:

- Entity pages that describe an entity or object, either a named entity such as "Michael Jordan" or a common entity such as "Basketball."
- Non-entity pages that do not describe a certain entity, including disambiguation pages, redirection pages, list pages, etc.

We have developed an in-house English NER system (Florian et al., 2004). The system has 51 entity types, and the main motivation of deploying such a fine-grained entity type set is to build cognitive question answering applications on top of the NER system. An important check for a question answering system is the capability to detect whether a particular answer matches the expected type derived from the question. The entity type system used in this paper has been engineered to cover many of the frequent types that are targeted by naturally-phrased questions (such as PERSON, ORGANIZATION, GPE, TITLEWORK, FACILITY, EVENT, DATE, TIME, LOCATION, etc), and it was created over a long period of time, being updated as more types were found to be useful for question answering, and to improve inter-annotator consistency.

We want to classify Wikipedia pages into one of the entity types used in the NER system. For non-entity pages and entity pages describing common entities, we assign them with a new type OTHER.

2.1 Wikipedia Entity Type Classification

2.1.1 Features

We build maximum entropy classifiers (Nigam et al., 1999) for Wikipedia entity type classification. We use both structured information and unstructured information of a Wikipedia page as features.

Each Wikipedia page has a unique *title*. The title of an entity page is usually the name of the entity, and may include auxiliary information in a bracket to distinguish entities with the same name. We use both the entity name and auxiliary information in a bracket (if any) of a Wikipedia title as features because each could provide useful information for entity type classification. For example, based on the word "Prize" in the title "Nobel Prize" or the word "Awards" in the title "Academy Awards", one can infer that the entity type is AWARD. Likewise, the auxiliary information "company" in the title "Jordan (company)" indicates that the entity is an ORGANIZATION, and the auxiliary information "film" in the title "Alien (film)" indicates that the entity is a TITLEWORK.

The *text* in a Wikipedia page of an entity provides rich information about the entity. A person can usually correctly infer the entity type by reading the first few sentences of the text in a Wikipedia page. Using more sentences provides additional information about the entity which might be helpful, but it is also more likely to introduce noisy information which could affect the classification accuracy adversely. Therefore, we use the first 200 tokens of the text in a Wikipedia page and create n-gram word features out of them. We have also found that including additional n-gram word features of the *first sentence* in a Wikipedia page results in a better classification accuracy.

Most Wikipedia pages also have a structured table called *infobox*, which is placed on the right top of a page. An infobox contains attribute-value pairs, often providing summary information about an entity. The attributes in an infobox could be particularly useful for entity type classification. For example, the attribute "Born" in an infobox provides strong evidence that the corresponding entity is a PERSON; and the attribute "Headquarters" implies that the corresponding entity is an ORGANIZATION. We include the infobox attributes as classifier features.

2.1.2 Training and Test Data

Entity linking (EL) or entity disambiguation is the task of determining the identities of entities mentioned in text, by linking each entity to an entry (if exists) in an open knowledge base such as Wikipedia (Han et al., 2011; Hoffart et al., 2011). We apply an EL system (Sil and Florian, 2014) to generate training data for Wikipedia entity type classification as follows: if a named entity in our NER training data with entity type T is linked to a Wikipedia page, that page will be labeled with entity type T. Similarly, we apply the EL system to generate a set of test data by linking named entities in our NER test data to Wikipedia pages. The English Wikipedia snapshot

Features	ALL	PER	ORG	GPE	TITL	FAC
Title	62.4	73.4	67.2	59.0	57.1	47.1
Infobox	77.3	92.6	87.8	92.0	95.4	50.0
Text	87.2	97.5	87.3	95.1	88.5	40.0
All	90.1	96.1	92.5	95.1	96.9	75.0

Table 1: F_1 score of English Wikipedia entity type classifiers.

Classifier	Train Size	$\mathbf{F_1}$
Original Classifier	4,699	90.1
Self-Training (Standard)	+2,352,836	91.1
Self-Training (Sampling)	+26,518	91.8

Table 2: Improving classifier accuracy via self-training.

was dumped in April 2014 which contains around 4.6M pages. Using this method we generate a training data set with 4,699 English Wikipedia pages and a test set of 415 English Wikipedia pages.

Notice that the automatically generated classifier training and test data are *weakly labeled* since the EL system may link an entity to a wrong Wikipedia page and thus the entity type assigned to that page could be wrong. Since the test data is crucial for evaluating the classification accuracy, we manually corrected the output.

2.1.3 Classifier Performance

To evaluate the prediction power of different types of features, we train a number of classifiers using only title features, only infobox features, only text features, and all features respectively. We show the F_1 score of the classifiers on different entity types in Table 1. ALL is the overall performance, and PER (PERSON), ORG (ORGANIZATION), GPE, TITL (TITLEWORK), FAC (FACILITY) are the top five most frequently entity types in the test data.

From Table 1, we can see that text features are the most important features for classifying Wikipedia pages, since the classifier trained with only text features achieves an overall F_1 score of 87.2, which is better than the classifier trained with either title or infobox features alone. Nevertheless, both infobox and title features provide additional useful information for entity type classification, and the classifier trained with all the features achieves an overall F_1 score of 90.1.

2.1.4 Improvement via Self-Training

Self-training is a *semi-supervised* learning technique that can be used in applications where there is only a small number of labeled training examples but a large number of unlabeled examples. Since our weakly annotated classifier training data only covers around 1% of all the Wikipedia pages, we are motivated to use self-training to further improve the classification accuracy.

We first apply a standard self-training approach. The classifier trained with the initial training data is used to decode (i.e., classify) all the unlabeled Wikipedia pages to predict their entity types with confidence scores. We add the self-decoded Wikipedia pages with high confidence scores to the training data and train a new classifier. Via experiments a threshold of 0.9 is used to sort out high-confident self-decoded examples. The F_1 score of the new classifier is improved to 91.1, as shown in Table 2.

Under the standard approach, about 2.3M self-decoded examples are added, the size of which is about 500 times of the size of the original training data. The errors of the original classifier could be amplified with such a big increase of the training size with so many self-decoded examples.

To address this issue, we have developed a *sampling-based* self-training approach. Instead of adding all the self-decoded examples with confidence scores greater than or equal to 0.9, we do a random sampling of those high-confident examples. We use a sampling probability $p(e) = q \cdot c(e)$, where q is a sampling ratio parameter and $c(e)$ is the confidence score of example e. Under this approach, examples with higher confidence scores are more likely to be selected, while the total number of selected examples is controlled by the sampling ratio q. Via experiments we found that a small sampling ratio like $q = 0.01$ yields good improvement (although the improvement is not sensitive to q). As shown in Table 2, the classification accuracy under the sampling-based approach is further improved to 91.8 F_1 score (the improvement is calculated by averaging over 5 random samples with $q = 0.01$).

2.2 Wikipedia Entity Type Mapping

We construct an English Wikipedia entity type mapping by applying the English Wikipedia entity type classifier on all the English Wikipedia pages

(∼4.6M). Each entry of the mapping includes an *entity name* (which is extracted from the title of a Wikipedia page) and the associated *entity type* with *confidence score* (which is determined by the classifier). We denote the English Wikipedia entity type mapping that includes all the pages by *English-Wiki-Mapping*.

To build a high-accuracy mapping, one may want to include only entities with confidence scores greater than or equal to a threshold t in the mapping, and we denote such a mapping by *English-Wiki-Mapping*(t). Notice that a mapping with a higher t will have more accurate entity types for its entities, but it will include fewer entities. Therefore, there is a trade-off between *accuracy* and *coverage* of the mapping, which can be tuned by the confidence threshold t. There are about 2.9M entities in *English-Wiki-Mapping*(0.9), which covers about 63% of all the English Wikipedia pages.

We have also found that the *length* of an entity name (i.e., number of words in an entity name) also plays an important role for determining which entities should be included in the mapping for improving an NER system. Therefore, we use *English-Wiki-Mapping*(t, i) to denote the English Wikipedia entity type mapping that includes all the entities with confidence scores greater than or equal to t and at least i words in their names. *English-Wiki-Mapping*(0.9,2) covers about 55% of all the English Wikipedia pages, and *English-Wiki-Mapping*(0.9,3) covers about 25% of all the English Wikipedia pages.

3 Multilingual Wikipedia Entity Type Mapping

Based on the English Wikipedia entity type mapping, we want to build high-accuracy, high-coverage Wikipedia entity type mappings for other languages with minimum human annotation and language-dependent knowledge involved. We utilize the *inter-language links* of Wikipedia, which are the links between one entity's pages in different languages. The inter-language links between English Wikipedia pages and Wikipedia pages of another language provide useful information for this task.

Suppose we want to build a Wikipedia entity type mapping for a new language, and we use Portuguese as an example. A direct approach is *projection* using the inter-language links between English and Portuguese Wikipedia pages: for each Portuguese Wikipedia page that has an inter-language link to an English Wikipedia page, we project the entity type of the English Wikipedia page (determined by the English entity type mapping) to the Portuguese Wikipedia page. The rationale is that both the English and Portuguese pages are describing the same entity, even probably with different spelling (e.g., **United States** in English vs. **Estados Unidos** in Portuguese), the entity type of that entity does not change from one language to another.

However, the main limitation of the direct projection approach is coverage. Only a fraction of all the Portuguese Wikipedia pages have inter-language links to English Wikipedia pages, and among those pages only a subset of them have classified entity types with confidence scores high enough (e.g., at least 0.9). For example, projecting *English-Wiki-Mapping*(0.9) to Portuguese Wikipedia returns 143K pages, which covers only 15% of all the Portuguese Wikipedia pages (around 920K in total).

We apply an alternative approach, which uses the 143K Portuguese Wikipedia pages (acquired by projection from *English-Wiki-Mapping*(0.9)) as weakly annotated training data to train a Portuguese Wikipedia entity type classifier. For feature engineering purpose, we also project the English Wikipedia entity type classifier training and test data (as described in Section 2.1.2) to Portuguese Wikipedia pages via inter-language links, and this produces 1,190 Portuguese Wikipedia pages which are used as the test data. Pages in the test data set are excluded from the 143K training data set.

We use similar features (title, infobox and text) as for the English classifiers to train the Portuguese classifiers. Again we find that the classifier trained with all the features achieves the best accuracy of 86.3 F_1 score. Notice that this is an approximated evaluation because the pages in the test data set are labeled via projection and not by human.

We build Portuguese Wikipedia entity type mappings by applying the Portuguese Wikipedia entity type classifier on all the Portuguese Wikipedia pages. We use *Portuguese-Wiki-Mapping*(t) to denote the mapping that includes entities with confidence scores greater than or equal to a thresh-

old t. There are 525K entities in *Portuguese-Wiki-Mapping*(0.9), which covers about 57% of all the Portuguese Wikipedia pages, a significant improvement of coverage compared to the direct projection approach (15%).

The main advantage of our approach is that no human annotation or language-dependent knowledge is required, so it can be easily applied to a new language. We have applied this approach to build high-accuracy, high-coverage Wikipedia entity type mappings for several new languages including Portuguese, Japanese, Spanish, Dutch and German.

4 Improving NER Systems

We have developed several approaches that utilize the Wikipedia entity type mappings to improve NER systems. Let $\mathcal{M}$ be a Wikipedia entity type mapping. For an entity name x, let $\mathcal{M}(x)$ denote the set of possible entity types for x determined by the mapping. If an entity name x is in the mapping, then $\mathcal{M}(x)$ includes at least one entity type, i.e., $|\mathcal{M}(x)| \geq 1$, where $|\mathcal{M}(x)|$ is the cardinality of $\mathcal{M}(x)$. Otherwise if an entity name x is not in the mapping, then $\mathcal{M}(x) = \emptyset$ is the empty set and $|\mathcal{M}(x)| = 0$.

The first approach is to use a Wikipedia entity type mapping $\mathcal{M}$ as a *decoding constraint* for an NER system. Under this approach, the mapping is applied as a constraint during the decoding procedure: if a sequence of words in the text form an entity name x that is included in the mapping, i.e., $|\mathcal{M}(x)| \geq 1$, then the sequence of words will be identified as an entity, and its entity type is determined by the decoding algorithm while being constrained to one of the entity types in $\mathcal{M}(x)$.

The second approach is to use a Wikipedia entity type mapping $\mathcal{M}$ to *post-process* the output of an NER system. Under this approach, the mapping is applied after the decoding procedure: if the name of a system entity x is in the mapping and the entity type for that entity name is unique based on the mapping, i.e., $|\mathcal{M}(x)| = 1$, then its entity type will be determined by the unique entity type in $\mathcal{M}(x)$.

The decoding constraint approach is more aggressive than the post-processing approach, because it may create new entities and change entity boundaries. This approach is more reliable for entities with longer names. Via experiments we find that using *Wiki-Mapping*(0.9,2) or *Wiki-Mapping*(0.9,3) achieves the best improvement under the decoding constraint approach. Remember *Wiki-Mapping*(t, i) includes all the entities with confidence scores at least t and at least i words in their names.

In contrast, the post-processing approach is a more conservative approach since it relies on the system entity boundaries and only changes their entity types if determined by the mapping, so it will not create new entities. Via experiments we find that using *Wiki-Mapping*(0.9,2) achieves the best improvement under the post-processing approach.

Based on the observation that the decoding constraint approach is more reliable for longer entities while the post-processing approach can better handle short entities, we have designed a *joint* approach that combines the two approaches as follows: it first applies *Wiki-Mapping*(0.9,3) as a decoding constraint for an NER system to produce system entities, and then applies *Wiki-Mapping*(0.9,2) to post-process the system output. The joint approach combines the advantages of both approaches and achieves robust performance in our experiments.

Finally, we can use a Wikipedia entity type mapping to create *dictionary features* for training an NER system. The idea of using Wikipedia to create training features was explored before, e.g., (Kazama and Torisawa, 2007; Ratinov and Roth, 2009; Radford et al., 2015). The difference between our approach and the previous approaches is how the features are created: we first build high-accuracy, high-coverage multilingual Wikipedia entity type mappings and then use the mappings to generate dictionary features. Via experiments we find that using *Wiki-Mapping*(0.9,1) or *Wiki-Mapping*(0.9,2) achieves the best improvement under the dictionary feature approach.

5 Experiments

In this section, we evaluate the effectiveness of the proposed Wikipedia-based approaches via experiments on NER systems trained for 6 languages: English, Portuguese, Japanese, Spanish, Dutch and German. For each language, we compare the baseline NER system with the following approaches:

- DC(i): the decoding constraint approach with

mapping *Language-Wiki-Mapping*(0.9,i).

- PP(i): the post-processing approach with mapping *Language-Wiki-Mapping*(0.9,i).
- Joint: the joint approach that combines DC(3) and PP(2).
- DF(i): the dictionary feature approach with mapping *Language-Wiki-Mapping*(0.9,i).

To evaluate the generalization capability of an NER system, we compute the F_1 score on the unseen entities (*Unseen*) as well as on all the entities (*All*) in a test data set.

5.1 English

The baseline English NER system is a CRF model trained with 328K tokens of human-annotated news articles. It uses standard NER features in the literature including n-gram word features, word type features, prefix and suffix features, Brown cluster type features, gazetteer features, document-level cache features, etc.

We have two human-annotated test data sets: the first set, Test (News), consists of 40K tokens of human-annotated news articles; and the second set, Test (Political), consists of 77K tokens of human-annotated political party articles from Wikipedia. The results are shown in Table 3.

For Test (News) which is in the same domain as the training data, the baseline system achieves 88.2 F_1 score on all the entities, and a relatively low F_1 score of 78.7 on the unseen entities (38% of all the entities are unseen entities). The dictionary feature approach DF(2) achieves the highest F_1 scores among the Wikipedia-based approaches. It improves the baseline system by 1.2 F_1 score on all the entities and by 3.1 F_1 score on the unseen entities. The joint approach achieves the second highest F_1 scores. It improves the baseline by 0.7 F_1 score on all the entities and by 2.0 F_1 score on the unseen entities.

For Test (Political) which is in a different domain from the training data, the fraction of unseen entities increases to 84%. In this case, the F_1 score of the baseline system drops to 64.1, and the Wikipedia-based approaches demonstrate larger improvements. For example, DF(2) improves the baseline system by 2.7 F_1 score on all the entities and by 3.6 F_1 score on the unseen entities.

NER System	**Test (News)**		**Test (Political)**	
	All 100%	*Unseen* 38%	*All* 100%	*Unseen* 84%
Baseline	88.2	78.7	64.1	60.9
DC(2)	88.1	79.4	66.3	63.5
DC(3)	88.7	80.2	65.8	62.9
PP(2)	88.6	79.8	64.7	61.7
Joint	88.9	80.7	66.3	63.6
DF(1)	88.5	80.0	66.3	64.2
DF(2)	**89.4**	**81.8**	**66.8**	**64.5**

Table 3: Experimental results for English NER (the highest F_1 score among all approaches in a column is shown in bold).

5.2 Portuguese

For Portuguese, we have applied a semi-supervised learning approach to build the baseline NER system. The training data set includes 31K tokens of human-annotated news articles, and 2M tokens of weakly annotated data. The weakly annotated data is generated as follows. We have a large number of parallel sentences between English and Portuguese news articles. We apply the English NER system on the English sentences and project the entity type tags to the Portuguese sentences via alignments between the English and Portuguese sentences.

The baseline NER system is an MEMM model (CRF cannot handle such a big size of training data, since our NER system has 51 entity types, and the number of features and training time of CRF grow at least quadratically in the number of entity types). The test data set consists of 34K tokens of human-annotated Portuguese news articles.

The results are shown in Table 4. Because the system is trained with little human-annotated training data, the performance of the baseline system achieves only 60.1 F_1 score on all the entities and 50.2 F_1 score on the unseen entities (80% of all the entities). In this case, the more aggressive decoding constraint approach DC(2) achieves the best improvement among the Wikipedia-based approaches, which improves the baseline by 5.9 F_1 score on all the entities and by 8.6 F_1 score on the unseen entities. The joint approach improves the baseline by 3.0 F_1 score on all the entities and by 4.3 F_1 score on the unseen entities.

NER System	Test (News)	
	All 100%	*Unseen* 80%
Baseline	60.1	50.2
DC(2)	**66.0**	**58.8**
DC(3)	62.2	53.4
PP(2)	60.9	51.4
Joint	63.1	54.5
DF(1)	62.4	52.7
DF(2)	61.3	51.9

Table 4: Experimental results for Portuguese NER.

NER System	Test (News)	
	All 100%	*Unseen* 59%
Baseline	50.8	27.3
DC(2)	**59.8**	**45.6**
DC(3)	55.6	36.9
PP(2)	50.8	27.3
Joint	55.6	36.9
DF(1)	52.9	29.0
DF(2)	51.8	28.0

Table 5: Experimental results for Japanese NER.

5.3 Japanese

For Japanese, the baseline NER system is an MEMM model trained with 20K tokens of human-annotated news articles and 2.1M tokens of weakly annotated data. The weakly annotated data was generated using similar steps as for the Portuguese NER system. The test data set consists of 22K tokens of human-annotated Japanese news articles.

The results are shown in Table 5. Again, in this low-resource case, DC(2) achieves the best improvement among the Wikipedia-based approaches. It improves the baseline by 9.0 F_1 score on all the entities and by 18.3 F_1 score on the unseen entities (59% of all the entities). The joint approach improves the baseline by 4.8 F_1 score on all the entities and by 9.6 F_1 score on the unseen entities.

5.4 Spanish, Dutch and German

We also evaluate the Wikipedia-based approaches on Spanish, Dutch and German NER systems trained with the CoNLL data sets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). There are only 4 entity types in the CoNLL data: PER (person), ORG (organization), LOC (location), MISC (miscellaneous names). Accordingly, we have trained a CoNLL-style Wikipedia entity type classifier that produces the CoNLL entity types. The training data for the classifier is generated by using the CoNLL English training data set and the AIDA-YAGO2 data set that provides the Wikipedia titles for the named entities in the CoNLL English data set (Hoffart et al., 2011). Applying the classifier on all the English Wikipedia pages, we construct a CoNLL-style English Wikipedia entity type mapping. We then build CoNLL-style Wikipedia entity type mappings for Spanish, Dutch and German using steps as described in Section 3.

For each of the three languages, the baseline NER system is a CRF model trained with human-annotated news data ($\sim$200K tokens), and there are two test data sets, TestA and TestB, that are also human-annotated news data (ranging from 40K to 70K tokens). The results are shown in Table 6. For Dutch and German, DF(1) achieves the best improvement among the Wikipedia-based approaches. For Spanish, the joint approach achieves the best improvement among the Wikipedia-based approaches. Again, in all cases, the Wikipedia-based approaches demonstrate larger improvements (ranging from 1.0 to 3.4 F_1 score) on the unseen entities.

5.5 Discussion

From the experimental results, we have the following observations:

- NER systems are more likely to make mistakes on unseen entities. In all cases, the F_1 score of an NER system on all the entities is always higher than the F_1 score on the unseen entities.
- The Wikipedia-based approaches are effective in improving the generalization capability of NER systems (i.e., improving the accuracy on unseen entities), especially when a system is applied to a new domain (3.6 F_1 score improvement on political party articles/English NER) or it is trained with little human-annotated training data (18.3 F_1 score improvement on Japanese NER).
- In the low-resource scenario where an NER

NER System	TestA		TestB	
	All	*Unseen*	*All*	*Unseen*
Spanish	100%	47%	100%	38%
Baseline	77.9	69.4	81.5	71.0
DC(2)	77.9	69.7	81.4	71.0
DC(3)	78.4	70.1	81.6	71.2
PP(2)	78.2	70.1	**82.0**	**72.1**
Joint	**78.5**	**70.4**	**82.0**	**72.1**
DF(1)	77.7	69.6	**82.0**	71.6
DF(2)	**78.5**	**70.4**	81.4	70.9
Dutch	100%	60%	100%	54%
Baseline	80.7	70.8	82.3	70.9
DC(2)	80.8	71.3	82.8	71.9
DC(3)	80.8	71.2	82.4	71.1
PP(2)	81.2	71.6	83.2	72.5
Joint	81.3	71.9	83.1	72.3
DF(1)	**82.3**	**73.2**	**84.5**	**74.3**
DF(2)	81.1	71.1	83.3	72.5
German	100%	72%	100%	70%
Baseline	69.6	63.0	70.3	63.0
DC(2)	70.1	63.8	70.1	62.8
DC(3)	69.9	63.5	70.4	63.1
PP(2)	70.5	64.4	70.6	63.4
Joint	70.8	64.8	70.6	63.4
DF(1)	**71.8**	**65.8**	**71.8**	**65.3**
DF(2)	71.2	65.4	70.5	63.6

Table 6: Experimental results for Spanish, Dutch, and German NER.

system is trained with little human-annotated data (e.g., 20K-30K tokens of training data for the Portuguese and Japanese systems), the decoding constraint approach, which uses a high-accuracy, high-coverage Wikipedia entity type mapping to create constraints during the decoding phase, achieves the best improvement.

- In the rich-resource scenario where an NER system is well trained (e.g., 200K-300K tokens of training data for the English, Dutch and German systems), the dictionary feature approach, which uses a Wikipedia entity type mapping to create dictionary type features, achieves the best improvement.

- In both scenarios, the joint approach, which combines the decoding constraint approach and the post-processing approach in a smart way, achieves relatively robust performance among the Wikipedia-based approaches.

6 Conclusion

In this paper, we proposed and evaluated several approaches that utilize high-accuracy, high-coverage Wikipedia entity type mappings to improve multilingual NER systems. These mappings are built from weakly annotated data, and can be easily extended to new languages with no human annotation or language-dependent knowledge involved.

Experimental results show that the Wikipedia-based approaches are effective in improving the generalization capability of NER systems. When a system is well trained, the dictionary feature approach achieves the best improvement over the baseline system; while when a system is trained with little human-annotated training data, a more aggressive decoding constraint approach achieves the best improvement. The improvements are larger on unseen entities, and the approaches are especially useful when a system is applied to a new domain or it is trained with little training data.

Acknowledgments

We would like to thank Avirup Sil for helpful comments, and for collecting the Wikipedia data. We also thank the anonymous reviewers for their suggestions.

References

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

Wisam Dakka and Silviu Cucerzan. 2008. Augmenting Wikipedia with named entity tags. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 545–552, Hyderabad, India.

Radu Florian, Hany Hassan, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technologies Conference 2004 (HLT-NAACL'04)*, pages 1–8, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 765–774, New York, NY, USA. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707, Prague, Czech Republic, June. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT (NAACL 2016)*, San Diego, US.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from Wikipedia. *Journal of Artificial Intelligence*, 194:151–175, January.

Will Radford, Xavier Carreras, and James Henderson. 2015. Named entity recognition with document-specific KB tag gazetteers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 512–517, Lisbon, Portugal, September. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Alexander E. Richman and Patrick Schone. 2008. Mining Wiki resources for multilingual named entity recognition. In *Proceedings of ACL-08: HLT*, pages 1–9, Columbus, Ohio, June. Association for Computational Linguistics.

Avirup Sil and Radu Florian. 2014. The IBM systems for English entity discovery and linking and Spanish entity linking at TAC 2014. In *Text Analysis Conference (TAC)*, Gaithersburg, Maryland, USA.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the Sixth Conference on Natural Language Learning - Volume 20*, CONLL '02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Antonio Toral and Rafael Muoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. In *EACL 2006*.

Learning Crosslingual Word Embeddings without Bilingual Corpora

Long Duong,[12] **Hiroshi Kanayama,**[3] **Tengfei Ma,**[3] **Steven Bird**[14] and **Trevor Cohn**[1]
[1]Department of Computing and Information Systems, University of Melbourne
[2]National ICT Australia, Victoria Research Laboratory
[3]IBM Research – Tokyo
[4]International Computer Science Institute, University of California Berkeley

Abstract

Crosslingual word embeddings represent lexical items from different languages in the same vector space, enabling transfer of NLP tools. However, previous attempts had expensive resource requirements, difficulty incorporating monolingual data or were unable to handle polysemy. We address these drawbacks in our method which takes advantage of a high coverage dictionary in an EM style training algorithm over monolingual corpora in two languages. Our model achieves state-of-the-art performance on bilingual lexicon induction task exceeding models using large bilingual corpora, and competitive results on the monolingual word similarity and cross-lingual document classification task.

1 Introduction

Monolingual word embeddings have had widespread success in many NLP tasks including sentiment analysis (Socher et al., 2013), dependency parsing (Dyer et al., 2015), machine translation (Bahdanau et al., 2014). Crosslingual word embeddings are a natural extension facilitating various crosslingual tasks, e.g. through transfer learning. A model built in a source resource-rich language can then applied to the target resource poor languages (Yarowsky and Ngai, 2001; Das and Petrov, 2011; Täckström et al., 2012; Duong et al., 2015). A key barrier for crosslingual transfer is lexical matching between the source and the target language. Crosslingual word embeddings are a natural remedy where both source and target language lexicon are presented as dense vectors in the same vector space (Klementiev et al., 2012).

Most previous work has focused on down-stream crosslingual applications such as document classification and dependency parsing. We argue that good crosslingual embeddings should preserve both monolingual and crosslingual quality which we will use as the main evaluation criterion through monolingual word similarity and bilingual lexicon induction tasks. Moreover, many prior work (Chandar A P et al., 2014; Kočiský et al., 2014) used bilingual or comparable corpus which is also expensive for many low-resource languages. Søgaard et al. (2015) impose a less onerous data condition in the form of linked Wikipedia entries across several languages, however this approach tends to underperform other methods. To capture the monolingual distributional properties of words it is crucial to train on large monolingual corpora (Luong et al., 2015). However, many previous approaches are not capable of scaling up either because of the complicated objective functions or the nature of the algorithm. Other methods use a dictionary as the bridge between languages (Mikolov et al., 2013a; Xiao and Guo, 2014), however they do not adequately handle translation ambiguity.

Our model uses a bilingual dictionary from Panlex (Kamholz et al., 2014) as the source of bilingual signal. Panlex covers more than a thousand languages and therefore our approach applies to many languages, including low-resource languages. Our method selects the translation based on the context in an Expectation-Maximization style training algorithm which explicitly handles polysemy through incorporating multiple dictionary translations (word sense and translation are closely linked (Resnik and Yarowsky, 1999)). In addition to the dictionary,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1285–1295,
Austin, Texas, November 1-5, 2016.

our method only requires monolingual data. Our approach is an extension of the continuous bag-of-words (CBOW) model (Mikolov et al., 2013b) to inject multilingual training signal based on dictionary translations. We experiment with several variations of our model, whereby we predict only the translation or both word and its translation and consider different ways of using the different learned center-word versus context embeddings in application tasks. We also propose a regularisation method to combine the two embedding matrices during training. Together, these modifications substantially improve the performance across several tasks. Our final model achieves state-of-the-art performance on bilingual lexicon induction task, large improvement over word similarity task compared with previous published crosslingual word embeddings, and competitive result on cross-lingual document classification task. Notably, our embedding combining techniques are general, yielding improvements also for monolingual word embedding.

This paper makes the following contributions:

- Proposing a new crosslingual training method for learning vector embeddings, based only on monolingual corpora and a bilingual dictionary;
- Evaluating several methods for combining embeddings, which are shown to help in both crosslingual and monolingual evaluations; and
- Achieving consistent results which are competitive in monolingual, bilingual and crosslingual transfer settings.

2 Related work

There is a wealth of prior work on crosslingual word embeddings, which all exploit some kind of bilingual resource. This is often in the form of a parallel bilingual text, using word alignments as a bridge between tokens in the source and target languages, such that translations are assigned similar embedding vectors (Luong et al., 2015; Klementiev et al., 2012). These approaches are affected by errors from automatic word alignments, motivating other approaches which operate at the sentence level (Chandar A P et al., 2014; Hermann and Blunsom, 2014; Gouws et al., 2015) through learning compositional vector representations of sentences, in order that sentences and their translations representations closely match. The word embeddings learned this way capture translational equivalence, despite not using explicit word alignments. Nevertheless, these approaches demand large parallel corpora, which are not available for many language pairs.

Vulić and Moens (2015) use bilingual comparable text, sourced from Wikipedia. Their approach creates a psuedo-document by forming a bag-of-words from the lemmatized nouns in each comparable document concatenated over both languages. These pseudo-documents are then used for learning vector representations using `Word2Vec`. Their system, despite its simplicity, performed surprisingly well on a bilingual lexicon induction task (we compare our method with theirs on this task.) Their approach is compelling due to its lesser resource requirements, although comparable bilingual data is scarce for many languages. Related, Søgaard et al. (2015) exploit the comparable part of Wikipedia. They represent word using Wikipedia entries which are shared for many languages.

A bilingual dictionary is an alternative source of bilingual information. Gouws and Søgaard (2015) randomly replace the text in a monolingual corpus with a random translation, using this corpus for learning word embeddings. Their approach doesn't handle polysemy, as very few of the translations for each word will be valid in context. For this reason a high coverage or noisy dictionary with many translations might lead to poor outcomes. Mikolov et al. (2013a), Xiao and Guo (2014) and Faruqui and Dyer (2014) filter a bilingual dictionary for one-to-one translations, thus side-stepping the problem, however discarding much of the information in the dictionary. Our approach also uses a dictionary, however we use all the translations and explicitly disambiguate translations during training.

Another distinguishing feature on the above-cited research is the method for training embeddings. Mikolov et al. (2013a) and Faruqui and Dyer (2014) use a cascade style of training where the word embeddings in both source and target language are trained separately and then combined later using the dictionary. Most of the other works train multlingual models jointly, which appears to have better performance over cascade training (Gouws et al., 2015).

For this reason we also use a form of joint training in our work.

3 Word2Vec

Our model is an extension of the contextual bag of words (CBOW) model of Mikolov et al. (2013b), a method for learning vector representations of words based on their distributional contexts. Specifically, their model describes the probability of a token w_i at position i using logistic regression with a factored parameterisation,

$$p(w_i|w_{i\pm k\setminus i}) = \frac{\exp(\mathbf{u}_{w_i}^\top \mathbf{h}_i)}{\sum_{w\in W}\exp(\mathbf{u}_w^\top \mathbf{h}_i)}\,, \qquad (1)$$

where $\mathbf{h}_i = \frac{1}{2k}\sum_{j=-k;j\neq 0}^{k}\mathbf{v}_{w_{i+j}}$ is a vector encoding the context over a window of size k centred around position i, W is the vocabulary and the parameters $\mathbf{V}$ and $\mathbf{U} \in \mathbb{R}^{|W|\times d}$ are matrices referred to as the context and word embeddings. The model is trained to maximise the log-pseudo likelihood of a training corpus, however due to the high complexity of computing the denominator of equation (1), Mikolov et al. (2013b) propose negative sampling as an approximation, by instead learning to differentiate data from noise (negative examples). This gives rise to the following optimisation objective

$$\sum_{i\in D}\left(\log\sigma(\mathbf{u}_{w_i}^\top \mathbf{h}_i) + \sum_{j=1}^{p}\mathbb{E}_{w_j\sim P_n(w)}\log\sigma(-\mathbf{u}_{w_j}^\top \mathbf{h}_i)\right), \qquad (2)$$

where D is the training data and p is the number of negative examples randomly drawn from a noise distribution $P_n(w)$.

4 Our Approach

Our approach extends CBOW to model bilingual text, using two monolingual corpora and a bilingual dictionary. We believe this data condition to be less stringent than requiring parallel or comparable texts as the source of the bilingual signal. It is common for field linguists to construct a bilingual dictionary when studying a new language, as one of the first steps in the language documentation process. Translation dictionaries are a rich information source, capturing much of the lexical ambiguity in a language through translation. For example, the word *bank* in English might mean the *river bank* or *financial bank* which corresponds to two different translations *sponda* and *banca* in Italian. If we are able to learn to select good translations, then this implicitly resolves much of the semantic ambiguity in the language, and accordingly we seek to use this idea to learn better semantic vector representations of words.

Algorithm 1 EM algorithm for selecting translation during training, where $\theta = (\mathbf{U}, \mathbf{V})$ are the model parameters and η is the learning rate.

randomly initialize $\mathbf{V}$, $\mathbf{U}$
for $i <$ `Iter` **do**
  for $i \in D_e \cup D_f$ **do**
    $\mathbf{s} \leftarrow \mathbf{v}_{w_i} + \mathbf{h}_i$
    $\bar{w}_i = \text{argmax}_{w\in\text{dict}(w_i)}\cos(\mathbf{s}, \mathbf{v}_w)$
    $\theta \leftarrow \theta + \eta\frac{\partial\mathcal{O}(\bar{w}_i, w_i, \mathbf{h}_i)}{\partial\theta}$ {see (3) or (5)}
  end for
end for

4.1 Dictionary replacement

To learn bilingual relations, we use the context in one language to predict the translation of the centre word in another language. This is motivated by the fact that the context is an excellent means of disambiguating the translation for a word. Our method is closely related to Gouws and Søgaard (2015), however we only replace the middle word w_i with a translation $\bar{w}_i$ while keeping the context fixed. We replace each centre word with a translation on the fly during training, predicting instead $p(\bar{w}_i|w_{i\pm k\setminus i})$ but using the same formulation as equation (1) albeit with an augmented $\mathbf{U}$ matrix to cover word types in both languages.

The translation $\bar{w}_i$ is selected from the possible translations of w_i listed in the dictionary. The problem of selecting the correct translation from the many options is reminiscent of the problem faced in expectation maximisation (EM), in that cross-lingual word embeddings will allow for accurate translation, however to learn these embeddings we need to know the translations. We propose an EM-inspired algorithm, as shown in Algorithm 1, which operates over both monolingual corpora, D_e and D_f. The vector $\mathbf{s}$ is the semantic representation combining both the centre word, w_i, and the con-

text,[1] which is used to choose the best translation into the other language from the bilingual dictionary $dict(w_i)$.[2] After selecting the translation, we use $\bar{w}_i$ together with the context vector $\mathbf{h}$ to make a stochastic gradient update of the CBOW log-likelihood.

4.2 Joint Training

Words and their translations should appear in very similar contexts. One way to enforce this is to jointly learn to predict both the word and its translation from its monolingual context. This gives rise to the following joint objective function,

$$\mathcal{O} = \sum_{i \in D_e \cup D_f} \Big(\alpha \log \sigma(\mathbf{u}_{w_i}^\top \mathbf{h}_i) + (1-\alpha) \log \sigma(\mathbf{u}_{\bar{w}_i}^\top \mathbf{h}_i) + \sum_{j=1}^{p} \mathbb{E}_{w_j \sim P_n(w)} \log \sigma(-\mathbf{u}_{w_j}^\top \mathbf{h}_i) \Big), \quad (3)$$

where α controls the contribution of the two terms. For our experiments, we set $\alpha = 0.5$. The negative examples are drawn from combined vocabulary unigram distribution calculated from combined data $D_e \cup D_f$.

4.3 Combining Embeddings

Many vector learning methods learn two embedding spaces $\mathbf{V}$ and $\mathbf{U}$. Usually only $\mathbf{V}$ is used in application. The use of $\mathbf{U}$, on the other hand, is understudied (Levy and Goldberg, 2014) with the exception of Pennington et al. (2014) who use a linear combination $\mathbf{U} + \mathbf{V}$, with minor improvement over $\mathbf{V}$ alone.

We argue that with our model, $\mathbf{V}$ is better at capturing the monolingual regularities and $\mathbf{U}$ is better at capturing bilingual signal. The intuition for this is as follows. Assuming that we are predicting the word *finance* and its Italian translation *finanze* from the context (*money, loan, bank, debt, credit*) as shown in figure 1. In $\mathbf{V}$ only the context word representations are updated and in $\mathbf{U}$ only the representations of *finance, finanze* and negative samples such as *tree* and *dog* are updated. CBOW learns good embeddings because each time it updates the parameters, the words in the contexts are pushed closer to each other in the $\mathbf{V}$ space. Similarly, the target word w_i and the translation $\bar{w}_i$ are also pushed closer in the $\mathbf{U}$ space. This is directly related to poitwise mutual information values of each pair of word and context explained in Levy and Goldberg (2014). Thus, $\mathbf{U}$ is bound to better at bilingual lexicon induction task and $\mathbf{V}$ is better at monolingual word similarity task.

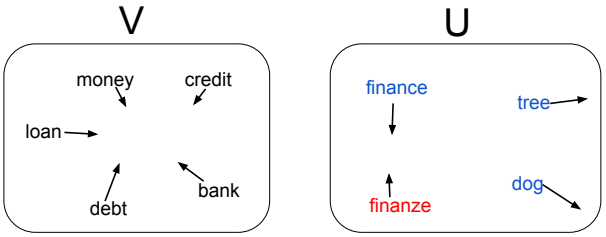


Figure 1: Example of $\mathbf{V}$ and $\mathbf{U}$ space during training.

The simple question is, how to combine both $\mathbf{V}$ and $\mathbf{U}$ to produce a better representation. We experiment with several ways to combine $\mathbf{V}$ and $\mathbf{U}$. First, we can follow Pennington et al. (2014) to *interpolate* $\mathbf{V}$ and $\mathbf{U}$ in the post-processing step. i.e.

$$\gamma \mathbf{V} + (1-\gamma)\mathbf{U} \quad (4)$$

where γ controls the contribution of each embedding space. Second, we can also *concatenate* $\mathbf{V}$ and $\mathbf{U}$ instead of interpolation such that $\mathbf{C} = [\mathbf{V} : \mathbf{U}]$ where $\mathbf{C} \in \mathbb{R}^{|W| \times 2d}$ and W is the combined vocabulary from $D_e \cup D_f$.

Moreover, we can also fuse $\mathbf{V}$ and $\mathbf{U}$ during training. For each word in the combined dictionary $V_e \cup V_f$, we encourage the model to learn similar representation in both $\mathbf{V}$ and $\mathbf{U}$ by adding a *regularization* term to the objective function in equation (3) during training.

$$\mathcal{O}' = \mathcal{O} + \delta \sum_{w \in V_e \cup V_f} \|\mathbf{u}_w - \mathbf{v}_w\|_2^2 \quad (5)$$

where δ controls to what degree we should bind two spaces together.[3]

5 Experimental Setup

Our experimental evaluation seeks to determine how well lexical distances in the learned embedding

[1] Using both embeddings gives a small improvement compared to just using context vector $\mathbf{h}$ alone.

[2] We also experimented with using expectations over translations, as per standard EM, with slight degredation in results.

[3] In the stochastic gradient update for a given word in context, we only compute the gradient of the regularisation term in (5) with respect to the words in the set of positive and negative examples.

spaces match with known lexical similarity judgements from bilingual and monolingual lexical resources. To this end, in §6 we test crosslingual distances using a bilingual lexicon induction task in which we evaluate the embeddings in terms of how well nearby pairs of words from two languages in the embedding space match with human judgements. Next, to evaluate the monolingual embeddings we evaluate word similarities in a single language against standard similarity datasets (§7). Lastly, to demonstrate the usefulness of our embeddings in a task-based setting, we evaluate on crosslingual document classification (§9).

Monolingual Data The monolingual data is taken from the pre-processed Wikipedia dump from Al-Rfou et al. (2013). The data is already cleaned and tokenized. We additionally lower-case all words. Normally monolingual word embeddings are trained on billions of words. However, obtaining that much monolingual data for a low-resource language is infeasible. Therefore, we only select the first 5 million sentences (around 100 million words) for each language.

Dictionary A bilingual dictionary is the only source of bilingual correspondence in our technique. We prefer a dictionary that covers many languages, such that our approach can be applied widely to many low-resource languages. We use Panlex, a dictionary which currently covers around 1300 language varieties with about 12 million expressions. The translations in PanLex come from various sources such as glossaries, dictionaries, automatic inference from other languages, etc. Accordingly, Panlex has high language coverage but often noisy translations.[4] Table 1 summarizes the sizes of monolingual corpora and dictionaries for each pair of language in our experiments.

[4]We also experimented with a crowd-sourced dictionary from Wiktionary. Our initial observation was that the translation quality was better but with a lower-coverage. For example, for `en-it` dictionary, Panlex and Wiktionary have a coverage of 42.1% and 16.8% respectively for the top 100k most frequent English words from Wikipedia. The average number of translations are 5.2 and 1.9 respectively. We observed similar trend using Panlex and Wiktionary dictionary in our model. However, using Panlex results in much better performance. We can run the model on the combined dictionary from both Panlex and Wiktionary but we leave it for future work.

	Source (M)	Target (M)	Dict (k)
`en-es`	120.1 (73.9%)	126.8 (74.4%)	712.0
`en-it`	120.1 (74.7%)	114.6 (67.4%)	560.1
`en-nl`	120.1 (69.1%)	80.2 (63.4%)	406.6
`en-de`	120.1 (77.8%)	90.8 (68.3%)	964.4
`en-sr`	120.1 (28.0%)	7.5 (17.5%)	35.1

Table 1: Number of tokens in millions for the source and target languages in each language pair. Also shown is the number of entries in the bilingual dictionary in thousands. The number in the parenthesis shows the token coverage in the dictionary on each monolingual corpus.

6 Bilingual Lexicon Induction

Given a word in a source language, the bilingual lexicon induction (BLI) task is to predict its translation in the target language. Vulić and Moens (2015) proposed this task to test crosslingual word embeddings. The difficulty of this is that it is evaluated using the recall of the top ranked word. The model must be very discriminative in order to score well.

We build the CLWE for 3 language pairs: `it-en`, `es-en` and `nl-en`, using similar parameters setting with Vulić and Moens (2015).[5] The remaining tunable parameters in our system are δ from Equation (5), and the choice of algorithm for combining embeddings. We use the regularization technique from §4.3 for combining context and word embeddings with $\delta = 0.01$, and word embeddings $\mathbf{U}$ are used as the output for all experiments (but see comparative experiments in §8.)

Qualitative evaluation We jointly train the model to predict both w_i and the translation $\bar{w_i}$, combine $\mathbf{V}$ and $\mathbf{U}$ during training for each language pair. Table 2 shows the top 10 closest words in both source and target languages according to cosine similarity. Note that the model correctly identifies the translation in `en` as the top candidate, and the top 10 words in both source and target languages are highly related. This qualitative evaluation initially demonstrates the ability of our CLWE to capture both the bilingual and monolingual relationship.

Quantitative evaluation Table 3 shows our results compared with prior work. We reimple-

[5]Default learning rate of 0.025, negative sampling with 25 samples, subsampling rate of value $1e^{-4}$, embedding dimension $d = 200$, window size $cs = 48$ and run for 15 epochs.

gravedad$_{es}$		*tassazione*$_{it}$	
es	en	it	en
gravitacional	**gravity***	tasse	**taxation***
gravitatoria	gravitation*	fiscale	taxes
aceleracin	acceleration	tassa	tax*
gravitacin	non-gravitational	imposte	levied
inercia	inertia	imposta	fiscal
gravity	centrifugal	fiscali	low-tax
msugra	free-falling	l'imposta	revenue
centrífuga	gravitational	tonnage	levy
curvatura	free-fall	tax	annates
masa	newton	accise	evasion

Table 2: Top 10 closest words in both source and target language corresponding to es word *gravedad* (left) and it word *tassazione* (right). They have 15 and 4 dictionary translations respectively. The en words in the dictionary translations are marked with $(^*)$. The correct translation is in bold.

ment Gouws and Søgaard (2015) using Panlex and Wiktionary dictionaries. The result with Panlex is substantially worse than with Wiktionary. This confirms our hypothesis in §2. That is the context might be corrupted if we just randomly replace the training data with the translation from noisy dictionary such as Panlex.

Our model when randomly picking the translation is similar to Gouws and Søgaard (2015), using the Panlex dictionary. The biggest difference is that they replace the training data (both context and middle word) while we fix the context and only replace the middle word. For a high coverage yet noisy dictionary such as Panlex, our approach gives better average score. Comparing our two most basic models (EM selection and random selection), it is clear that the model using EM to select the translation outperforms random selection by a significant margin.

Our joint model, as described in equation (3) which predicts both target word and the translation, further improves the performance, especially for nl-en. We use equation (5) to combine both context embeddings $\mathbf{V}$ and word embeddings $\mathbf{U}$ for all three language pairs. This modification during training substantially improves the performance. More importantly, all our improvements are consistent for all three language pairs and both evaluation metrics, showing the robustness of our models.

Our combined model out-performed previous approaches by a large margin. Vulić and Moens (2015) used bilingual comparable data, but this might be hard to obtain for some language pairs. Their performance on nl-en is poor because their comparable data between en and nl is small. Besides, they also use POS tagger and lemmatizer to filter only *Noun* and reduce the morphology complexity during training. These tools might not be available for many languages. For a fairer comparison to their work, we also use the same Treetagger (Schmid, 1995) to lemmatize the output of our combined model before evaluation. Table 3 (+lemmatization) shows some improvements but minor. It demonstrates that our model is already good at disambiguating morphology. For example, the top 2 translations for es word *lenguas* in en are *languages* and *language* which correctly prefer the plural translation.

7 Monolingual Word Similarity

Now we consider the efficacy of our CLWE on monolingual word similarity. We evaluate on English monolingual similarity on WordSim353 (WS-en), RareWord (RW-en) and German version of WordSim353 (WS-de) (Finkelstein et al., 2001; Luong et al., 2013; Luong et al., 2015). Each of those datasets contain many tuples (w_1, w_2, s) where s is a scalar denoting the semantic similarity between w_1 and w_2 given by human annotators. Good system should produce the score correlated with human judgement.

We train the model as described in §4, which is the *combine embeddings* setting from Table 3. Since the evaluation involves de and en word similarity, we train the CLWE for en-de pair. Table 4 shows the performance of our combined model compared with several baselines. Our combined model out-performed both Luong et al. (2015) and Gouws and Søgaard (2015)[6] which represent the best published crosslingual embeddings trained on bitext and monolingual data respectively.

We also compare our system with the monolingual CBOW model trained on the monolingual data for each language, using the same parameter settings from earlier (§6). Surprisingly, our combined model performs better than the monolingual CBOW baseline which makes our result close to the monolingual state-of-the-art on each different dataset. However, the best monolingual methods use much larger

[6]trained using the Panlex dictionary

Model	es-en		it-en		nl-en		Average	
	rec_1	rec_5	rec_1	rec_5	rec_1	rec_5	rec_1	rec_5
Gouws and Søgaard (2015) + Panlex	37.6	63.6	26.6	56.3	49.8	76.0	38.0	65.3
Gouws and Søgaard (2015) + Wikt	61.6	78.9	62.6	81.1	65.6	79.7	63.3	79.9
BilBOWA: Gouws et al. (2015)	51.6	-	55.7	-	57.5	-	54.9	-
Vulić and Moens (2015)	68.9	-	68.3	-	39.2	-	58.8	-
Our model (random selection)	41.1	62.0	57.4	75.4	34.3	55.5	44.3	64.3
Our model (EM selection)	67.3	79.5	66.8	82.3	64.7	82.4	66.3	81.4
+ Joint model	68.0	80.5	70.5	83.3	68.8	84.0	69.1	82.6
+ combine embeddings ($\delta = 0.01$)	74.7	85.4	80.8	90.4	79.1	90.5	78.2	88.8
+ lemmatization	**74.9**	**86.0**	**81.3**	**91.3**	**79.8**	**91.3**	**78.7**	**89.5**

Table 3: Bilingual Lexicon Induction performance from es, it, nl to en. Gouws and Søgaard (2015) + Panlex/Wikt is our reimplementation using Panlex/Wiktionary dictionary. All our models use Panlex as the dictionary. We reported the recall at 1 and 5. The best performance is bold.

	Model	WS-de	WS-en	RW-en
Baselines	Klementiev et al. (2012)	23.8	13.2	7.3
	Chandar A P et al. (2014)	34.6	39.8	20.5
	Hermann and Blunsom (2014)	28.3	19.8	13.6
	Luong et al. (2015)	47.4	49.3	25.3
	Gouws and Søgaard (2015)	67.4	71.8	31.0
Mono	CBOW	62.2	70.3	42.7
	+ combine	65.8	74.1	43.1
	Yih and Qazvinian (2012)	-	81.0	-
	Shazeer et al. (2016)	-	74.8	48.3
Ours	Our joint-model	59.3	68.6	38.1
	+ combine	**71.1**	**76.2**	**44.0**

Table 4: Spearman's rank correlation for monolingual similarity measurement on 3 datasets WS-de (353 pairs), WS-en (353 pairs) and RW-en (2034 pairs). We compare against 5 baseline crosslingual word embeddings. The best CLWE performance is bold. For reference, we add the monolingual CBOW with and without embeddings combination, Yih and Qazvinian (2012) and Shazeer et al. (2016) which represents the monolingual state-of-the-art results for WS-en and RW-en.

monolingual corpora (Shazeer et al., 2016), WordNet or the output of commercial search engines (Yih and Qazvinian, 2012).

Next we explain the gain of our combined model compared with the monolingual CBOW model. First, we compare the combined model with the joint-model with respect to monolingual CBOW model (Table 4). It shows that the improvement seems mostly come from combining $\mathbf{V}$ and $\mathbf{U}$. If we apply the combining algorithm to the monolingual CBOW model (CBOW + combine), we also observe an improvement. Clearly most of the improvement is from combining $\mathbf{V}$ and $\mathbf{U}$, however our $\mathbf{V}$ and $\mathbf{U}$ are more complementary as the gain is more marked. Other improvements can be explained by the observation that a dictionary can improve monolingual accuracy through linking synonyms (Faruqui and Dyer, 2014). For example, since *plane*, *airplane* and *aircraft* have the same Italian translation *aereo*, the model will encourage those words to be closer in the embedding space.

8 Model selection

Combining context embeddings and word embeddings results in an improvement in both monolingual similarity and bilingual lexicon induction. In §4.3, we introduce several combination methods including post-processing (interpolation and concatenation) and during training (regularization). In this section, we justify our parameter and model choices.

We use en-it pair for tuning purposes, considering the value of γ in equation 4. Figure 2 shows the performances using different values of γ. The two extremes where $\gamma = 0$ and $\gamma = 1$ corresponds to no interpolation where we just use $\mathbf{U}$ or $\mathbf{V}$ respectively. As γ increases, the performance on WS-en increases yet BLI decreases. These results confirm our hypothesis in §4.3 that $\mathbf{U}$ is better at capturing bilingual relations and $\mathbf{V}$ is better at capturing monolingual relations. As a compromise, we choose $\gamma = 0.5$ in our experiments. Similarly, we tune the regularization sensitivity δ in equation (5) which combines embeddings space during training. We test $\delta = 10^{-n}$ with $n = \{0, 1, 2, 3, 4\}$ and us-

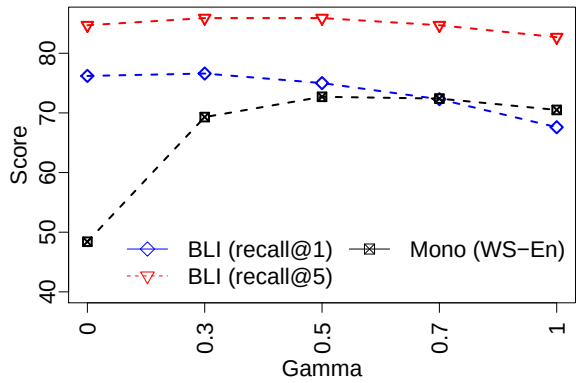


Figure 2: Performance of word embeddings interpolated using different values of γ evaluated using BLI (Recall@1, Recall@5) and English monolingual WordSim353 (WS-en).

	Model	BLI rec_1	BLI rec_5	Mono WS-en
Alone	Joint-model + **V**	67.6	82.8	70.5
	Joint-model + **U**	76.2	84.7	48.4
Combine	Interpolation $\left[\frac{\mathbf{V}+\mathbf{U}}{2}\right]$	75.0	85.9	72.7
	Concatenation	72.7	85.2	71.2
	Regularization + **V**	80.3	89.8	45.9
	Regularization + **U**	80.8	90.4	**74.8**
	Regularization + $\frac{\mathbf{V}+\mathbf{U}}{2}$	**80.9**	**91.1**	72.3

Table 5: Performance on en−it BLI and en monolingual similarity WordSim353 (WS-en) for various combining algorithms mentioned in §4.3 w.r.t just using **U** or **V** alone (after joint-training). We use $\gamma = 0.5$ for interpolation and $\delta = 0.01$ for regularization with the choice of **V**, **U** or interpolation of both $\frac{\mathbf{V}+\mathbf{U}}{2}$ for the output. The best scores are bold.

ing **V**, **U** or the interpolation of both $\frac{\mathbf{V}+\mathbf{U}}{2}$ as the learned embeddings, evaluated on the same BLI and WS-en. We select $\delta = 0.01$.

Table 5 shows the performance with and without using combining algorithms mentioned in §4.3. As the compromise between both monolingual and crosslingual tasks, we choose regularization + **U** as the combination algorithm. All in all, we apply the regularization algorithm for combining **V** and **U** with $\delta = 0.01$ and **U** as the output for all language pairs without further tuning.

9 Crosslingual Document Classification

In this section, we evaluate our CLWE on a downstream crosslingual document classification (CLDC) task. In this task, the document classifier is trained on a source language and then applied directly to classify a document in the target language. This is convenient for a target low-resource language where we do not have document annotations. The experimental setup is the same as Klementiev et al. (2012)[7] with the training and testing data sourced from Reuter RCV1/RCV2 corpus (Lewis et al., 2004).

Model	en → de	de → en
MT baseline	68.1	67.4
Klementiev et al. (2012)	77.6	71.1
Gouws et al. (2015)	86.5	75.0
Kočiský et al. (2014)	83.1	75.4
Chandar A P et al. (2014)	**91.8**	74.2
Hermann and Blunsom (2014)	86.4	74.7
Luong et al. (2015)	88.4	**80.3**
Our model	86.3	76.8

Table 6: CLDC performance for both en → de and de → en direction for many CLWE. The MT baseline uses phrase-based statistical machine translation to translate the source language to target language (Klementiev et al., 2012). The best scores are bold.

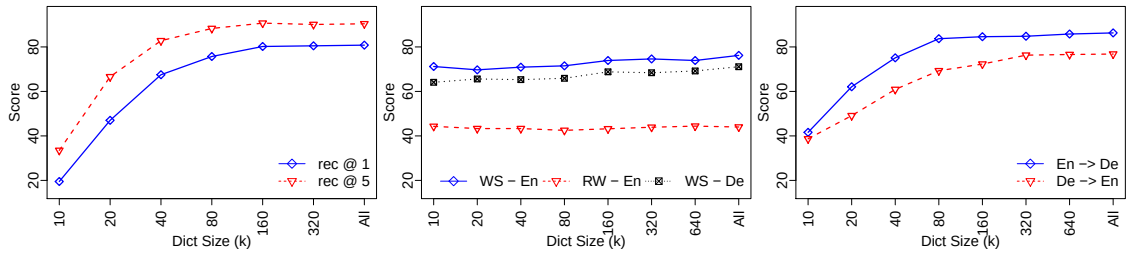
The documents are represented as the bag of word embeddings weighted by tf.idf. A multi-class classifier is trained using the average perceptron algorithm on 1000 documents in the source language and tested on 5000 documents in the target language. We use the CLWE, such that the document representation in the target language embeddings is in the same space with the source language.

We build the en-de CLWE using combined models as described in section §4. Following prior work, we also use monolingual data[8] from the RCV1/RCV2 corpus (Klementiev et al., 2012; Gouws et al., 2015; Chandar A P et al., 2014).

Table 6 shows the CLDC results for various CLWE. Despite its simplicity, our model achieves competitive performance. Note that aside from our model, all other models in Table 6 use a large bitext (Europarl) which may not exist for many low-resource languages, limiting their applicability.

[7]The data split and code are kindly provided by the authors.

[8]We randomly sample documents in RCV1 and RCV2 corpora and selected around 85k documents to form 400k monolingual sentences for both en and de. For each document, we perform basic pre-processing including: lower-casing, remove html tags and tokenization. These monolingual data are then concatenated with the monolingual data from Wikipedia to form the final training data.



(a) BLI (b) Mono Similarity (c) CLDC

Figure 3: Learning curve showing how task scores increase with increasing dictionary size; showing bilingual lexicon induction (BLI) task (left), monolingual similarity (center) and crosslingual document classification (right). BLI is trained on `en-it`, and monolingual similarity and CLDC are trained on `en-de`.

10 Low-resource languages

Our model exploits dictionaries, which are more widely available than parallel corpora. However the question remains as to how well this performs of a real low-resource language, rather than a simulated condition like above, whereupon the quality of the dictionary is likely to be worse. To test this, we evaluation on Serbian, a language with few annotated language resources. Table 1 shows the relative size of monolingual data and dictionary for `en-sr` compared with other language pairs. Both the Serbian monolingual data and the dictionary size is more than 10 times smaller than other language pairs. We build the `en-sr` CLWE using our best model (joint + combine) and evaluate on the bilingual word induction task using 939 gold translation pairs.[9] We achieved recall score of 35.8% and 45.5% at 1 and 5 respectively. Although worse than the earlier results, these numbers are still well above chance.

We can also simulate low-resource setting using our earlier datasets. For estimating the performance loss on all three tasks, we down sample the dictionary for `en-it` and `en-de` based on `en` word frequency. Figure 3 shows the performance with different dictionary sizes for all three tasks. The monolingual similarity performance is very similar across various sizes. For BLI and CLDC, dictionary size is more important, although performance levels off at around 80k dictionary pairs. We conclude that this size is sufficient for decent performance.

[9]The `sr`→`en` translations are sourced from Google Translate by translating one word at a time, followed by manually verification, after which 61 translation pairs were ruled out as being bad or questionable.

11 Conclusion

Previous CLWE methods often impose high resource requirements yet have low accuracy. We introduce a simple framework based on a large noisy dictionary. We model polysemy using EM translation selection during training to learn bilingual correspondences from monolingual corpora. Our algorithm allows to train on massive amount of monolingual data efficiently, representing monolingual and bilingual properties of language. This allows us to achieve state-of-the-art performance on bilingual lexicon induction task, competitive result on monolingual word similarity and crosslingual document classification task. Our combination techniques during training, especially using regularization, are highly effective and could be used to improve monolingual word embeddings.

Acknowledgments

This work was conducted during Duong's internship at IBM Research – Tokyo and partially supported by the University of Melbourne and National ICT Australia (NICTA). We are grateful for support from NSF Award 1464553 and the DARPA/I2O, Contract No. HR0011-15-C-0114. We thank Yuta Tsuboi and Alvin Grissom II for helpful discussions, Jan Šnajder for helping with `sr-en` evaluation.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 600–609.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China. Association for Computational Linguistics.

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

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414, New York, NY, USA. ACM.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado, May–June. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 748–756. JMLR Workshop and Conference Proceedings.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland, June. Association for Computational Linguistics.

David Kamholz, Jonathan Pool, and Susan Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3145–50, Reykjavik, Iceland. European Language Resources Association (ELRA).

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India, December. The COLING 2012 Organizing Committee.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–229, Baltimore, Maryland, June. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as a factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a.

Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Nat. Lang. Eng.*, 5(2):113–133, June.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving embeddings by noticing what's missing. *CoRR*, abs/1602.02215.

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

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1713–1722, Beijing, China, July. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 477–487. Association for Computational Linguistics.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725, Beijing, China, July. Association for Computational Linguistics.

Min Xiao and Yuhong Guo, 2014. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, chapter Distributed Word Representation Learning for Cross-Lingual Dependency Parsing, pages 119–129. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8, Pittsburgh, Pennsylvania.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 616–620, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sequence-to-Sequence Learning as Beam-Search Optimization

Sam Wiseman and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,srush}@seas.harvard.edu

Abstract

Sequence-to-Sequence (seq2seq) modeling has rapidly become an important general-purpose NLP tool that has proven effective for many text-generation and sequence-labeling tasks. Seq2seq builds on deep neural language modeling and inherits its remarkable accuracy in estimating local, next-word distributions. In this work, we introduce a model and beam-search training scheme, based on the work of Daumé III and Marcu (2005), that extends seq2seq to learn global sequence scores. This structured approach avoids classical biases associated with local training and unifies the training loss with the test-time usage, while preserving the proven model architecture of seq2seq and its efficient training approach. We show that our system outperforms a highly-optimized attention-based seq2seq system and other baselines on three different sequence to sequence tasks: word ordering, parsing, and machine translation.

1 Introduction

Sequence-to-Sequence learning with deep neural networks (herein, seq2seq) (Sutskever et al., 2011; Sutskever et al., 2014) has rapidly become a very useful and surprisingly general-purpose tool for natural language processing. In addition to demonstrating impressive results for machine translation (Bahdanau et al., 2015), roughly the same model and training have also proven to be useful for sentence compression (Filippova et al., 2015), parsing (Vinyals et al., 2015), and dialogue systems (Serban et al., 2016), and they additionally underlie other text generation applications, such as image or video captioning (Venugopalan et al., 2015; Xu et al., 2015).

The dominant approach to training a seq2seq system is as a conditional language model, with training maximizing the likelihood of each successive target word conditioned on the input sequence and the *gold* history of target words. Thus, training uses a strictly word-level loss, usually cross-entropy over the target vocabulary. This approach has proven to be very effective and efficient for training neural language models, and seq2seq models similarly obtain impressive perplexities for word-generation tasks.

Notably, however, seq2seq models are not used as conditional language models at test-time; they must instead generate fully-formed word sequences. In practice, generation is accomplished by searching over output sequences greedily or with beam search. In this context, Ranzato et al. (2016) note that the combination of the training and generation scheme just described leads to at least two major issues:

1. *Exposure Bias*: the model is never exposed to its own errors during training, and so the inferred histories at test-time do not resemble the gold training histories.

2. *Loss-Evaluation Mismatch*: training uses a word-level loss, while at test-time we target improving sequence-level evaluation metrics, such as BLEU (Papineni et al., 2002).

We might additionally add the concern of *label bias* (Lafferty et al., 2001) to the list, since word-probabilities at each time-step are locally normalized, guaranteeing that successors of incorrect his-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1296–1306,
Austin, Texas, November 1-5, 2016.

tories receive the same mass as do the successors of the true history.

In this work we develop a non-probabilistic variant of the seq2seq model that can assign a score to any possible target *sequence*, and we propose a training procedure, inspired by the learning as search optimization (LaSO) framework of Daumé III and Marcu (2005), that defines a loss function in terms of errors made during beam search. Furthermore, we provide an efficient algorithm to back-propagate through the beam-search procedure during seq2seq training.

This approach offers a possible solution to each of the three aforementioned issues, while largely maintaining the model architecture and training efficiency of standard seq2seq learning. Moreover, by scoring sequences rather than words, our approach also allows for enforcing hard-constraints on sequence generation *at training time*. To test out the effectiveness of the proposed approach, we develop a general-purpose seq2seq system with beam search optimization. We run experiments on three very different problems: word ordering, syntactic parsing, and machine translation, and compare to a highly-tuned seq2seq system with attention (Luong et al., 2015). The version with beam search optimization shows significant improvements on all three tasks, and particular improvements on tasks that require difficult search.

2 Related Work

The issues of exposure bias and label bias have received much attention from authors in the structured prediction community, and we briefly review some of this work here. One prominent approach to combating exposure bias is that of SEARN (Daumé III et al., 2009), a meta-training algorithm that learns a search policy in the form of a cost-sensitive classifier trained on examples generated from an interpolation of an oracle policy and the model's current (learned) policy. Thus, SEARN explicitly targets the mismatch between oracular training and non-oracular (often greedy) test-time inference by training on the output of the model's own policy. DAgger (Ross et al., 2011) is a similar approach, which differs in terms of how training examples are generated and aggregated, and there have additionally been important refinements to this style of training over the past several years (Chang et al., 2015). When it comes to training RNNs, SEARN/DAgger has been applied under the name "scheduled sampling" (Bengio et al., 2015), which involves training an RNN to generate the $t+1$'st token in a target sequence after consuming either the true t'th token, or, with probability that increases throughout training, the predicted t'th token.

Though technically possible, it is uncommon to use beam search when training with SEARN/DAgger. The early-update (Collins and Roark, 2004) and LaSO (Daumé III and Marcu, 2005) training strategies, however, explicitly account for beam search, and describe strategies for updating parameters when the gold structure becomes unreachable during search. Early update and LaSO differ primarily in that the former discards a training example after the first search error, whereas LaSO resumes searching after an error from a state that includes the gold partial structure. In the context of feed-forward neural network training, early update training has been recently explored in a feed-forward setting by Zhou et al. (2015) and Andor et al. (2016). Our work differs in that we adopt a LaSO-like paradigm (with some minor modifications), and apply it to the training of seq2seq RNNs (rather than feed-forward networks). We also note that Watanabe and Sumita (2015) apply maximum-violation training (Huang et al., 2012), which is similar to early-update, to a parsing model with recurrent components, and that Yazdani and Henderson (2015) use beam-search in training a discriminative, locally normalized dependency parser with recurrent components.

Recently authors have also proposed alleviating exposure bias using techniques from reinforcement learning. Ranzato et al. (2016) follow this approach to train RNN decoders in a seq2seq model, and they obtain consistent improvements in performance, even over models trained with scheduled sampling. As Daumé III and Marcu (2005) note, LaSO is similar to reinforcement learning, except it does not require "exploration" in the same way. Such exploration may be unnecessary in supervised text-generation, since we typically know the gold partial sequences at each time-step. Shen et al. (2016) use minimum risk training (approximated by

sampling) to address the issues of exposure bias and loss-evaluation mismatch for seq2seq MT, and show impressive performance gains.

Whereas exposure bias results from training in a certain way, label bias results from properties of the model itself. In particular, label bias is likely to affect structured models that make sub-structure predictions using locally-normalized scores. Because the neural and non-neural literature on this point has recently been reviewed by Andor et al. (2016), we simply note here that RNN models are typically locally normalized, and we are unaware of any specifically seq2seq work with RNNs that does *not* use locally-normalized scores. The model we introduce here, however, is not locally normalized, and so should not suffer from label bias. We also note that there are some (non-seq2seq) exceptions to the trend of locally normalized RNNs, such as the work of Sak et al. (2014) and Voigtlaender et al. (2015), who train LSTMs in the context of HMMs for speech recognition using sequence-level objectives; their work does not consider search, however.

3 Background and Notation

In the simplest seq2seq scenario, we are given a collection of source-target sequence pairs and tasked with learning to generate target sequences from source sequences. For instance, we might view machine translation in this way, where in particular we attempt to generate English sentences from (corresponding) French sentences. Seq2seq models are part of the broader class of "encoder-decoder" models (Cho et al., 2014), which first use an encoding model to transform a source object into an encoded representation $\boldsymbol{x}$. Many different sequential (and non-sequential) encoders have proven to be effective for different source domains. In this work we are agnostic to the form of the encoding model, and simply assume an abstract source representation $\boldsymbol{x}$.

Once the input sequence is encoded, seq2seq models generate a target sequence using a *decoder*. The decoder is tasked with generating a target sequence of words from a target vocabulary $\mathcal{V}$. In particular, words are generated sequentially by conditioning on the input representation $\boldsymbol{x}$ and on the previously generated words or *history*. We use the notation $w_{1:T}$ to refer to an arbitrary word sequence of length T, and the notation $y_{1:T}$ to refer to the *gold* (i.e., correct) target word sequence for an input $\boldsymbol{x}$.

Most seq2seq systems utilize a recurrent neural network (RNN) for the decoder model. Formally, a recurrent neural network is a parameterized nonlinear function $\mathbf{RNN}$ that recursively maps a sequence of vectors to a sequence of hidden states. Let $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T$ be a sequence of T vectors, and let $\boldsymbol{h}_0$ be some initial state vector. Applying an RNN to any such sequence yields hidden states $\boldsymbol{h}_t$ at each time-step t, as follows:

$$\boldsymbol{h}_t \leftarrow \mathbf{RNN}(\boldsymbol{m}_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ is the set of model parameters, which are shared over time. In this work, the vectors $\boldsymbol{m}_t$ will always correspond to the embeddings of a target word sequence $w_{1:T}$, and so we will also write $\boldsymbol{h}_t \leftarrow \mathbf{RNN}(w_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta})$, with w_t standing in for its embedding.

RNN decoders are typically trained to act as conditional language models. That is, one attempts to model the probability of the t'th target word conditioned on $\boldsymbol{x}$ and the target history by stipulating that $p(w_t|w_{1:t-1}, \boldsymbol{x}) = g(w_t, \boldsymbol{h}_{t-1}, \boldsymbol{x})$, for some parameterized function g typically computed with an affine layer followed by a softmax. In computing these probabilities, the state $\boldsymbol{h}_{t-1}$ represents the target history, and $\boldsymbol{h}_0$ is typically set to be some function of $\boldsymbol{x}$. The complete model (including encoder) is trained, analogously to a neural language model, to minimize the cross-entropy loss at each time-step while conditioning on the gold history in the training data. That is, the model is trained to minimize $-\ln \prod_{t=1}^{T} p(y_t|y_{1:t-1}, \boldsymbol{x})$.

Once the decoder is trained, discrete sequence generation can be performed by approximately maximizing the probability of the target sequence under the conditional distribution, $\hat{y}_{1:T} = \mathrm{argbeam}_{w_{1:T}} \prod_{t=1}^{T} p(w_t|w_{1:t-1}, \boldsymbol{x})$, where we use the notation $\mathrm{argbeam}$ to emphasize that the decoding process requires heuristic search, since the RNN model is non-Markovian. In practice, a simple beam search procedure that explores K prospective histories at each time-step has proven to be an effective decoding approach. However, as noted above, decoding in this manner after conditional language-model style training *potentially* suffers from the is-

sues of exposure bias and label bias, which motivates the work of this paper.

4 Beam Search Optimization

We begin by making one small change to the seq2seq modeling framework. Instead of predicting the probability of the next word, we instead learn to produce (non-probabilistic) scores for ranking sequences. Define the score of a sequence consisting of *history* $w_{1:t-1}$ followed by a single word w_t as $f(w_t, \boldsymbol{h}_{t-1}, \boldsymbol{x})$, where f is a parameterized function examining the current hidden-state of the relevant RNN at time $t-1$ as well as the input representation $\boldsymbol{x}$. In experiments, our f will have an identical form to g but *without* the final softmax transformation (which transforms unnormalized scores into probabilities), thereby allowing the model to avoid issues associated with the label bias problem.

More importantly, we also modify how this model is trained. Ideally we would train by comparing the gold sequence to the highest-scoring complete sequence. However, because finding the argmax sequence according to this model is intractable, we propose to adopt a LaSO-like (Daumé III and Marcu, 2005) scheme to train, which we will refer to as beam search optimization (BSO). In particular, we define a loss that penalizes the gold sequence falling off the beam during training.[1] The proposed training approach is a simple way to expose the model to incorrect histories and to match the training procedure to test generation. Furthermore we show that it can be implemented efficiently without changing the asymptotic run-time of training, beyond a factor of the beam size K.

4.1 Search-Based Loss

We now formalize this notion of a search-based loss for RNN training. Assume we have a set S_t of K candidate sequences of length t. We can calculate a score for each sequence in S_t using a scoring function f parameterized with an RNN, as above, and we define the sequence $\hat{y}^{(K)}_{1:t} \in S_t$ to be the K'th ranked sequence in S_t according to f. That is, assuming distinct scores,

$$|\{\hat{y}^{(k)}_{1:t} \in S_t \mid f(\hat{y}^{(k)}_t, \hat{\boldsymbol{h}}^{(k)}_{t-1}) > f(\hat{y}^{(K)}_t, \hat{\boldsymbol{h}}^{(K)}_{t-1})\}| = K - 1,$$

where $\hat{y}^{(k)}_t$ is the t'th token in $\hat{y}^{(k)}_{1:t}$, $\hat{\boldsymbol{h}}^{(k)}_{t-1}$ is the RNN state corresponding to its $t-1$'st step, and where we have omitted the $\boldsymbol{x}$ argument to f for brevity.

We now define a loss function that gives loss each time the score of the gold prefix $y_{1:t}$ does not exceed that of $\hat{y}^{(K)}_{1:t}$ by a margin:

$$\mathcal{L}(f) = \sum_{t=1}^{T} \Delta(\hat{y}^{(K)}_{1:t}) \left[1 - f(y_t, \boldsymbol{h}_{t-1}) + f(\hat{y}^{(K)}_t, \hat{\boldsymbol{h}}^{(K)}_{t-1})\right].$$

Above, the $\Delta(\hat{y}^{(K)}_{1:t})$ term denotes a mistake-specific cost-function, which allows us to scale the loss depending on the severity of erroneously predicting $\hat{y}^{(K)}_{1:t}$; it is assumed to return 0 when the margin requirement is satisfied, and a positive number otherwise. It is this term that allows us to use sequence- rather than word-level costs in training (addressing the 2nd issue in the introduction). For instance, when training a seq2seq model for machine translation, it may be desirable to have $\Delta(\hat{y}^{(K)}_{1:t})$ be inversely related to the partial sentence-level BLEU score of $\hat{y}^{(K)}_{1:t}$ with $y_{1:t}$; we experiment along these lines in Section 5.3.

Finally, because we want the full gold sequence to be at the top of the beam at the end of search, when $t = T$ we modify the loss to require the score of $y_{1:T}$ to exceed the score of the *highest* ranked incorrect prediction by a margin.

We can optimize the loss $\mathcal{L}$ using a two-step process: (1) in a forward pass, we compute candidate sets S_t and record margin violations (sequences with non-zero loss); (2) in a backward pass, we back-propagate the errors through the seq2seq RNNs. Unlike standard seq2seq training, the first-step requires running search (in our case beam search) to find margin violations. The second step can be done by adapting back-propagation through time (BPTT). We next discuss the details of this process.

4.2 Forward: Find Violations

In order to minimize this loss, we need to specify a procedure for constructing candidate sequences $\hat{y}^{(k)}_{1:t}$

[1] Using a non-probabilistic model further allows us to incur no loss (and thus require no update to parameters) when the gold sequence *is* on the beam; this contrasts with models based on a CRF loss, such as those of Andor et al. (2016) and Zhou et al. (2015), though in training those models are simply not updated when the gold sequence remains on the beam.

at each time step t so that we find margin violations. We follow LaSO (rather than early-update [2]; see Section 2) and build candidates in a recursive manner. If there was no margin violation at $t-1$, then S_t is constructed using a standard beam search update. If there was a margin violation, S_t is constructed as the K best sequences assuming the gold history $y_{1:t-1}$ through time-step $t-1$.

Formally, assume the function succ maps a sequence $w_{1:t-1} \in \mathcal{V}^{t-1}$ to the set of all valid sequences of length t that can be formed by appending to it a valid word $w \in \mathcal{V}$. In the simplest, unconstrained case, we will have

$$\text{succ}(w_{1:t-1}) = \{w_{1:t-1}, w \mid w \in \mathcal{V}\}.$$

As an important aside, note that for some problems it may be preferable to define a succ function which imposes hard constraints on successor sequences. For instance, if we would like to use seq2seq models for parsing (by emitting a constituency or dependency structure encoded into a sequence in some way), we will have hard constraints on the sequences the model can output, namely, that they represent valid parses. While hard constraints such as these would be difficult to add to standard seq2seq at training time, in our framework they can naturally be added to the succ function, allowing us to *train* with hard constraints; we experiment along these lines in Section 5.3, where we refer to a model trained with constrained beam search as ConBSO.

Having defined an appropriate succ function, we specify the candidate set as:

$$S_t = \text{topK} \begin{cases} \text{succ}(y_{1:t-1}) & \text{violation at } t-1 \\ \bigcup_{k=1}^{K} \text{succ}(\hat{y}_{1:t-1}^{(k)}) & \text{otherwise,} \end{cases}$$

where we have a margin violation at $t-1$ iff $f(y_{t-1}, \boldsymbol{h}_{t-2}) < f(\hat{y}_{t-1}^{(K)}, \hat{\boldsymbol{h}}_{t-2}^{(K)}) + 1$, and where topK considers the scores given by f. This search procedure is illustrated in the top portion of Figure 1.

In the forward pass of our training algorithm, shown as the first part of Algorithm 1, we run this version of beam search and collect all sequences and their hidden states that lead to losses.

[2] We found that training with early-update rather than (delayed) LaSO did not work well, even after pre-training. Given the success of early-update in many NLP tasks this was somewhat surprising. We leave this question to future work.

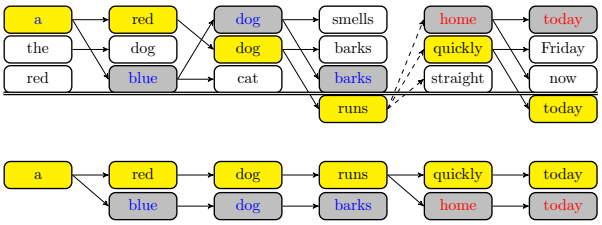


Figure 1: Top: possible $\hat{y}_{1:t}^{(k)}$ formed in training with a beam of size $K=3$ and with gold sequence $y_{1:6}$ = "a red dog runs quickly today". The gold sequence is highlighted in yellow, and the predicted prefixes involved in margin violations (at $t=4$ and $t=6$) are in gray. Note that time-step $T=6$ uses a different loss criterion. Bottom: prefixes that actually participate in the loss, arranged to illustrate the back-propagation process.

4.3 Backward: Merge Sequences

Once we have collected margin violations we can run backpropagation to compute parameter updates. Assume a margin violation occurs at time-step t between the predicted history $\hat{y}_{1:t}^{(K)}$ and the gold history $y_{1:t}$. As in standard seq2seq training we must back-propagate this error through the gold history; however, unlike seq2seq we also have a gradient for the wrongly predicted history.

Recall that to back-propagate errors through an RNN we run a recursive backward procedure — denoted below by **BRNN** — at each time-step t, which accumulates the gradients of next-step and future losses with respect to $\boldsymbol{h}_t$. We have:

$$\nabla_{\boldsymbol{h}_t}\mathcal{L} \leftarrow \mathbf{BRNN}(\nabla_{\boldsymbol{h}_t}\mathcal{L}_{t+1}, \nabla_{\boldsymbol{h}_{t+1}}\mathcal{L}),$$

where $\mathcal{L}_{t+1}$ is the loss at step $t+1$, deriving, for instance, from the score $f(y_{t+1}, \boldsymbol{h}_t)$. Running this **BRNN** procedure from $t=T-1$ to $t=0$ is known as back-propagation through time (BPTT).

In determining the total computational cost of back-propagation here, first note that in the worst case there is one violation at each time-step, which leads to T independent, incorrect sequences. Since we need to call **BRNN** $O(T)$ times for each sequence, a naive strategy of running BPTT for each incorrect sequence would lead to an $O(T^2)$ backward pass, rather than the $O(T)$ time required for the standard seq2seq approach.

Fortunately, our combination of search-strategy and loss make it possible to efficiently share **BRNN** operations. This shared structure comes

naturally from the LaSO update, which resets the beam in a convenient way.

We informally illustrate the process in Figure 1. The top of the diagram shows a possible sequence of $\hat{y}^{(k)}_{1:t}$ formed during search with a beam of size 3 for the target sequence $y =$ "a red dog runs quickly today." When the gold sequence falls off the beam at $t = 4$, search resumes with $S_5 = \text{succ}(y_{1:4})$, and so all subsequent predicted sequences have $y_{1:4}$ as a prefix and are thus functions of $\boldsymbol{h}_4$. Moreover, because our loss function only involves the scores of the gold prefix and the violating prefix, we end up with the relatively simple computation tree shown at the bottom of Figure 1. It is evident that we can backpropagate in a single pass, accumulating gradients from sequences that diverge from the gold at the time-step that precedes their divergence. The second half of Algorithm 1 shows this explicitly for a single sequence, though it is straightforward to extend the algorithm to operate in batch.[3]

Algorithm 1 Seq2seq Beam-Search Optimization

procedure BSO($\boldsymbol{x}, K_{tr}, \text{succ}$)
  /*FORWARD*/
  Init empty storage $\hat{y}_{1:T}$ and $\hat{\boldsymbol{h}}_{1:T}$; init S_1
  $r \leftarrow 0$; $violations \leftarrow \{0\}$
  for $t = 1, \ldots, T$ **do**
    $K = K_{tr}$ if $t \neq T$ else $\underset{k:\hat{y}^{(k)}_{1:t} \neq y_{1:t}}{\arg\max}\ f(\hat{y}^{(k)}_t, \hat{\boldsymbol{h}}^{(k)}_{t-1})$
    if $f(y_t, \boldsymbol{h}_{t-1}) < f(\hat{y}^{(K)}_t, \hat{\boldsymbol{h}}^{(K)}_{t-1}) + 1$ **then**
      $\hat{\boldsymbol{h}}_{r:t-1} \leftarrow \hat{\boldsymbol{h}}^{(K)}_{r:t-1}$
      $\hat{y}_{r+1:t} \leftarrow \hat{y}^{(K)}_{r+1:t}$
      Add t to $violations$
      $r \leftarrow t$
      $S_{t+1} \leftarrow \text{topK}(\text{succ}(y_{1:t}))$
    else
      $S_{t+1} \leftarrow \text{topK}(\bigcup_{k=1}^{K} \text{succ}(\hat{y}^{(k)}_{1:t}))$
  /*BACKWARD*/
  $grad\_\boldsymbol{h}_T \leftarrow \mathbf{0}$; $grad\_\widehat{\boldsymbol{h}}_T \leftarrow \mathbf{0}$
  for $t = T - 1, \ldots, 1$ **do**
    $grad\_\boldsymbol{h}_t \leftarrow \mathbf{BRNN}(\nabla_{\boldsymbol{h}_t} \mathcal{L}_{t+1}, grad\_\boldsymbol{h}_{t+1})$
    $grad\_\widehat{\boldsymbol{h}}_t \leftarrow \mathbf{BRNN}(\nabla_{\widehat{\boldsymbol{h}}_t} \mathcal{L}_{t+1}, grad\_\widehat{\boldsymbol{h}}_{t+1})$
    if $t - 1 \in violations$ **then**
      $grad\_\boldsymbol{h}_t \leftarrow grad\_\boldsymbol{h}_t + grad\_\widehat{\boldsymbol{h}}_t$
      $grad\_\widehat{\boldsymbol{h}}_t \leftarrow \mathbf{0}$

5 Data and Methods

We run experiments on three different tasks, comparing our approach to the seq2seq baseline, and to other relevant baselines.

5.1 Model

While the method we describe applies to seq2seq RNNs in general, for all experiments we use the global attention model of Luong et al. (2015) — which consists of an LSTM (Hochreiter and Schmidhuber, 1997) encoder and an LSTM decoder with a global attention model — as both the baseline seq2seq model (i.e., as the model that computes the g in Section 3) and as the model that computes our sequence-scores $f(w_t, \boldsymbol{h}_{t-1}, \boldsymbol{x})$. As in Luong et al. (2015), we also use "input feeding," which involves feeding the attention distribution from the previous time-step into the decoder at the current step. This model architecture has been found to be highly performant for neural machine translation and other seq2seq tasks.

To distinguish the models we refer to our system as BSO (beam search optimization) and to the baseline as seq2seq. When we apply constrained training (as discussed in Section 4.2), we refer to the model as ConBSO. In providing results we also distinguish between the beam size K_{tr} with which the model is trained, and the beam size K_{te} which is used at test-time. In general, if we plan on evaluating with a beam of size K_{te} it makes sense to train with a beam of size $K_{tr} = K_{te} + 1$, since our objective requires the gold sequence to be scored higher than the *last* sequence on the beam.

5.2 Methodology

Here we detail additional techniques we found necessary to ensure the model learned effectively. First, we found that the model failed to learn when trained from a random initialization.[4] We therefore found it necessary to pre-train the model using a standard, word-level cross-entropy loss as described in Sec-

[3] We also note that because we do not update the parameters until after the T'th search step, our training procedure differs slightly from LaSO (which is online), and in this aspect is essentially equivalent to the "delayed LaSO update" of Björkelund and Kuhn (2014).

[4] This may be because there is relatively little signal in the sparse, sequence-level gradient, but this point requires further investigation.

tion 3. The necessity of pre-training in this instance is consistent with the findings of other authors who train non-local neural models (Kingsbury, 2009; Sak et al., 2014; Andor et al., 2016; Ranzato et al., 2016).[5]

Similarly, it is clear that the smaller the beam used in training is, the less room the model has to make erroneous predictions without running afoul of the margin loss. Accordingly, we also found it useful to use a "curriculum beam" strategy in training, whereby the size of the beam is increased gradually during training. In particular, given a desired training beam size K_{tr}, we began training with a beam of size 2, and increased it by 1 every 2 epochs until reaching K_{tr}.

Finally, it has been established that *dropout* (Srivastava et al., 2014) regularization improves the performance of LSTMs (Pham et al., 2014; Zaremba et al., 2014), and in our experiments we run beam search under dropout.[6]

For all experiments, we trained both seq2seq and BSO models with mini-batch Adagrad (Duchi et al., 2011) (using batches of size 64), and we renormalized all gradients so they did not exceed 5 before updating parameters. We did not extensively tune learning-rates, but we found initial rates of 0.02 for the encoder and decoder LSTMs, and a rate of 0.1 or 0.2 for the final linear layer (i.e., the layer tasked with making word-predictions at each time-step) to work well across all the tasks we considered. Code implementing the experiments described below can be found at `https://github.com/harvardnlp/BSO`.[7]

5.3 Tasks and Results

Our experiments are primarily intended to evaluate the effectiveness of beam search optimization over standard seq2seq training. As such, we run experiments with the same model across three very different problems: word ordering, dependency parsing, and machine translation. While we do not include all the features and extensions necessary to reach state-of-the-art performance, even the baseline seq2seq model is generally quite performant.

Word Ordering The task of correctly ordering the words in a shuffled sentence has recently gained some attention as a way to test the (syntactic) capabilities of text-generation systems (Zhang and Clark, 2011; Zhang and Clark, 2015; Liu et al., 2015; Schmaltz et al., 2016). We cast this task as seq2seq problem by viewing a shuffled sentence as a source sentence, and the correctly ordered sentence as the target. While word ordering is a somewhat synthetic task, it has two interesting properties for our purposes. First, it is a task which plausibly requires search (due to the exponentially many possible orderings), and, second, there is a clear hard constraint on output sequences, namely, that they be a permutation of the source sequence. For both the baseline and BSO models we enforce this constraint at test-time. However, we also experiment with constraining the BSO model during training, as described in Section 4.2, by defining the succ function to only allow successor sequences containing un-used words in the source sentence.

For experiments, we use the same PTB dataset (with the standard training, development, and test splits) and evaluation procedure as in Zhang and Clark (2015) and later work, with performance reported in terms of BLEU score with the correctly ordered sentences. For all word-ordering experiments we use 2-layer encoder and decoder LSTMs, each with 256 hidden units, and dropout with a rate of 0.2 between LSTM layers. We use simple 0/1 costs in defining the Δ function.

We show our test-set results in Table 1. We see that on this task there is a large improvement at each beam size from switching to BSO, and a further improvement from using the constrained model.

Inspired by a similar analysis in Daumé III and Marcu (2005), we further examine the relationship between K_{tr} and K_{te} when training with ConBSO in Table 2. We see that larger K_{tr} hurt greedy inference, but that results continue to improve, at least initially, when using a K_{te} that is (somewhat) bigger than $K_{tr} - 1$.

[5] Andor et al. (2016) found, however, that pre-training only increased convergence-speed, but was not necessary for obtaining good results.

[6] However, it is important to ensure that the same mask applied at each time-step of the forward search is also applied at the corresponding step of the backward pass. We accomplish this by pre-computing masks for each time-step, and sharing them between the partial sequence LSTMs.

[7] Our code is based on Yoon Kim's seq2seq code, `https://github.com/harvardnlp/seq2seq-attn`.

	Word Ordering (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	**28.6**	**34.3**	**34.5**
LSTM-LM	15.4	-	26.8

Table 1: Word ordering. BLEU Scores of seq2seq, BSO, constrained BSO, and a vanilla LSTM language model (from Schmaltz et al, 2016). All experiments above have $K_{tr} = 6$.

	Word Ordering Beam Size (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
K_{tr} = 2	30.59	31.23	30.26
K_{tr} = 6	28.20	34.22	34.67
K_{tr} = 11	26.88	34.42	34.88
seq2seq	26.11	30.20	31.04

Table 2: Beam-size experiments on word ordering development set. All numbers reflect training with constraints (ConBSO).

Dependency Parsing We next apply our model to dependency parsing, which also has hard constraints and plausibly benefits from search. We treat dependency parsing with arc-standard transitions as a seq2seq task by attempting to map from a source sentence to a target sequence of source sentence words interleaved with the arc-standard, reduce-actions in its parse. For example, we attempt to map the source sentence

> But it was the Quotron problems that ...

to the target sequence

> But it was @L_SBJ @L_DEP the Quotron problems @L_NMOD @L_NMOD that ...

We use the standard Penn Treebank dataset splits with Stanford dependency labels, and the standard UAS/LAS evaluation metric (excluding punctuation) following Chen and Manning (2014). All models thus see only the words in the source and, when decoding, the actions it has emitted so far; no other features are used. We use 2-layer encoder and decoder LSTMs with 300 hidden units per layer and dropout with a rate of 0.3 between LSTM layers. We replace singleton words in the training set with an UNK token, normalize digits to a single symbol, and initialize word embeddings for both source and target words from the publicly available `word2vec` (Mikolov et al., 2013) embeddings. We use simple 0/1 costs in defining the Δ function.

	Dependency Parsing (UAS/LAS)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	**87.33/82.26**	88.53/84.16	88.66/84.33
BSO	86.91/82.11	91.00/**87.18**	91.17/**87.41**
ConBSO	85.11/79.32	**91.25**/86.92	**91.57**/87.26
Andor	93.17/91.18	-	-

Table 3: Dependency parsing. UAS/LAS of seq2seq, BSO, ConBSO and baselines on PTB test set. Andor is the current state-of-the-art model for this data set (Andor et al. 2016), and we note that with a beam of size 32 they obtain 94.41/92.55. All experiments above have $K_{tr} = 6$.

As in the word-ordering case, we also experiment with modifying the succ function in order to train under hard constraints, namely, that the emitted target sequence be a valid parse. In particular, we constrain the output at each time-step to obey the stack constraint, and we ensure words in the source are emitted in order.

We show results on the test-set in Table 3. BSO and ConBSO both show significant improvements over seq2seq, with ConBSO improving most on UAS, and BSO improving most on LAS. We achieve a reasonable final score of 91.57 UAS, which lags behind the state-of-the-art, but is promising for a general-purpose, word-only model.

Translation We finally evaluate our model on a small machine translation dataset, which allows us to experiment with a cost function that is not 0/1, and to consider other baselines that attempt to mitigate exposure bias in the seq2seq setting. We use the dataset from the work of Ranzato et al. (2016), which uses data from the German-to-English portion of the IWSLT 2014 machine translation evaluation campaign (Cettolo et al., 2014). The data comes from translated TED talks, and the dataset contains roughly 153K training sentences, 7K development sentences, and 7K test sentences. We use the same preprocessing and dataset splits as Ranzato et

	Machine Translation (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	22.53	24.03	23.87
BSO, SB-Δ	**23.83**	**26.36**	**25.48**
XENT	17.74	20.10	20.28
DAD	20.12	22.25	22.40
MIXER	20.73	21.81	21.83

Table 4: Machine translation experiments on test set; results below middle line are from MIXER model of Ranzato et al. (2016). SB-Δ indicates sentence BLEU costs are used in defining Δ. XENT is similar to our seq2seq model but with a convolutional encoder and simpler attention. DAD trains seq2seq with scheduled sampling (Bengio et al., 2015). BSO, SB-Δ experiments above have $K_{tr} = 6$.

al. (2016), and like them we also use a single-layer LSTM decoder with 256 units. We also use dropout with a rate of 0.2 between each LSTM layer. We emphasize, however, that while our decoder LSTM is of the same size as that of Ranzato et al. (2016), our results are not directly comparable, because we use an LSTM encoder (rather than a convolutional encoder as they do), a slightly different attention mechanism, and input feeding (Luong et al., 2015).

For our main MT results, we set $\Delta(\hat{y}^{(k)}_{1:t})$ to $1 - \mathrm{SB}(\hat{y}^{(K)}_{r+1:t}, y_{r+1:t})$, where r is the last margin violation and SB denotes smoothed, sentence-level BLEU (Chen and Cherry, 2014). This setting of Δ should act to penalize erroneous predictions with a relatively low sentence-level BLEU score more than those with a relatively high sentence-level BLEU score. In Table 4 we show our final results and those from Ranzato et al. (2016).[8] While we start with an improved baseline, we see similarly large increases in accuracy as those obtained by DAD and MIXER, in particular when $K_{te} > 1$.

We further investigate the utility of these sequence-level costs in Table 5, which compares using sentence-level BLEU costs in defining Δ with using 0/1 costs. We see that the more sophisticated sequence-level costs have a moderate effect on BLEU score.

[8] Some results from personal communication.

	Machine Translation (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
0/1-Δ	25.73	28.21	27.43
SB-Δ	25.99	28.45	27.58

Table 5: BLEU scores obtained on the machine translation development data when training with $\Delta(\hat{y}^{(k)}_{1:t}) = 1$ (top) and $\Delta(\hat{y}^{(k)}_{1:t}) = 1 - \mathrm{SB}(\hat{y}^{(K)}_{r+1:t}, y_{r+1:t})$ (bottom), and K_{tr} = 6.

Timing Given Algorithm 1, we would expect training time to increase linearly with the size of the beam. On the above MT task, our highly tuned seq2seq baseline processes an average of 13,038 tokens/second (including both source and target tokens) on a GTX 970 GPU. For beams of size K_{tr} = 2, 3, 4, 5, and 6, our implementation processes on average 1,985, 1,768, 1,709, 1,521, and 1,458 tokens/second, respectively. Thus, we appear to pay an initial constant factor of ≈ 3.3 due to the more complicated forward and backward passes, and then training scales with the size of the beam. Because we batch beam predictions on a GPU, however, we find that in practice training time scales sub-linearly with the beam-size.

6 Conclusion

We have introduced a variant of seq2seq and an associated beam search training scheme, which addresses exposure bias as well as label bias, and moreover allows for both training with sequence-level cost functions as well as with hard constraints. Future work will examine scaling this approach to much larger datasets.

Acknowledgments

We thank Yoon Kim for helpful discussions and for providing the initial seq2seq code on which our implementations are based. We thank Allen Schmaltz for help with the word ordering experiments. We also gratefully acknowledge the support of a Google Research Award.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav

Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference Resolution with Latent Antecedents and Non-local Features. *ACL, Baltimore, MD, USA, June*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign. In *Proceedings of IWSLT, 20014*.

Kai-Wei Chang, Hal Daumé III, John Langford, and Stephane Ross. 2015. Efficient programmable learning to search. In *Arxiv*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. *ACL 2014*, page 362.

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

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML 2005)*, pages 169–176.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Brian Kingsbury. 2009. Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3761–3764. IEEE.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of NAACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1412–1421.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635.

Hasim Sak, Oriol Vinyals, Georg Heigold, Andrew W. Senior, Erik McDermott, Rajat Monga, and Mark Z.

Mao. 2014. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *INTERSPEECH 2014*, pages 1209–1213.

Allen Schmaltz, Alexander M Rush, and Stuart M Shieber. 2016. Word ordering without syntax. *arXiv preprint arXiv:1604.08633*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3784.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *ICCV*, pages 4534–4542.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Paul Voigtlaender, Patrick Doetsch, Simon Wiesler, Ralf Schluter, and Hermann Ney. 2015. Sequence-discriminative training of recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2100–2104. IEEE.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. *Proceedings of ACL-IJCNLP*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proceedings of the 19th Conference on Computational Natural Language Learning, (CoNLL 2015)*, pages 142–152.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222.

Online Segment to Segment Neural Transduction

Lei Yu[1], **Jan Buys**[1] and **Phil Blunsom**[1,2]
[1]University of Oxford
[2]DeepMind
{lei.yu, jan.buys, phil.blunsom}@cs.ox.ac.uk

Abstract

We introduce an online neural sequence to sequence model that learns to alternate between encoding and decoding segments of the input as it is read. By independently tracking the encoding and decoding representations our algorithm permits exact polynomial marginalization of the latent segmentation during training, and during decoding beam search is employed to find the best alignment path together with the predicted output sequence. Our model tackles the bottleneck of vanilla encoder-decoders that have to read and memorize the entire input sequence in their fixed-length hidden states before producing any output. It is different from previous attentive models in that, instead of treating the attention weights as output of a deterministic function, our model assigns attention weights to a sequential latent variable which can be marginalized out and permits online generation. Experiments on abstractive sentence summarization and morphological inflection show significant performance gains over the baseline encoder-decoders.

1 Introduction

The problem of mapping from one sequence to another is an importance challenge of natural language processing. Common applications include machine translation and abstractive sentence summarisation. Traditionally this type of problem has been tackled by a combination of hand-crafted features, alignment models, segmentation heuristics, and language models, all of which are tuned separately.

The recently introduced encoder-decoder paradigm has proved very successful for machine translation, where an input sequence is encoded into a fixed-length vector and an output sequence is then decoded from said vector (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). This architecture is appealing, as it makes it possible to tackle the problem of sequence-to-sequence mapping by training a large neural network in an end-to-end fashion. However it is difficult for a fixed-length vector to memorize all the necessary information of an input sequence, especially for long sequences. Often a very large encoding needs to be employed in order to capture the longest sequences, which invariably wastes capacity and computation for short sequences. While the attention mechanism of Bahdanau et al. (2015) goes some way to address this issue, it still requires the full input to be seen before any output can be produced.

In this paper we propose an architecture to tackle the limitations of the vanilla encoder-decoder model, a segment to segment neural transduction model (SSNT) that learns to generate and align simultaneously. Our model is inspired by the HMM word alignment model proposed for statistical machine translation (Vogel et al., 1996; Tillmann et al., 1997); we impose a monotone restriction on the alignments but incorporate recurrent dependencies on the input which enable rich locally non-monotone alignments to be captured. This is similar to the sequence transduction model of Graves (2012), but we propose alignment distributions which are parameterised separately, making the model more flexible

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1307–1316,
Austin, Texas, November 1-5, 2016.

and allowing online inference.

Our model introduces a latent segmentation which determines correspondences between tokens of the input sequence and those of the output sequence. The aligned hidden states of the encoder and decoder are used to predict the next output token and to calculate the transition probability of the alignment. We carefully design the input and output RNNs such that they independently update their respective hidden states. This enables us to derive an exact dynamic programme to marginalize out the hidden segmentation during training and an efficient beam search to generate online the best alignment path together with the output sequence during decoding. Unlike previous recurrent segmentation models that only capture dependencies in the input (Graves et al., 2006; Kong et al., 2016), our segmentation model is able to capture unbounded dependencies in both the input and output sequences while still permitting polynomial inference.

While attentive models treat the attention weights as output of a deterministic function, our model assigns attention weights to a sequential latent variable which can be marginalized out. Our model is general and could be incorporated into any RNN-based encoder-decoder architecture, such as Neural Turing Machines (Graves et al., 2014), memory networks (Weston et al., 2015; Kumar et al., 2016) or stack-based networks (Grefenstette et al., 2015), enabling such models to process data online.

We conduct experiments on two different transduction tasks, abstractive sentence summarisation (sequence to sequence mapping at word level) and morphological inflection generation (sequence to sequence mapping at character level). We evaluate our proposed algorithms in both the online setting, where the input is encoded with a unidirectional LSTM, and where the whole input is available such that it can be encoded with a bidirectional network. The experimental results demonstrate the effectiveness of SSNT — it consistently output performs the baseline encoder-decoder approach while requiring significantly smaller hidden layers, thus showing that the segmentation model is able to learn to break one large transduction task into a series of smaller encodings and decodings. When bidirectional encodings are used the segmentation model outperforms an attention-based benchmark. Qualitative analysis shows that the alignments found by our model are highly intuitive and demonstrates that the model learns to read ahead the required number of tokens before producing output.

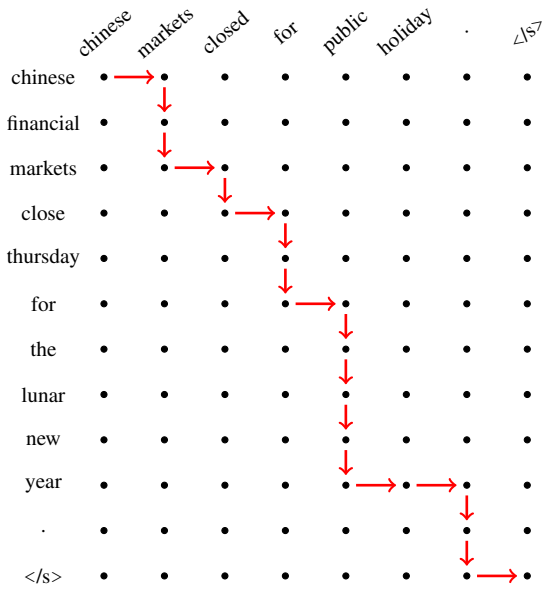


Figure 1: Example output of our recurrent segmentation model on the task of abstractive sentence summarisation. The path highlighted is the alignment found by the model during decoding.

2 Model

Let $\boldsymbol{x}_1^I$ be the input sequence of length I and $\boldsymbol{y}_1^J$ the output sequence of length J. Let y_j denote the j-th token of $\boldsymbol{y}$. Our goal is to model the conditional distribution

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{J} p(y_j|\boldsymbol{y}_1^{j-1}, \boldsymbol{x}). \quad (1)$$

We introduce a hidden alignment sequence $\boldsymbol{a}_1^J$ where each $a_j = i$ corresponds to an input position $i \in \{1, \ldots, I\}$ that we want to focus on when generating y_j. Then $p(\boldsymbol{y}|\boldsymbol{x})$ is calculated by marginalizing over all the hidden alignments,

$$
\begin{aligned}
& p(\boldsymbol{y}|\boldsymbol{x}) = \textstyle\sum_{\boldsymbol{a}} p(\boldsymbol{y}, \boldsymbol{a}|\boldsymbol{x}) \qquad (2)\\
\approx \quad & \textstyle\sum_{\boldsymbol{a}} \prod_{j=1}^{J} \underbrace{p(a_j|a_{j-1}, \boldsymbol{y}_1^{j-1}, \boldsymbol{x})}_{\text{transition probability}} \cdot \\
& \qquad \underbrace{p(y_j|\boldsymbol{y}_1^{j-1}, a_j, \boldsymbol{x})}_{\text{word prediction}}.
\end{aligned}
$$

Figure 1 illustrates the model graphically. Each path from the top left node to the right-most column in the graph corresponds to an alignment. We constrain the alignments to be monotone, i.e. only forward and downward transitions are permitted at each point in the grid. This constraint enables the model to learn to perform online generation. Additionally, the model learns to align input and output segments, which means that it can learn local reorderings by memorizing phrases. Another possible constraint on the alignments would be to ensure that the entire input sequence is consumed before last output word is emitted, i.e. all valid alignment paths have to end in the bottom right corner of the grid. However, we do not enforce this constraint in our setup.

The probability contributed by an alignment is obtained by accumulating the probability of word predictions at each point on the path and the transition probability between points. The transition probabilities and the word output probabilities are modeled by neural networks, which are described in detail in the following sub-sections.

2.1 Probabilities of Output Word Predictions

The input sentence $\boldsymbol{x}$ is encoded with a Recurrent Neural Network (RNN), in particular an LSTM (Hochreiter and Schmidhuber, 1997). The encoder can either be a unidirectional or bidirectional LSTM. If a unidirectional encoder is used the model is able to read input and generate output symbols online. The hidden state vectors are computed as

$$\mathbf{h}_i^{\rightarrow} = \text{RNN}(\mathbf{h}_{i-1}^{\rightarrow}, v^{(e)}(x_i)), \qquad (3)$$

$$\mathbf{h}_i^{\leftarrow} = \text{RNN}(\mathbf{h}_{i+1}^{\leftarrow}, v^{(e)}(x_i)), \qquad (4)$$

where $v^{(e)}(x_i)$ denotes the vector representation of the token x, and $\mathbf{h}_i^{\rightarrow}$ and $\mathbf{h}_i^{\leftarrow}$ are the forward and backward hidden states, respectively. For a bidirectional encoder, they are concatenated as $\mathbf{h}_i = [\mathbf{h}_i^{\rightarrow}; \mathbf{h}_i^{\leftarrow}]$; and for unidirectional encoder $\mathbf{h}_i = \mathbf{h}_i^{\rightarrow}$. The hidden state $\mathbf{s}_j$ of the RNN for the output sequence $\boldsymbol{y}$ is computed as

$$\mathbf{s}_j = \text{RNN}(\mathbf{s}_{j-1}, v^{(d)}(y_{j-1})), \qquad (5)$$

where $v^{(d)}(y_{j-1})$ is the encoded vector of the previously generated output word y_{j-1}. That is, $\mathbf{s}_j$ encodes $\boldsymbol{y}_1^{j-1}$.

To calculate the probability of the next word, we concatenate the aligned hidden state vectors $\mathbf{s}_j$ and $\mathbf{h}_{a_j}$ and feed the result into a softmax layer,

$$
\begin{aligned}
& p(y_j = l|\boldsymbol{y}_1^{j-1}, a_j, \boldsymbol{x}) \\
&= p(y_j = l|\mathbf{h}_{a_j}, \mathbf{s}_j) \qquad (6)\\
&= \text{softmax}(\mathbf{W}_w[\mathbf{h}_{a_j}; \mathbf{s}_j] + \mathbf{b}_w)_l.
\end{aligned}
$$

The word output distribution in Graves (2012) is parameterised in similar way.

Figure 2 illustrates the model structure. Note that the hidden states of the input and output decoders are kept independent to permit tractable inference, while the output distributions are conditionally dependent on both.

2.2 Transition Probabilities

As the alignments are constrained to be monotone, we can treat the transition from timestep j to $j+1$ as a sequence of `shift` and `emit` operations. Specifically, at each input position, a decision of `shift` or `emit` is made by the model; if the operation is `emit` then the next output word is generated; otherwise, the model will `shift` to the next input word. While the multinomial distribution is an alternative for parameterising alignments, the shift/emit parameterisation does not place an upper limit on the jump size, as a multinomial distribution would, and biases the model towards shorter jump sizes, which a multinomial model would have to learn.

We describe two methods for modelling the alignment transition probability. The first approach is independent of the input or output words. To parameterise the alignment distribution in terms of shift and emit operations we use a geometric distribution,

$$p(a_j|a_{j-1}) = (1-e)^{a_j - a_{j-1}} e, \qquad (7)$$

where e is the emission probability. This transition probability only has one parameter e, which can be

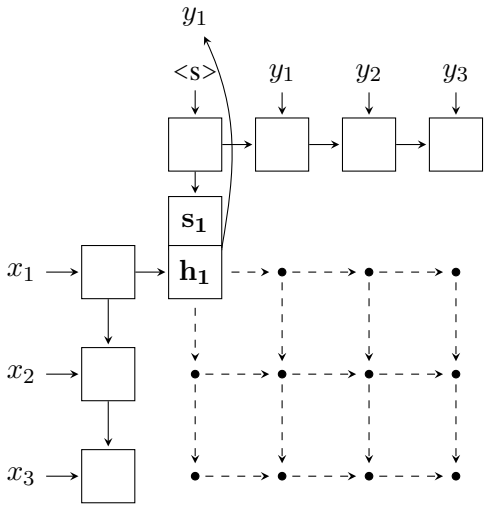


Figure 2: The structure of our model. (x_1, x_2, x_3) and (y_1, y_2, y_3) denote the input and output sequences, respectively. The points, e.g. (i, j), in the grid represent an alignment between x_i and y_j. For each column j, the concatenation of the hidden states $[\mathbf{h}_i, \mathbf{s}_j]$ is used to predict y_j.

estimated directly by maximum likelihood as

$$e = \frac{\sum_n J_n}{\sum_n I_n + \sum_n J_n}, \tag{8}$$

where I_n and J_n are the lengths of the input and output sequences of training example n, respectively.

For the second method we model the transition probability with a neural network,

$$\begin{aligned} p(a_1 = i) &= \prod_{d=1}^{i-1}(1 - p(e_{d,1}))p(e_{i,1}), \\ p(a_j = i | a_{j-1} = k) &= \prod_{d=k}^{i-1}(1 - p(e_{d,j}))p(e_{i,j}), \end{aligned} \tag{9}$$

where $p(e_{i,j})$ denotes the probability of `emit` for the alignment $a_j = i$. This probability is obtained by feeding $[\mathbf{h}_i; \boldsymbol{s}_j]$ into a feed forward neural network,

$$p(e_{i,j}) = \sigma(\text{MLP}(\mathbf{W}_t[\mathbf{h}_i; \mathbf{s}_j] + b_t)). \tag{10}$$

For simplicity, $p(a_j = i | a_{j-1} = k, \mathbf{s}_j, \mathbf{h}_k^i)$ is abbreviated as $p(a_j = i | a_{j-1} = k)$.

3 Training and Decoding

Since there are an exponential number of possible alignments, it is computationally intractable to explicitly calculate every $p(\boldsymbol{y}, \boldsymbol{a}|\boldsymbol{x})$ and then sum them to get the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$. We instead approach the problem using a dynamic-programming algorithm similar to the forward-backward algorithm for HMMs (Rabiner, 1989).

3.1 Training

For an input $\boldsymbol{x}$ and output $\boldsymbol{y}$, the forward variable $\alpha(i, j) = p(a_j = i, \boldsymbol{y}_1^j | \boldsymbol{x})$. The value of $\alpha(i, j)$ is computed by summing over the probabilities of every path that could lead to this cell. Formally, $\alpha(i, j)$ is defined as follows:

For $i \in [1, I]$:

$$\alpha(i, 1) = p(a_1 = i) p(y_1 | \mathbf{h}_i, \mathbf{s}_1). \tag{11}$$

For $j \in [2, J]$, $i \in [1, I]$:

$$\begin{aligned} \alpha(i, j) = p(y_j | \mathbf{h}_i, \mathbf{s}_j) \cdot \\ \sum_{k=1}^{i} \alpha(k, j-1) p(a_j = i | a_{j-1} = k). \end{aligned} \tag{12}$$

The backward variables, defined as $\beta(i, j) = p(\boldsymbol{y}_{j+1}^J | a_j = i, \boldsymbol{y}_1^j, \boldsymbol{x})$, are computed as:

For $i \in [1, I]$:

$$\beta(i, J) = 1. \tag{13}$$

For $j \in [1, J-1]$, $i \in [1, I]$:

$$\begin{aligned} \beta(i, j) = \sum_{k=i}^{I} p(a_{j+1} = k | a_j = i) \beta(k, j+1) \cdot \\ p(y_{j+1} | \mathbf{h}_k, \mathbf{s}_{j+1}). \end{aligned} \tag{14}$$

During training we estimate the parameters by minimizing the negative log likelihood of the training set S:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= -\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S} \log p(\boldsymbol{y} | \boldsymbol{x}; \boldsymbol{\theta}) \\ &= -\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S} \log \sum_{i=1}^{I} \alpha(i, J). \end{aligned} \tag{15}$$

Let $\boldsymbol{\theta_j}$ be the neural network parameters w.r.t. the model output at position j. The gradient is computed as:

$$\frac{\partial \log p(\boldsymbol{y} | \boldsymbol{x}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{j=1}^{J} \sum_{i=1}^{I} \frac{\partial \log p(\boldsymbol{y} | \boldsymbol{x}; \boldsymbol{\theta})}{\partial \alpha(i, j)} \frac{\partial \alpha(i, j)}{\partial \boldsymbol{\theta_j}}. \tag{16}$$

The derivative w.r.t. the forward weights is

$$\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}{\partial \alpha(i,j)} = \frac{\beta(i,j)}{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}. \quad (17)$$

The derivative of the forward weights w.r.t. the model parameters at position j is

$$\frac{\partial \alpha(i,j)}{\partial \boldsymbol{\theta_j}} = \frac{\partial p(y_j|\mathbf{h}_i,\mathbf{s}_j)}{\partial \boldsymbol{\theta_j}} \frac{\alpha(i,j)}{p(y_j|\mathbf{h}_i,\mathbf{s}_j)} + p(y_j|\mathbf{h}_i,\mathbf{s}_j) \sum_{k=1}^{i} \alpha(j-1,k) \frac{\partial}{\partial \boldsymbol{\theta_j}} p(a_j{=}i|a_{j-1}{=}k). \quad (18)$$

For the geometric distribution transition probability model $\frac{\partial}{\partial \boldsymbol{\theta_j}} p(a_j = i|a_{j-1} = k) = 0$.

3.2 Decoding

Algorithm 1 DP search algorithm

Input: source sentence $\boldsymbol{x}$
Output: best output sentence $\boldsymbol{y*}$
Initialization: $Q \in \mathbb{R}^{I \times J_{\text{max}}}$, bp $\in \mathbb{N}^{I \times J_{\text{max}}}$, $W \in \mathbb{N}^{I \times J_{\text{max}}}$, I_{end}, J_{end}.
for $i \in [1, I]$ **do**
  $Q[i,1] \leftarrow \max_{y \in \mathcal{V}} p(a_1 = i) p(y|\mathbf{h}_i, \mathbf{s}_1)$
  $bp[i,1] \leftarrow 0$
  $W[i,1] \leftarrow \arg\max_{y \in \mathcal{V}} p(a_1 = i) p(y|\mathbf{h}_i, \mathbf{s}_1)$
end for
for $j \in [2, J_{\text{max}}]$ **do**
  for $i \in [1, I]$ **do**
    $Q[i,j] \leftarrow \max_{y \in \mathcal{V}, k \in [1,i]} Q[k, j-1] \cdot p(a_j = i|a_{j-1} = k) p(y|\mathbf{h}_i, \mathbf{s}_j)$
    $bp[i,j], W[i,j] \leftarrow \arg\max_{y \in \mathcal{V}, k \in [1,i]} \cdot Q[k, j-1] p(a_j = i|a_{j-1} = k) p(y|\mathbf{h}_i, \mathbf{s}_j)$
  end for
  $I_{\text{end}} \leftarrow \arg\max_i Q[i,j]$
  if $W[I_{\text{end}}, j] = \text{EOS}$ **then**
    $J_{\text{end}} \leftarrow j$
    break
  end if
end for
return a sequence of words stored in W by following backpointers starting from $(I_{\text{end}}, J_{\text{end}})$.

For decoding, we aim to find the best output sequence $\boldsymbol{y}^*$ for a given input sequence $\boldsymbol{x}$:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \quad (19)$$

The search algorithm is based on dynamic programming (Tillmann et al., 1997). The main idea is to create a path probability matrix Q, and fill each cell $Q[i,j]$ by recursively taking the most probable path that could lead to this cell. We present the greedy search algorithm in Algorithm 1. We also implemented a beam search that tracks the k best partial sequences at position (i,j). The notation bp refers to backpointers, W stores words to be predicted, $\mathcal{V}$ denotes the output vocabulary, J_{max} is the maximum length of the output sequences that the model is allowed to predict.

4 Experiments

We evaluate the effectiveness of our model on two representative natural language processing tasks, sentence compression and morphological inflection. The primary aim of this evaluation is to assess whether our proposed architecture is able to outperform the baseline encoder-decoder model by overcoming its encoding bottleneck. We further benchmark our results against an attention model in order to determine whether our alternative alignment strategy is able to provide similar benefits while processing the input online.

4.1 Abstractive Sentence Summarisation

Sentence summarisation is the task of generating a condensed version of a sentence while preserving its meaning. In abstractive sentence summarisation, summaries are generated from the given vocabulary without the constraint of copying words in the input sentence. Rush et al. (2015) compiled a data set for this task from the annotated Gigaword data set (Graff et al., 2003; Napoles et al., 2012), where sentence-summary pairs are obtained by pairing the headline of each article with its first sentence. Rush et al. (2015) use the splits of 3.8m/190k/381k for training, validation and testing. In previous work on this dataset, Rush et al. (2015) proposed an attention-based model with feed-forward neural networks, and Chopra et al. (2016) proposed an attention-based recurrent encoder-decoder, similar to one of our baselines.

Due to computational constraints we place the following restrictions on the training and validation set:

1. The maximum lengths for the input sentences

Model	ROUGE-1	ROUGE-2	ROUGE-L
Seq2seq	25.16	9.09	23.06
Attention	29.25	12.85	27.32
uniSSNT	26.96	10.54	24.59
biSSNT	27.05	10.62	24.64
uniSSNT+	30.15	13.59	27.88
biSSNT+	**30.27**	**13.68**	**27.91**

Table 1: ROUGE F1 scores on the sentence summarisation test set. Seq2seq refers to the vanilla encoder-decoder and attention denotes the attention-based model. SSNT denotes our model with alignment transition probability modelled as geometric distribution. SSNT+ refers to our model with transition probability modelled using neural networks. The prefixes uni- and bi- denote using unidirectional and bidirectional encoder LSTMs, respectively.

Model	Configuration	Perplexity
Seq2seq	$H = 128, L = 1$	48.5
	$H = 256, L = 1$	35.6
	$H = 256, L = 2$	32.1
	$H = 256, L = 3$	31.0
biSSNT+	$H = 128, L = 1$	26.7
	$H = 256, L = 1$	**22.6**

Table 2: Perplexity on the validation set with 172k sentence-summary pairs.

and summaries are 50 and 25, respectively.

2. For each sentence-summary pair, the product of the input and output lengths should be no greater than 500.

We use the filtered 172k pairs for validation and sample 1m pairs for training. While this training set is smaller than that used in previous work (and therefore our results cannot be compared directly against reported results), it serves our purpose for evaluating our algorithm against sequence to sequence and attention-based approaches under identical data conditions. Following from previous work (Rush et al., 2015; Chopra et al., 2016; Gülçehre et al., 2016), we report results on a randomly sampled test set of 2000 sentence-summary pairs. The quality of the generated summaries are evaluated by three versions of ROUGE for different match lengths, namely ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest-common substring).

For training, we use Adam (Kingma and Ba, 2015) for optimization, with an initial learning rate of 0.001. The mini-batch size is set to 32. The number of hidden units H is set to 256 for both our model and the baseline models, and dropout of 0.2 is applied to the input of LSTMs. All hyperparameters were optimised via grid search on the perplexity of the validation set. We use greedy decoding to generate summaries.

Table 1 displays the ROUGE-F1 scores of our models on the test set, together with baseline models, including the attention-based model. Our models achieve significantly better results than the vanilla encoder-decoder and outperform the attention-based model. The fact that SSNT+ performs better is in line with our expectations, as the neural network-parameterised alignment model is more expressive than that modelled by geometric distribution.

To make further comparison, we experimented with different sizes of hidden units and adding more layers to the baseline encoder-decoder. Table 2 lists the configurations of different models and their corresponding perplexities on the validation set. We can see that the vanilla encoder-decoder tends to get better results by adding more hidden units and stacking more layers. This is due to the limitation of compressing information into a fixed-size vector. It has to use larger vectors and deeper structure in order to memorize more information. By contrast, our model can do well with smaller networks. In fact, even with 1 layer and 128 hidden units, our model works much better than the vanilla encoder-decoder with 3 layers and 256 hidden units per layer.

4.2 Morphological Inflection

Morphological inflection generation is the task of predicting the inflected form of a given lexical item based on a morphological attribute. The transformation from a base form to an inflected form usually includes concatenating it with a prefix or a suffix and substituting some characters. For example, the inflected form of a German stem *abgang* is *abgängen* when the case is dative and the number is plural.

In our experiments, we use the same dataset as

Model	Avg. accuracy
Seq2Seq	79.08
Seq2Seq w/ Attention	95.64
Adapted-seq2seq (FTND16)	**96.20**
uniSSNT+	87.85
biSSNT+	95.32

Table 3: Average accuracy over all the morphological inflection datasets. The baseline results for Seq2Seq variants are taken from (Faruqui et al., 2016).

Dataset	DDN13	NCK15	FTND16	biSSNT+
de-N	88.31	**88.60**	88.12	87.50
de-V	94.76	97.50	**97.72**	92.11
es-V	99.61	99.80	**99.81**	99.52
fi-NA	92.14	93.00	95.44	**95.48**
fi-V	97.23	98.10	97.81	**98.10**
fr-V	98.80	**99.20**	98.82	98.65
nl-V	90.50	96.10	96.71	95.90
Avg.	94.47	96.04	**96.20**	95.32

Table 4: Comparison of the performance of our model (biSSNT+) against the previous state-of-the-art on each morphological inflection dataset.

Faruqui et al. (2016). This dataset was originally created by Durrett and DeNero (2013) from Wiktionary, containing inflections for German nouns (de-N), German verbs (de-V), Spanish verbs (es-V), Finnish noun and adjective (fi-NA), and Finnish verbs (fi-V). It was further expanded by Nicolai et al. (2015) by adding Dutch verbs (nl-V) and French verbs (fr-V). The number of inflection types for each language ranges from 8 to 57. The number of base forms, i.e. the number of instances in each dataset, ranges from 2000 to 11200. The predefined split is 200/200 for dev and test sets, and the rest of the data for training.

Our model is trained separately for each type of inflection, the same setting as the factored model described in Faruqui et al. (2016). The model is trained to predict the character sequence of the inflected form given that of the stem. The output is evaluated by accuracies of string matching. For all the experiments on this task we use 128 hidden units for the LSTMs and apply dropout of 0.5 on the input and output of the LSTMs. We use Adam (Kingma and Ba, 2015) for optimisation with initial learning rate of 0.001. During decoding, beam search is employed with beam size of 30.

Table 3 gives the average accuracy of the uniSSNT+, biSSNT+, vanilla encoder-decoder, and attention-based models. The model with the best previous average result — denoted as adapted-seq2seq (FTND16) (Faruqui et al., 2016) — is also included for comparison. Our biSSNT+ model outperforms the vanilla encoder-decoder by a large margin and almost matches the state-of-the-art result on this task. As mentioned earlier, a characteristic of these datasets is that the stems and their corresponding inflected forms mostly overlap. Compare to the vanilla encoder-decoder, our model is better at copying and finding correspondences between prefix, stem and suffix segments.

Table 4 compares the results of biSSNT+ and previous models on each individual dataset. DDN13 and NCK15 denote the models of Durrett and DeNero (2013) and Nicolai et al. (2015), respectively. Both models tackle the task by feature engineering. FTND16 (Faruqui et al., 2016) adapted the vanilla encoder-decoder by feeding the i-th character of the encoded string as an extra input into the i-th position of the decoder. It can be considered as a special case of our model by forcing a fixed diagonal alignment between input and output sequences. Our model achieves comparable results to these models on all the datasets. Notably it outperforms other models on the Finnish noun and adjective, and verbs datasets, whose stems and inflected forms are the longest.

5 Alignment Quality

Figure 3 presents visualisations of segment alignments generated by our model for sample instances from both tasks. We see that the model is able to learn the correct correspondences between segments of the input and output sequences. For instance, the alignment follows a nearly diagonal path for the example in Figure 3c, where the input and output sequences are identical. In Figure 3b, it learns to add the prefix 'ge' at the start of the sequence and replace 'en' with 't' after copying 'zock'. We observe that the model is robust on long phrasal mappings. As

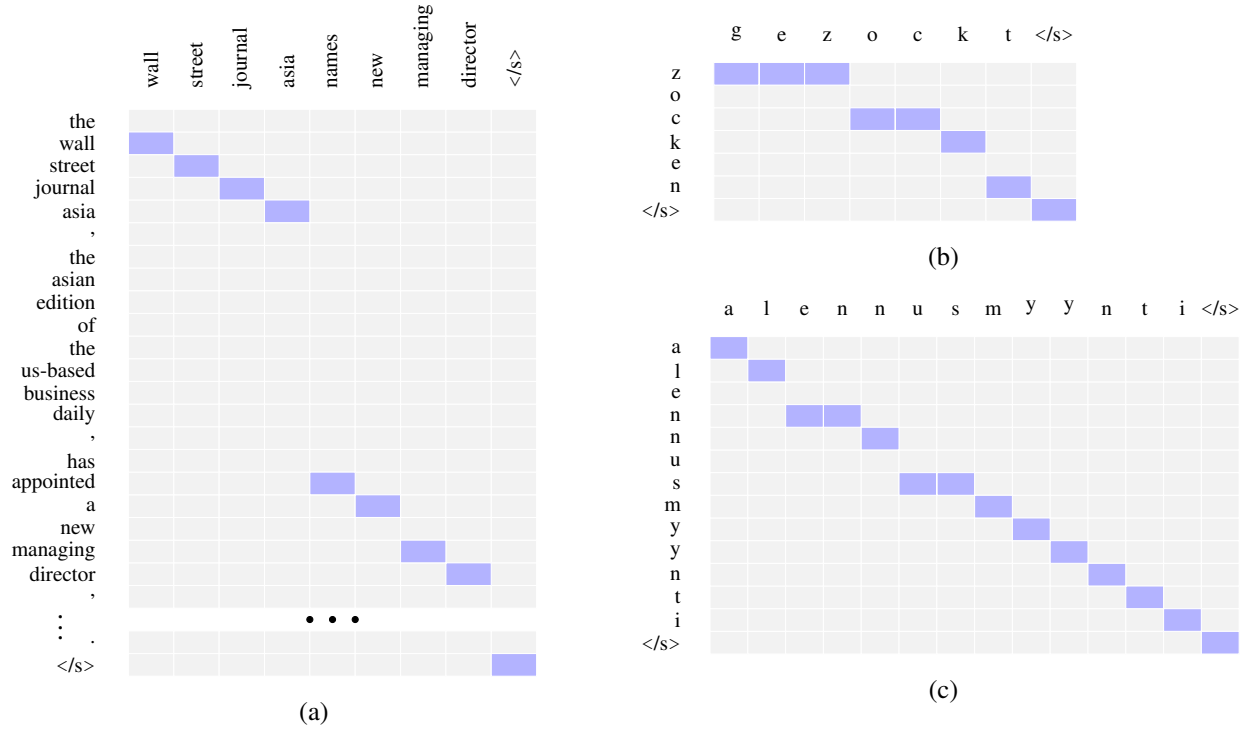


Figure 3: Example alignments found by BiSSNT+. Highlighted grid cells represent the correspondence between the input and output tokens.

shown in Figure 3a, the mapping between 'the wall street journal asia, the asian edition of the us-based business daily' and 'wall street journal asia' demonstrates that our model learns to ignore phrasal modifiers containing additional information. We also find some examples of word reordering, e.g., the phrase 'industrial production in france' is reordered as 'france industrial output' in the model's predicted output.

6 Related Work

Our work is inspired by the seminal HMM alignment model (Vogel et al., 1996; Tillmann et al., 1997) proposed for machine translation. In contrast to that work, when predicting a target word we additionally condition on all previously generated words, which is enabled by the recurrent neural models. This means that the model also functions as a conditional language model. It can therefore be applied directly, while traditional models have to be combined with a language model through a noisy channel in order to be effective. Additionally, instead of EM training on the most likely alignments at each iteration, our model is trained with direct gradient descent, marginalizing over all the alignments.

Latent variables have been employed in neural network-based models for sequence labelling tasks in the past. Examples include connectionist temporal classification (CTC) (Graves et al., 2006) for speech recognition and the more recent segmental recurrent neural networks (SRNNs) (Kong et al., 2016), with applications on handwriting recognition and part-of-speech tagging. Weighted finite-state transducers (WFSTs) have also been augmented to encode input sequences with bidirectional LSTMs (Rastogi et al., 2016), permitting exact inference over all possible output strings. While these models have been shown to achieve appealing performance on different applications, they have common limitations in terms of modelling dependencies between labels. It is not possible for CTCs to model explicit dependencies. SRNNs and neural WFSTs model fixed-length dependencies, making it is difficult to carry out effective inference as the dependencies become longer.

Our model shares the property of the sequence

transduction model of Graves (2012) in being able to model unbounded dependencies between output tokens via an output RNN. This property makes it possible to apply our model to tasks like summarisation and machine translation that require the tokens in the output sequence to be modelled highly dependently. Graves (2012) models the joint distribution over outputs and alignments by inserting null symbols (representing shift operations) into the output sequence. During training the model uses dynamic programming to marginalize over permutations of the null symbols, while beam search is employed during decoding. In contrast our model defines a separate latent alignment variable, which adds flexibility to the way the alignment distribution can be defined (as a geometric distribution or parameterised by a neural network) and how the alignments can be constrained, without redefining the dynamic program. In addition to marginalizing during training, our decoding algorithm also makes use of dynamic programming, allowing us to use either no beam or small beam sizes.

Our work is also related to the attention-based models first introduced for machine translation (Bahdanau et al., 2015). Luong et al. (2015) proposed two alternative attention mechanisms: a global method that attends all words in the input sentence, and a local one that points to parts of the input words. Another variation on this theme are pointer networks (Vinyals et al., 2015), where the outputs are pointers to elements of the variable-length input, predicted by the attention distribution. Jaitly et al. (2016) propose an online sequence to sequence model with attention that conditions on fixed-sized blocks of the input sequence and emits output tokens corresponding to each block. The model is trained with alignment information to generate supervised segmentations.

Although our model shares the same idea of joint training and aligning with the attention-based models, our design has fundamental differences and advantages. While attention-based models treat the attention weights as output of a deterministic function (soft-alignment), in our model the attention weights correspond to a hidden variable, that can be marginalized out using dynamic programming. Further, our model's inherent online nature permits it the flexibility to use its capacity to chose how much input to encode before decoding each segment.

7 Conclusion

We have proposed a novel segment to segment neural transduction model that tackles the limitations of vanilla encoder-decoders that have to read and memorize an entire input sequence in a fixed-length context vector before producing any output. By introducing a latent segmentation that determines correspondences between tokens of the input and output sequences, our model learns to generate and align jointly. During training, the hidden alignment is marginalized out using dynamic programming, and during decoding the best alignment path is generated alongside the predicted output sequence. By employing a unidirectional LSTM as encoder, our model is capable of doing online generation. Experiments on two representative natural language processing tasks, abstractive sentence summarisation and morphological inflection generation, showed that our model significantly outperforms encoder-decoder baselines while requiring much smaller hidden layers. For future work we would like to incorporate attention-based models to our framework to enable such models to process data online.

Acknowledgments

We thank Chris Dyer, Karl Moritz Hermann, Edward Grefenstette, Tomáš Kočiský, Gabor Melis, Yishu Miao and many others for their helpful comments. The first author is funded by EPSRC.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of HLT-NAACL*.

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

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Proceedings of NIPS*, pages 1819–1827.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *CoRR*, abs/1603.08148.

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

Navdeep Jaitly, David Sussillo, Quoc V. Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2016. A neural transducer. In *Proceedings of NIPS*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICIR*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proceedings of ICLR*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of ICML*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of NAACL*.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of NAACL*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. In *Proceedings of EACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of ICLR*.

Sequence-Level Knowledge Distillation

Yoon Kim
yoonkim@seas.harvard.edu

Alexander M. Rush
srush@seas.harvard.edu

School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA

Abstract

Neural machine translation (NMT) offers a novel alternative formulation of translation that is potentially simpler than statistical approaches. However to reach competitive performance, NMT models need to be exceedingly large. In this paper we consider applying *knowledge distillation* approaches (Bucila et al., 2006; Hinton et al., 2015) that have proven successful for reducing the size of neural models in other domains to the problem of NMT. We demonstrate that standard knowledge distillation applied to word-level prediction can be effective for NMT, and also introduce two novel *sequence-level* versions of knowledge distillation that further improve performance, and somewhat surprisingly, seem to eliminate the need for beam search (even when applied on the original teacher model). Our best student model runs 10 times faster than its state-of-the-art teacher with little loss in performance. It is also significantly better than a baseline model trained without knowledge distillation: by $4.2/1.7$ BLEU with greedy decoding/beam search. Applying weight pruning on top of knowledge distillation results in a student model that has $13\times$ fewer parameters than the original teacher model, with a decrease of 0.4 BLEU.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) is a deep learning-based method for translation that has recently shown promising results as an alternative to statistical approaches. NMT systems directly model the probability of the next word in the target sentence simply by conditioning a recurrent neural network on the source sentence and previously generated target words.

While both simple and surprisingly accurate, NMT systems typically need to have very high capacity in order to perform well: Sutskever et al. (2014) used a 4-layer LSTM with 1000 hidden units per layer (herein 4×1000) and Zhou et al. (2016) obtained state-of-the-art results on English $\rightarrow$ French with a 16-layer LSTM with 512 units per layer. The sheer size of the models requires cutting-edge hardware for training and makes using the models on standard setups very challenging.

This issue of excessively large networks has been observed in several other domains, with much focus on fully-connected and convolutional networks for multi-class classification. Researchers have particularly noted that large networks seem to be necessary for training, but learn redundant representations in the process (Denil et al., 2013). Therefore compressing deep models into smaller networks has been an active area of research. As deep learning systems obtain better results on NLP tasks, compression also becomes an important practical issue with applications such as running deep learning models for speech and translation locally on cell phones.

Existing compression methods generally fall into two categories: (1) *pruning* and (2) *knowledge distillation*. *Pruning* methods (LeCun et al., 1990; He et al., 2014; Han et al., 2016), zero-out weights or entire neurons based on an importance criterion: LeCun et al. (1990) use (a diagonal approximation to)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1317–1327,
Austin, Texas, November 1-5, 2016.

the Hessian to identify weights whose removal minimally impacts the objective function, while Han et al. (2016) remove weights based on thresholding their absolute values. *Knowledge distillation* approaches (Bucila et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) learn a smaller *student* network to mimic the original *teacher* network by minimizing the loss (typically L_2 or cross-entropy) between the student and teacher output.

In this work, we investigate knowledge distillation in the context of neural machine translation. We note that NMT differs from previous work which has mainly explored non-recurrent models in the multi-class prediction setting. For NMT, while the model is trained on multi-class prediction at the word-level, it is tasked with predicting complete sequence outputs conditioned on previous decisions. With this difference in mind, we experiment with standard knowledge distillation for NMT and also propose two new versions of the approach that attempt to approximately match the sequence-level (as opposed to word-level) distribution of the teacher network. This sequence-level approximation leads to a simple training procedure wherein the student network is trained on a newly generated dataset that is the result of running beam search with the teacher network.

We run experiments to compress a large state-of-the-art 4×1000 LSTM model, and find that with sequence-level knowledge distillation we are able to learn a 2×500 LSTM that roughly matches the performance of the full system. We see similar results compressing a 2×500 model down to 2×100 on a smaller data set. Furthermore, we observe that our proposed approach has other benefits, such as not requiring any beam search at test-time. As a result we are able to perform greedy decoding on the 2×500 model 10 times faster than beam search on the 4×1000 model with comparable performance. Our student models can even be run efficiently on a standard smartphone.[1] Finally, we apply weight pruning on top of the student network to obtain a model that has $13\times$ fewer parameters than the original teacher model. We have released all the code for the models described in this paper.[2]

[1] https://github.com/harvardnlp/nmt-android

[2] https://github.com/harvardnlp/seq2seq-attn

2 Background

2.1 Sequence-to-Sequence with Attention

Let $\mathbf{s} = [s_1, \ldots, s_I]$ and $\mathbf{t} = [t_1, \ldots, t_J]$ be (random variable sequences representing) the source/target sentence, with I and J respectively being the source/target lengths. Machine translation involves finding the most probable target sentence given the source:

$$\underset{\mathbf{t} \in \mathcal{T}}{\operatorname{argmax}}\, p(\mathbf{t} \mid \mathbf{s})$$

where $\mathcal{T}$ is the set of all possible sequences. NMT models parameterize $p(\mathbf{t} \mid \mathbf{s})$ with an *encoder* neural network which reads the source sentence and a *decoder* neural network which produces a distribution over the target sentence (one word at a time) given the source. We employ the attentional architecture from Luong et al. (2015), which achieved state-of-the-art results on English $\rightarrow$ German translation.[3]

2.2 Knowledge Distillation

Knowledge distillation describes a class of methods for training a smaller *student* network to perform better by learning from a larger *teacher* network (in addition to learning from the training data set). We generally assume that the teacher has previously been trained, and that we are estimating parameters for the student. Knowledge distillation suggests training by matching the student's predictions to the teacher's predictions. For classification this usually means matching the probabilities either via L_2 on the $\log$ scale (Ba and Caruana, 2014) or by cross-entropy (Li et al., 2014; Hinton et al., 2015).

Concretely, assume we are learning a multi-class classifier over a data set of examples of the form (x, y) with possible classes $\mathcal{V}$. The usual training criteria is to minimize NLL for each example from the training data,

$$\mathcal{L}_{\text{NLL}}(\theta) = -\sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y = k\} \log p(y = k \mid x; \theta)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and p the distribution from our model (parameterized by θ).

[3] Specifically, we use the *global-general* attention model with the *input-feeding* approach. We refer the reader to the original paper for further details.

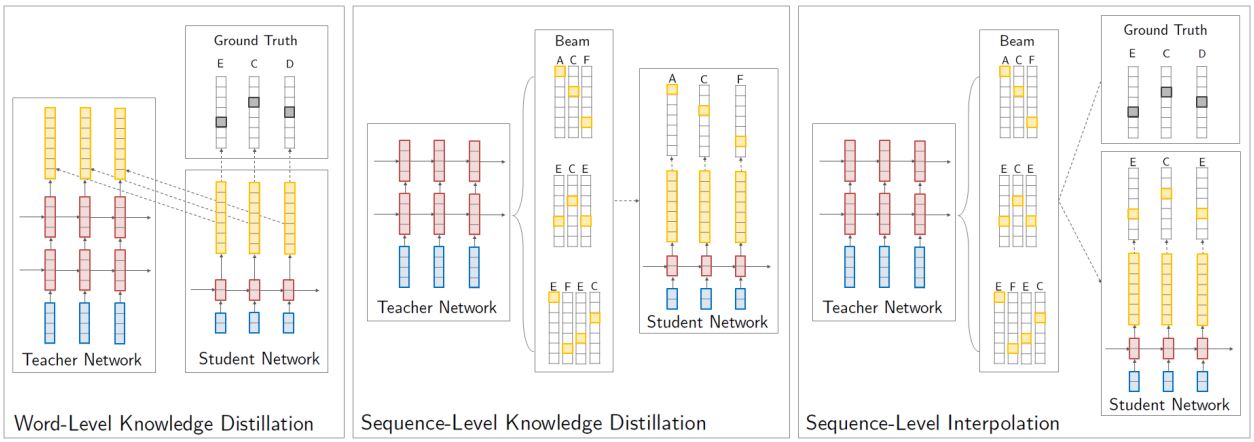


Figure 1: Overview of the different knowledge distillation approaches. In word-level knowledge distillation (left) cross-entropy is minimized between the student/teacher distributions (yellow) for each word in the actual target sequence (ECD), as well as between the student distribution and the degenerate data distribution, which has all of its probabilitiy mass on one word (black). In sequence-level knowledge distillation (center) the student network is trained on the output from beam search of the teacher network that had the highest score (ACF). In sequence-level interpolation (right) the student is trained on the output from beam search of the teacher network that had the highest sim with the target sequence (ECE).

This objective can be seen as minimizing the cross-entropy between the degenerate data distribution (which has all of its probability mass on one class) and the model distribution $p(y \mid x; \theta)$.

In knowledge distillation, we assume access to a learned teacher distribution $q(y \mid x; \theta_T)$, possibly trained over the same data set. Instead of minimizing cross-entropy with the observed data, we instead minimize the cross-entropy with the teacher's probability distribution,

$$\mathcal{L}_{\text{KD}}(\theta; \theta_T) = -\sum_{k=1}^{|\mathcal{V}|} q(y = k \mid x; \theta_T) \times \log p(y = k \mid x; \theta)$$

where θ_T parameterizes the teacher distribution and remains fixed. Note the cross-entropy setup is identical, but the target distribution is no longer a sparse distribution.[4] Training on $q(y \mid x; \theta_T)$ is attractive since it gives more information about other classes for a given data point (e.g. similarity between classes) and has less variance in gradients (Hinton et al., 2015).

[4] In some cases the entropy of the teacher/student distribution is increased by annealing it with a temperature term $\tau > 1$

$$\tilde{p}(y \mid x) \propto p(y \mid x)^{\frac{1}{\tau}}$$

After testing $\tau \in \{1, 1.5, 2\}$ we found that $\tau = 1$ worked best.

Since this new objective has no direct term for the training data, it is common practice to interpolate between the two losses,

$$\mathcal{L}(\theta; \theta_T) = (1 - \alpha)\mathcal{L}_{\text{NLL}}(\theta) + \alpha\mathcal{L}_{\text{KD}}(\theta; \theta_T)$$

where α is mixture parameter combining the one-hot distribution and the teacher distribution.

3 Knowledge Distillation for NMT

The large sizes of neural machine translation systems make them an ideal candidate for knowledge distillation approaches. In this section we explore three different ways this technique can be applied to NMT.

3.1 Word-Level Knowledge Distillation

NMT systems are trained directly to minimize word NLL, $\mathcal{L}_{\text{WORD-NLL}}$, at each position. Therefore if we have a teacher model, standard knowledge distillation for multi-class cross-entropy can be applied. We define this distillation for a sentence as,

$$\mathcal{L}_{\text{WORD-KD}} = -\sum_{j=1}^{J}\sum_{k=1}^{|\mathcal{V}|} q(t_j = k \mid \mathbf{s}, \mathbf{t}_{<j}) \times \log p(t_j = k \mid \mathbf{s}, \mathbf{t}_{<j})$$

where $\mathcal{V}$ is the target vocabulary set. The student can further be trained to optimize the mixture of

$\mathcal{L}_{\text{WORD-KD}}$ and $\mathcal{L}_{\text{WORD-NLL}}$. In the context of NMT, we refer to this approach as *word-level* knowledge distillation and illustrate this in Figure 1 (left).

3.2 Sequence-Level Knowledge Distillation

Word-level knowledge distillation allows transfer of these local word distributions. Ideally however, we would like the student model to mimic the teacher's actions at the *sequence-level*. The sequence distribution is particularly important for NMT, because wrong predictions can propagate forward at test-time.

First, consider the sequence-level distribution specified by the model over all possible sequences $\mathbf{t} \in \mathcal{T}$,

$$p(\mathbf{t} \mid \mathbf{s}) = \prod_{j=1}^{J} p(t_j \mid \mathbf{s}, \mathbf{t}_{<j})$$

for any length J. The sequence-level negative log-likelihood for NMT then involves matching the one-hot distribution over all complete sequences,

$$\begin{aligned}
\mathcal{L}_{\text{SEQ-NLL}} &= -\sum_{\mathbf{t} \in \mathcal{T}} \mathbb{1}\{\mathbf{t} = \mathbf{y}\} \log p(\mathbf{t} \mid \mathbf{s}) \\
&= -\sum_{j=1}^{J} \sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y_j = k\} \log p(t_j = k \mid \mathbf{s}, \mathbf{t}_{<j}) \\
&= \mathcal{L}_{\text{WORD-NLL}}
\end{aligned}$$

where $\mathbf{y} = [y_1, \ldots, y_J]$ is the observed sequence. Of course, this just shows that from a negative log likelihood perspective, minimizing word-level NLL and sequence-level NLL are equivalent in this model.

But now consider the case of sequence-level knowledge distillation. As before, we can simply replace the distribution from the data with a probability distribution derived from our teacher model. However, instead of using a single word prediction, we use $q(\mathbf{t} \mid \mathbf{s})$ to represent the teacher's sequence distribution over the sample space of all possible sequences,

$$\mathcal{L}_{\text{SEQ-KD}} = -\sum_{\mathbf{t} \in \mathcal{T}} q(\mathbf{t} \mid \mathbf{s}) \log p(\mathbf{t} \mid \mathbf{s})$$

Note that $\mathcal{L}_{\text{SEQ-KD}}$ is inherently different from $\mathcal{L}_{\text{WORD-KD}}$, as the sum is over an exponential number of terms. Despite its intractability, we posit that this sequence-level objective is worthwhile. It gives the teacher the chance to assign probabilities to complete sequences and therefore transfer a broader range of knowledge. We thus consider an approximation of this objective.

Our simplest approximation is to replace the teacher distribution q with its mode,

$$q(\mathbf{t} \mid \mathbf{s}) \sim \mathbb{1}\{\mathbf{t} = \operatorname*{argmax}_{\mathbf{t} \in \mathcal{T}} q(\mathbf{t} \mid \mathbf{s})\}$$

Observing that finding the mode is itself intractable, we use beam search to find an approximation. The loss is then

$$\begin{aligned}
\mathcal{L}_{\text{SEQ-KD}} &\approx -\sum_{\mathbf{t} \in \mathcal{T}} \mathbb{1}\{\mathbf{t} = \hat{\mathbf{y}}\} \log p(\mathbf{t} \mid \mathbf{s}) \\
&= -\log p(\mathbf{t} = \hat{\mathbf{y}} \mid \mathbf{s})
\end{aligned}$$

where $\hat{\mathbf{y}}$ is now the output from running beam search with the teacher model.

Using the mode seems like a poor approximation for the teacher distribution $q(\mathbf{t} \mid \mathbf{s})$, as we are approximating an exponentially-sized distribution with a single sample. However, previous results showing the effectiveness of beam search decoding for NMT lead us to belief that a large portion of q's mass lies in a single output sequence. In fact, in experiments we find that with beam of size 1, $q(\hat{\mathbf{y}} \mid \mathbf{s})$ (on average) accounts for 1.3% of the distribution for German $\rightarrow$ English, and 2.3% for Thai $\rightarrow$ English (Table 1: $p(\mathbf{t} = \hat{\mathbf{y}})$).[5]

To summarize, sequence-level knowledge distillation suggests to: (1) train a teacher model, (2) run beam search over the training set with this model, (3) train the student network with cross-entropy on this new dataset. Step (3) is identical to the word-level NLL process except now on the newly-generated data set. This is shown in Figure 1 (center).

[5] Additionally there are simple ways to better approximate $q(\mathbf{t} \mid \mathbf{s})$. One way would be to consider a K-best list from beam search and renormalizing the probabilities,

$$q(\mathbf{t} \mid \mathbf{s}) \sim \frac{q(\mathbf{t} \mid \mathbf{s})}{\sum_{\mathbf{t} \in \mathcal{T}_K} q(\mathbf{t} \mid \mathbf{s})}$$

where $\mathcal{T}_K$ is the K-best list from beam search. This would increase the training set by a factor of K. A beam of size 5 captures 2.8% of the distribution for German $\rightarrow$ English, and 3.8% for Thai $\rightarrow$ English. Another alternative is to use a Monte Carlo estimate and sample from the teacher model (since $\mathcal{L}_{\text{SEQ-KD}} = \mathbb{E}_{\mathbf{t} \sim q(\mathbf{t} \mid \mathbf{s})}[-\log p(\mathbf{t} \mid \mathbf{s})]$). However in practice we found the (approximate) mode to work well.

3.3 Sequence-Level Interpolation

Next we consider integrating the training data back into the process, such that we train the student model as a mixture of our sequence-level teacher-generated data ($\mathcal{L}_{\text{SEQ-KD}}$) with the original training data ($\mathcal{L}_{\text{SEQ-NLL}}$),

$$\begin{aligned}\mathcal{L} &= (1-\alpha)\mathcal{L}_{\text{SEQ-NLL}} + \alpha\mathcal{L}_{\text{SEQ-KD}} \\ &= -(1-\alpha)\log p(\mathbf{y}\,|\,\mathbf{s}) - \alpha\sum_{\mathbf{t}\in\mathcal{T}} q(\mathbf{t}\,|\,\mathbf{s})\log p(\mathbf{t}\,|\,\mathbf{s})\end{aligned}$$

where $\mathbf{y}$ is the gold target sequence.

Since the second term is intractable, we could again apply the mode approximation from the previous section,

$$\mathcal{L} = -(1-\alpha)\log p(\mathbf{y}\,|\,\mathbf{s}) - \alpha\log p(\hat{\mathbf{y}}\,|\,\mathbf{s})$$

and train on both observed ($\mathbf{y}$) and teacher-generated ($\hat{\mathbf{y}}$) data. However, this process is non-ideal for two reasons: (1) unlike for standard knowledge distribution, it doubles the size of the training data, and (2) it requires training on both the teacher-generated sequence and the true sequence, conditioned on the same source input. The latter concern is particularly problematic since we observe that $\mathbf{y}$ and $\hat{\mathbf{y}}$ are often quite different.

As an alternative, we propose a single-sequence approximation that is more attractive in this setting. This approach is inspired by *local updating* (Liang et al., 2006), a method for discriminative training in statistical machine translation (although to our knowledge not for knowledge distillation). Local updating suggests selecting a training sequence which is close to $\mathbf{y}$ *and* has high probability under the teacher model,

$$\tilde{\mathbf{y}} = \underset{\mathbf{t}\in\mathcal{T}}{\operatorname{argmax}}\, sim(\mathbf{t},\mathbf{y})q(\mathbf{t}\,|\,\mathbf{s})$$

where sim is a function measuring closeness (e.g. Jaccard similarity or BLEU (Papineni et al., 2002)). Following local updating, we can approximate this sequence by running beam search and choosing

$$\tilde{\mathbf{y}} \approx \underset{\mathbf{t}\in\mathcal{T}_K}{\operatorname{argmax}}\, sim(\mathbf{t},\mathbf{y})$$

where $\mathcal{T}_K$ is the K-best list from beam search. We take sim to be smoothed sentence-level BLEU (Chen and Cherry, 2014).

We justify training on $\tilde{\mathbf{y}}$ from a knowledge distillation perspective with the following generative process: suppose that there is a true target sequence (which we do not observe) that is first generated from the underlying data distribution $\mathcal{D}$. And further suppose that the target sequence that we observe ($\mathbf{y}$) is a noisy version of the unobserved true sequence: i.e. (i) $\mathbf{t} \sim \mathcal{D}$, (ii) $\mathbf{y} \sim \epsilon(\mathbf{t})$, where $\epsilon(\mathbf{t})$ is, for example, a noise function that independently replaces each element in $\mathbf{t}$ with a random element in $\mathcal{V}$ with some small probability.[6] In such a case, ideally the student's distribution should match the mixture distribution,

$$\mathcal{D}_{\text{SEQ-Inter}} \sim (1-\alpha)\mathcal{D} + \alpha q(\mathbf{t}\,|\,\mathbf{s})$$

In this setting, due to the noise assumption, $\mathcal{D}$ now has significant probability mass around a neighborhood of $\mathbf{y}$ (not just at $\mathbf{y}$), and therefore the argmax of the mixture distribution is likely something other than $\mathbf{y}$ (the observed sequence) or $\hat{\mathbf{y}}$ (the output from beam search). We can see that $\tilde{\mathbf{y}}$ is a natural approximation to the argmax of this mixture distribution between $\mathcal{D}$ and $q(\mathbf{t}\,|\,\mathbf{s})$ for some α. We illustrate this framework in Figure 1 (right) and visualize the distribution over a real example in Figure 2.

4 Experimental Setup

To test out these approaches, we conduct two sets of NMT experiments: high resource (English $\rightarrow$ German) and low resource (Thai $\rightarrow$ English).

The **English-German** data comes from WMT 2014.[7] The training set has 4m sentences and we take newstest2012/newstest2013 as the dev set and newstest2014 as the test set. We keep the top 50k most frequent words, and replace the rest with UNK. The teacher model is a 4×1000 LSTM (as in Luong et al. (2015)) and we train two student models: 2×300 and 2×500. The **Thai-English** data comes from IWSLT 2015.[8] There are 90k sentences in the

[6] While we employ a simple (unrealistic) noise function for illustrative purposes, the generative story is quite plausible if we consider a more elaborate noise function which includes additional sources of noise such as phrase reordering, replacement of words with synonyms, etc. One could view translation having two sources of variance that should be modeled separately: variance due to the source sentence ($\mathbf{t} \sim \mathcal{D}$), and variance due to the individual translator ($\mathbf{y} \sim \epsilon(\mathbf{t})$).

[7] http://statmt.org/wmt14

[8] https://sites.google.com/site/iwsltevaluation2015/mt-track

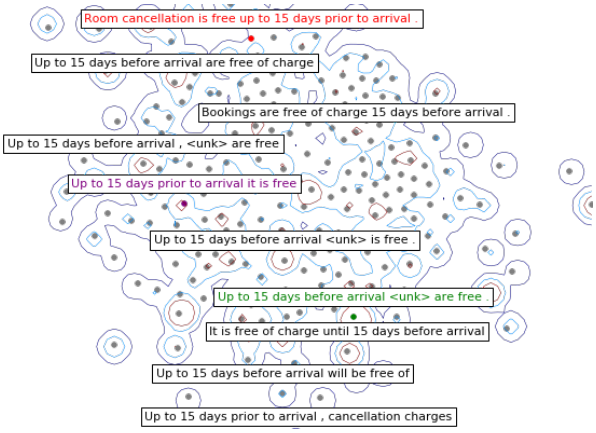


Figure 2: Visualization of sequence-level interpolation on an example German → English sentence: Bis 15 Tage vor Anreise sind Zimmer-Annullationen kostenlos. We run beam search, plot the final hidden state of the hypotheses using t-SNE and show the corresponding (smoothed) probabilities with contours. In the above example, the sentence that is at the top of the beam after beam search (green) is quite far away from gold (red), so we train the model on a sentence that is on the beam but had the highest sim (e.g. BLEU) to gold (purple).

training set and we take 2010/2011/2012 data as the dev set and 2012/2013 as the test set, with a vocabulary size is 25k. Size of the teacher model is 2×500 (which performed better than 4×1000, 2×750 models), and the student model is 2×100. Other training details mirror Luong et al. (2015).

We evaluate on tokenized BLEU with `multi-bleu.perl`, and experiment with the following variations:

Word-Level Knowledge Distillation (Word-KD) Student is trained on the original data and additionally trained to minimize the cross-entropy of the teacher distribution at the word-level. We tested $\alpha \in \{0.5, 0.9\}$ and found $\alpha = 0.5$ to work better.

Sequence-Level Knowledge Distillation (Seq-KD) Student is trained on the teacher-generated data, which is the result of running beam search and taking the highest-scoring sequence with the teacher model. We use beam size $K = 5$ (we did not see improvements with a larger beam).

Sequence-Level Interpolation (Seq-Inter) Student is trained on the sequence on the teacher's beam that had the highest BLEU (beam size $K = 35$). We adopt a fine-tuning approach where we begin training from a pretrained model (either on original data or Seq-KD data) and train with a smaller learning rate (0.1). For English-German we generate Seq-Inter data on a smaller portion of the training set ($\sim 50\%$) for efficiency.

The above methods are complementary and can be combined with each other. For example, we can train on teacher-generated data but still include a word-level cross-entropy term between the teacher/student (Seq-KD + Word-KD in Table 1), or fine-tune towards Seq-Inter data starting from the baseline model trained on original data (Baseline + Seq-Inter in Table 1).[9]

5 Results and Discussion

Results of our experiments are shown in Table 1. We find that while word-level knowledge distillation (Word-KD) does improve upon the baseline, sequence-level knowledge distillation (Seq-KD) does better on English → German and performs similarly on Thai → English. Combining them (Seq-KD + Word-KD) results in further gains for the 2×300 and 2×100 models (although not for the 2×500 model), indicating that these methods provide orthogonal means of transferring knowledge from the teacher to the student: Word-KD is transferring knowledge at the the local (i.e. word) level while Seq-KD is transferring knowledge at the global (i.e. sequence) level.

Sequence-level interpolation (Seq-Inter), in addition to improving models trained via Word-KD and Seq-KD, also improves upon the original teacher model that was trained on the actual data but fine-tuned towards Seq-Inter data (Baseline + Seq-Inter). In fact, greedy decoding with this fine-tuned model has similar performance (19.6) as beam search with the original model (19.5), allowing for faster decoding even with an identically-sized model.

We hypothesize that sequence-level knowledge distillation is effective because it allows the student network to only model relevant parts of the teacher distribution (i.e. around the teacher's mode) instead of 'wasting' parameters on trying to model the entire

[9]For instance, 'Seq-KD + Seq-Inter + Word-KD' in Table 1 means that the model was trained on Seq-KD data and fine-tuned towards Seq-Inter data with the mixture cross-entropy loss at the word-level.

Model	$\text{BLEU}_{K=1}$	$\Delta_{K=1}$	$\text{BLEU}_{K=5}$	$\Delta_{K=5}$	PPL	$p(\mathbf{t} = \hat{\mathbf{y}})$
English → German WMT 2014						
Teacher Baseline 4×1000 (Params: 221m)	17.7	–	19.5	–	6.7	1.3%
Baseline + Seq-Inter	19.6	+1.9	19.8	+0.3	10.4	8.2%
Student Baseline 2×500 (Params: 84m)	14.7	–	17.6	–	8.2	0.9%
Word-KD	15.4	+0.7	17.7	+0.1	8.0	1.0%
Seq-KD	18.9	**+4.2**	19.0	+1.4	22.7	16.9%
Baseline + Seq-Inter	18.5	+3.6	18.7	+1.1	11.3	5.7%
Word-KD + Seq-Inter	18.3	+3.6	18.5	+0.9	11.8	6.3%
Seq-KD + Seq-Inter	18.9	**+4.2**	19.3	**+1.7**	15.8	7.6%
Seq-KD + Word-KD	18.7	+4.0	18.9	+1.3	10.9	4.1%
Seq-KD + Seq-Inter + Word-KD	18.8	+4.1	19.2	+1.6	14.8	7.1%
Student Baseline 2×300 (Params: 49m)	14.1	–	16.9	–	10.3	0.6%
Word-KD	14.9	+0.8	17.6	+0.7	10.9	0.7%
Seq-KD	18.1	+4.0	18.1	+1.2	64.4	14.8%
Baseline + Seq-Inter	17.6	+3.5	17.9	+1.0	13.0	10.0%
Word-KD + Seq-Inter	17.8	+3.7	18.0	+1.1	14.5	4.3%
Seq-KD + Seq-Inter	18.2	+4.1	18.5	+1.6	40.8	5.6%
Seq-KD + Word-KD	17.9	+3.8	18.8	+1.9	44.1	3.1%
Seq-KD + Seq-Inter + Word-KD	18.5	**+4.4**	18.9	**+2.0**	97.1	5.9%
Thai → English IWSLT 2015						
Teacher Baseline 2×500 (Params: 47m)	14.3	–	15.7	–	22.9	2.3%
Baseline + Seq-Inter	15.6	+1.3	16.0	+0.3	55.1	6.8%
Student Baseline 2×100 (Params: 8m)	10.6	–	12.7	–	37.0	1.4%
Word-KD	11.8	+1.2	13.6	+0.9	35.3	1.4%
Seq-KD	12.8	+2.2	13.4	+0.7	125.4	6.9%
Baseline + Seq-Inter	12.9	+2.3	13.1	+0.4	52.8	2.5%
Word-KD + Seq-Inter	13.0	+2.4	13.7	+1.0	58.7	3.2%
Seq-KD + Seq-Inter	13.6	+3.0	14.0	+1.3	106.4	3.9%
Seq-KD + Word-KD	13.7	+3.1	14.2	+1.5	67.4	3.1%
Seq-KD + Seq-Inter + Word-KD	14.2	**+3.6**	14.4	**+1.7**	117.4	3.2%

Table 1: Results on English-German (newstest2014) and Thai-English (2012/2013) test sets. $\text{BLEU}_{K=1}$: BLEU score with beam size $K = 1$ (i.e. greedy decoding); $\Delta_{K=1}$: BLEU gain over the baseline model without any knowledge distillation with greedy decoding; $\text{BLEU}_{K=5}$: BLEU score with beam size $K = 5$; $\Delta_{K=5}$: BLEU gain over the baseline model without any knowledge distillation with beam size $K = 5$; PPL: perplexity on the test set; $p(\mathbf{t} = \hat{\mathbf{y}})$: Probability of output sequence from greedy decoding (averaged over the test set). Params: number of parameters in the model. Best results (as measured by improvement over the baseline) within each category are highlighted in bold.

space of translations. Our results suggest that this is indeed the case: the probability mass that Seq-KD models assign to the approximate mode is much higher than is the case for baseline models trained on original data (Table 1: $p(\mathbf{t} = \hat{\mathbf{y}})$). For example, on English → German the (approximate) argmax for the 2×500 Seq-KD model (on average) accounts for 16.9% of the total probability mass, while the corresponding number is 0.9% for the baseline. This also explains the success of greedy decoding for Seq-KD models—since we are only modeling around the teacher's mode, the student's distribution is more peaked and therefore the argmax is much easier to find. Seq-Inter offers a compromise between the two, with the greedily-decoded sequence accounting for 7.6% of the distribution.

Finally, although past work has shown that models with lower perplexity generally tend to have

Model Size	GPU	CPU	Android
Beam = 1 (Greedy)			
4×1000	425.5	15.0	–
2×500	1051.3	63.6	8.8
2×300	1267.8	104.3	15.8
Beam $= 5$			
4×1000	101.9	7.9	–
2×500	181.9	22.1	1.9
2×300	189.1	38.4	3.4

Table 2: Number of source words translated per second across GPU (GeForce GTX Titan X), CPU, and smartphone (Samsung Galaxy 6) for the various English $\rightarrow$ German models. We were unable to open the 4×1000 model on the smartphone.

Model	Prune %	Params	BLEU	Ratio
4×1000	0%	221 m	19.5	$1\times$
2×500	0%	84 m	19.3	$3\times$
2×500	50%	42 m	19.3	$5\times$
2×500	80%	17 m	19.1	$13\times$
2×500	85%	13 m	18.8	$18\times$
2×500	90%	8 m	18.5	$26\times$

Table 3: Performance of student models with varying % of the weights pruned. Top two rows are models without any pruning. Params: number of parameters in the model; Prune %: Percentage of weights pruned based on their absolute values; BLEU: BLEU score with beam search decoding ($K = 5$) after retraining the pruned model; Ratio: Ratio of the number of parameters versus the original teacher model (which has 221m parameters).

higher BLEU, our results indicate that this is not necessarily the case. The perplexity of the baseline 2×500 English $\rightarrow$ German model is 8.2 while the perplexity of the corresponding Seq-KD model is 22.7, despite the fact that Seq-KD model does significantly better for both greedy (+4.2 BLEU) and beam search (+1.4 BLEU) decoding.

5.1 Decoding Speed

Run-time complexity for beam search grows linearly with beam size. Therefore, the fact that sequence-level knowledge distillation allows for greedy decoding is significant, with practical implications for running NMT systems across various devices. To test the speed gains, we run the teacher/student models on GPU, CPU, and smartphone, and check the average number of source words translated per second (Table 2). We use a GeForce GTX Titan X for GPU and a Samsung Galaxy 6 smartphone. We find that we can run the student model 10 times faster with greedy decoding than the teacher model with beam search on GPU (1051.3 vs 101.9 words/sec), with similar performance.

5.2 Weight Pruning

Although knowledge distillation enables training faster models, the number of parameters for the student models is still somewhat large (Table 1: Params), due to the word embeddings which dominate most of the parameters.[10] For example, on the 2×500 English $\rightarrow$ German model the word embeddings account for approximately 63% (50m out of 84m) of the parameters. The size of word embeddings have little impact on run-time as the word embedding layer is a simple lookup table that only affects the first layer of the model.

We therefore focus next on reducing the memory footprint of the student models further through weight pruning. Weight pruning for NMT was recently investigated by See et al. (2016), who found that up to $80 - 90\%$ of the parameters in a large NMT model can be pruned with little loss in performance. We take our best English $\rightarrow$ German student model (2×500 Seq-KD + Seq-Inter) and prune $x\%$ of the parameters by removing the weights with the lowest absolute values. We then retrain the pruned model on Seq-KD data with a learning rate of 0.2 and fine-tune towards Seq-Inter data with a learning rate of 0.1. As observed by See et al. (2016), retraining proved to be crucial. The results are shown in Table 3.

Our findings suggest that compression benefits achieved through weight pruning and knowledge distillation are orthogonal.[11] Pruning 80% of the weight in the 2×500 student model results in a model with $13\times$ fewer parameters than the original teacher model with only a decrease of 0.4 BLEU. While pruning 90% of the weights results in a more appreciable decrease of 1.0 BLEU, the model is

[10] Word embeddings scale linearly while RNN parameters scale quadratically with the dimension size.

[11] To our knowledge combining pruning and knowledge distillation has not been investigated before.

drastically smaller with 8m parameters, which is $26\times$ fewer than the original teacher model.

5.3 Further Observations

- For models trained with word-level knowledge distillation, we also tried regressing the student network's top-most hidden layer at each time step to the teacher network's top-most hidden layer as a pretraining step, noting that Romero et al. (2015) obtained improvements with a similar technique on feed-forward models. We found this to give comparable results to standard knowledge distillation and hence did not pursue this further.

- There have been promising recent results on eliminating word embeddings completely and obtaining word representations directly from characters with character composition models, which have many fewer parameters than word embedding lookup tables (Ling et al., 2015a; Kim et al., 2016; Ling et al., 2015b; Jozefowicz et al., 2016; Costa-Jussa and Fonollosa, 2016). Combining such methods with knowledge distillation/pruning to further reduce the memory footprint of NMT systems remains an avenue for future work.

6 Related Work

Compressing deep learning models is an active area of current research. *Pruning* methods involve pruning weights or entire neurons/nodes based on some criterion. LeCun et al. (1990) prune weights based on an approximation of the Hessian, while Han et al. (2016) show that a simple magnitude-based pruning works well. Prior work on removing neurons/nodes include Srinivas and Babu (2015) and Mariet and Sra (2016). See et al. (2016) were the first to apply pruning to Neural Machine Translation, observing that that different parts of the architecture (input word embeddings, LSTM matrices, etc.) admit different levels of pruning. *Knowledge distillation* approaches train a smaller student model to mimic a larger teacher model, by minimizing the loss between the teacher/student predictions (Bucila et al., 2006; Ba and Caruana, 2014; Li et al., 2014; Hinton et al., 2015). Romero et al. (2015) additionally regress on the intermediate hidden layers of the student/teacher network as a pretraining step, while Mou et al. (2015) obtain smaller word embeddings from a teacher model via regression. There has also been work on transferring knowledge across different network architectures: Chan et al. (2015b) show that a deep non-recurrent neural network can learn from an RNN; Geras et al. (2016) train a CNN to mimic an LSTM for speech recognition. Kuncoro et al. (2016) recently investigated knowledge distillation for structured prediction by having a single parser learn from an ensemble of parsers.

Other approaches for compression involve low rank factorizations of weight matrices (Denton et al., 2014; Jaderberg et al., 2014; Lu et al., 2016; Prabhavalkar et al., 2016), sparsity-inducing regularizers (Murray and Chiang, 2015), binarization of weights (Courbariaux et al., 2016; Lin et al., 2016), and weight sharing (Chen et al., 2015; Han et al., 2016). Finally, although we have motivated sequence-level knowledge distillation in the context of training a smaller model, there are other techniques that train on a mixture of the model's predictions and the data, such as local updating (Liang et al., 2006), hope/fear training (Chiang, 2012), SEARN (Daumé III et al., 2009), DAgger (Ross et al., 2011), and minimum risk training (Och, 2003; Shen et al., 2016).

7 Conclusion

In this work we have investigated existing knowledge distillation methods for NMT (which work at the word-level) and introduced two *sequence-level* variants of knowledge distillation, which provide improvements over standard word-level knowledge distillation.

We have chosen to focus on translation as this domain has generally required the largest capacity deep learning models, but the sequence-to-sequence framework has been successfully applied to a wide range of tasks including parsing (Vinyals et al., 2015a), summarization (Rush et al., 2015), dialogue (Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2016), NER/POS-tagging (Gillick et al., 2016), image captioning (Vinyals et al., 2015b; Xu et al., 2015), video generation (Srivastava et al., 2015), and speech recognition (Chan et al., 2015a). We anticipate that methods described in this paper can be used to similarly train smaller models in other domains.

References

[Ba and Caruana2014] Lei Jimmy Ba and Rich Caruana. 2014. Do Deep Nets Really Need to be Deep? In *Proceedings of NIPS*.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

[Bucila et al.2006] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model Compression. In *Proceedings of KDD*.

[Chan et al.2015a] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2015a. Listen, Attend and Spell. *arXiv:1508.01211*.

[Chan et al.2015b] William Chan, Nan Rosemary Ke, and Ian Laner. 2015b. Transfering Knowledge from a RNN to a DNN. *arXiv:1504.01483*.

[Chen and Cherry2014] Boxing Chen and Colin Cherry. 2014. A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

[Chen et al.2015] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. 2015. Compressing Neural Networks with the Hashing Trick. In *Proceedings of ICML*.

[Chiang2012] David Chiang. 2012. Hope and Fear for Discriminative Training of Statistical Translation Models. In *JMLR*.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*.

[Costa-Jussa and Fonollosa2016] Marta R. Costa-Jussa and Jose A.R. Fonollosa. 2016. Character-based Neural Machine Translation. *arXiv:1603.00810*.

[Courbariaux et al.2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or -1. *arXiv:1602.02830*.

[Daumé III et al.2009] Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based Structured Prediction. *Machine Learning*.

[Denil et al.2013] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. 2013. Predicting Parameters in Deep Learning. In *Proceedings of NIPS*.

[Denton et al.2014] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting Linear Structure within Convolutional Neural Networks for Efficient Evaluation. In *Proceedings of NIPS*.

[Geras et al.2016] Krzysztof J. Geras, Abdel rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. 2016. Blending LSTMs into CNNs. In *Proceedings of ICLR Workshop*.

[Gillick et al.2016] Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual Language Processing from Bytes. In *Proceedings of NAACL*.

[Han et al.2016] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *Proceedings of ICLR*.

[He et al.2014] Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu. 2014. Reshaping Deep Neural Network for Fast Decoding by Node-Pruning. In *Proceedings of ICASSP*.

[Hinton et al.2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv:1503.0253*.

[Jaderberg et al.2014] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up Convolutional Neural Networks with Low Rank Expansions. In *BMCV*.

[Jozefowicz et al.2016] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *arXiv:1602.02410*.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of EMNLP*.

[Kim et al.2016] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. In *Proceedings of AAAI*.

[Kuncoro et al.2016] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of EMNLP*.

[LeCun et al.1990] Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal Brain Damage. In *Proceedings of NIPS*.

[Li et al.2014] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning Small-Size DNN with Output-Distribution-Based Criteria. In *Proceedings of INTERSPEECH*.

[Li et al.2016] Jiwei Li, Michael Galley, Chris Brockett, Jianfeg Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversational Models. In *Proceedings of NAACL 2016*.

[Liang et al.2006] Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006. An End-to-End Discriminative Approach to Machine Translation. In *Proceedings of COLING-ACL*.

[Lin et al.2016] Zhouhan Lin, Matthieu Coubariaux, Roland Memisevic, and Yoshua Bengio. 2016. Neural Networks with Few Multiplications. In *Proceedings of ICLR*.

[Ling et al.2015a] Wang Ling, Tiago Lui, Luis Marujo, Ramon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding Function in Form: Composition Character Models for Open Vocabulary Word Representation. In *Proceedings of EMNLP*.

[Ling et al.2015b] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based Neural Machine Translation. *arXiv:1511.04586*.

[Lu et al.2016] Zhiyun Lu, Vikas Sindhwani, and Tara N. Sainath. 2016. Learning Compact Recurrent Neural Networks. In *Proceedings of ICASSP*.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*.

[Mariet and Sra2016] Zelda Mariet and Suvrit Sra. 2016. Diversity Networks. In *Proceedings of ICLR*.

[Mou et al.2015] Lili Mou, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Distilling Word Embeddings: An Encoding Approach. *arXiv:1506.04488*.

[Murray and Chiang2015] Kenton Murray and David Chiang. 2015. Auto-sizing Neural Networks: With Applications to N-Gram Language Models. In *Proceedings of EMNLP*.

[Och2003] Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*.

[Papineni et al.2002] Kishore Papineni, Slim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of ICML*.

[Prabhavalkar et al.2016] Rohit Prabhavalkar, Ouais Alsharif, Antoine Bruguier, and Ian McGraw. 2016. On the Compression of Recurrent Neural Networks with an Application to LVCSR Acoustic Modeling for Embedded Speech Recognition. In *Proceedings of ICASSP*.

[Romero et al.2015] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. In *Proceedings of ICLR*.

[Ross et al.2011] Stephane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of AISTATS*.

[Rush et al.2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP*.

[See et al.2016] Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of Neural Machine Translation via Pruning. In *Proceedings of CoNLL*.

[Serban et al.2016] Iulian V. Serban, Allesandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of AAAI*.

[Shen et al.2016] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Masong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *Proceedings of ACL*.

[Srinivas and Babu2015] Suraj Srinivas and R. Venkatesh Babu. 2015. Data-free Parameter Pruning for Deep Neural Networks. *BMVC*.

[Srivastava et al.2015] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised Learning of Video Representations using LSTMs. *Proceedings of ICML*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*.

[Vinyals and Le2015] Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. In *Proceedings of ICML Deep Learning Workshop*.

[Vinyals et al.2015a] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slave Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a Foreign Language. In *Proceedings of NIPS*.

[Vinyals et al.2015b] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and Tell: A Neural Image Caption Generator. In *Proceedings of CVPR*.

[Xu et al.2015] Kelvin Xu, Jimma Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICML*.

[Zhou et al.2016] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. In *Proceedings of TACL*.

Controlling Output Length in Neural Encoder-Decoders

Yuta Kikuchi[1]* kikuchi@lr.pi.titech.ac.jp
Graham Neubig[2]† gneubig@cs.cmu.edu
Ryohei Sasano[1] sasano@pi.titech.ac.jp
Hiroya Takamura[1] takamura@pi.titech.ac.jp
Manabu Okumura[1] oku@pi.titecjh.ac.jp

[1]Tokyo Institute of Technology, Japan
[2]Carnegie Mellon University, USA

Abstract

Neural encoder-decoder models have shown great success in many sequence generation tasks. However, previous work has not investigated situations in which we would like to control the length of encoder-decoder outputs. This capability is crucial for applications such as text summarization, in which we have to generate concise summaries with a desired length. In this paper, we propose methods for controlling the output sequence length for neural encoder-decoder models: two decoding-based methods and two learning-based methods.[1] Results show that our learning-based methods have the capability to control length without degrading summary quality in a summarization task.

1 Introduction

Since its first use for machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), the encoder-decoder approach has demonstrated great success in many other sequence generation tasks including image caption generation (Vinyals et al., 2015b; Xu et al., 2015), parsing (Vinyals et al., 2015a), dialogue response generation (Li et al., 2016a; Serban et al., 2016) and sentence summarization (Rush et al., 2015; Chopra et al., 2016). In particular, in this paper we focus on sentence summarization, which as its name suggests, consists of generating shorter versions of sentences for applications such as document summarization (Nenkova and McKeown, 2011) or headline generation (Dorr et al., 2003). Recently, Rush et al. (2015) automatically constructed large training data for sentence summarization, and this has led to the rapid development of neural sentence summarization (NSS) or neural headline generation (NHG) models. There are already many studies that address this task (Nallapati et al., 2016; Ayana et al., 2016; Ranzato et al., 2015; Lopyrev, 2015; Gulcehre et al., 2016; Gu et al., 2016; Chopra et al., 2016).

One of the essential properties that text summarization systems should have is the ability to generate a summary with the desired length. Desired lengths of summaries strongly depends on the scene of use, such as the granularity of information the user wants to understand, or the monitor size of the device the user has. The length also depends on the amount of information contained in the given source document. Hence, in the traditional setting of text summarization, both the source document and the desired length of the summary will be given as input to a summarization system. However, methods for controlling the output sequence length of encoder-decoder models have not been investigated yet, despite their importance in these settings.

In this paper, we propose and investigate four methods for controlling the output sequence length for neural encoder-decoder models. The former two methods are decoding-based; they receive the desired length during the decoding process, and the training process is the same as standard encoder-decoder models. The latter two methods are

*Now at Preferred Networks.
† This work was done when the author was at the Nara Institute of Science and Technology.
[1]Available at https://github.com/kiyukuta/lencon.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1328–1338,
Austin, Texas, November 1-5, 2016.

learning-based; we modify the network architecture to receive the desired length as input.

In experiments, we show that the learning-based methods outperform the decoding-based methods for long (such as 50 or 75 byte) summaries. We also find that despite this additional length-control capability, the proposed methods remain competitive to existing methods on standard settings of the DUC2004 shared task-1.

2 Background

2.1 Related Work

Text summarization is one of the oldest fields of study in natural language processing, and many summarization methods have focused specifically on sentence compression or headline generation. Traditional approaches to this task focus on word deletion using rule-based (Dorr et al., 2003; Zajic et al., 2004) or statistical (Woodsend et al., 2010; Galanis and Androutsopoulos, 2010; Filippova and Strube, 2008; Filippova and Altun, 2013; Filippova et al., 2015) methods. There are also several studies of abstractive sentence summarization using syntactic transduction (Cohn and Lapata, 2008; Napoles et al., 2011) or taking a phrase-based statistical machine translation approach (Banko et al., 2000; Wubben et al., 2012; Cohn and Lapata, 2013).

Recent work has adopted techniques such as encoder-decoder (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) and attentional (Bahdanau et al., 2015; Luong et al., 2015) neural network models from the field of machine translation, and tailored them to the sentence summarization task. Rush et al. (2015) were the first to pose sentence summarization as a new target task for neural sequence-to-sequence learning. Several studies have used this task as one of the benchmarks of their neural sequence transduction methods (Ranzato et al., 2015; Lopyrev, 2015; Ayana et al., 2016). Some studies address the other important phenomena frequently occurred in human-written summaries, such as copying from the source document (Gu et al., 2016; Gulcehre et al., 2016). Nallapati et al. (2016) investigate a way to solve many important problems capturing keywords, or inputting multiple sentences.

Neural encoder-decoders can also be viewed as statistical language models conditioned on the target sentence context. Rosenfeld et al. (2001) have proposed whole-sentence language models that can consider features such as sentence length. However, as described in the introduction, to our knowledge, explicitly controlling length of output sequences in neural language models or encoder-decoders has not been investigated.

Finally, there are some studies to modify the output sequence according some meta information such as the dialogue act (Wen et al., 2015), user personality (Li et al., 2016b), or politeness (Sennrich et al., 2016). However, these studies have not focused on length, the topic of this paper.

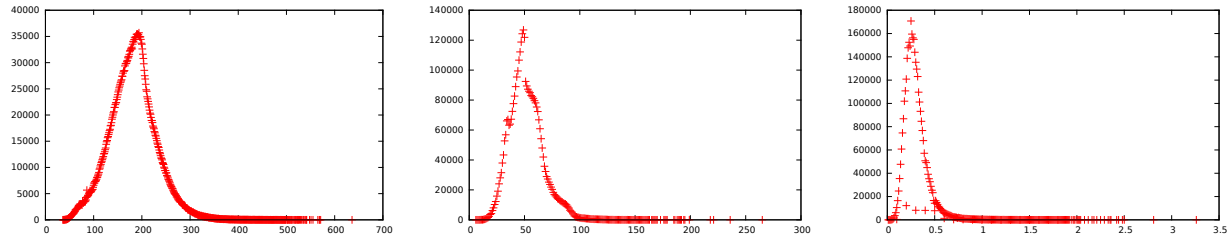
2.2 Importance of Controlling Output Length

As we already mentioned in Section 1, the most standard setting in text summarization is to input both the source document and the desired length of the summary to a summarization system. Summarization systems thus must be able to generate summaries of various lengths. Obviously, this property is also essential for summarization methods based on neural encoder-decoder models.

Since an encoder-decoder model is a completely data-driven approach, the output sequence length depends on the training data that the model is trained on. For example, we use sentence-summary pairs extracted from the Annotated English Gigaword corpus as training data (Rush et al., 2015), and the average length of human-written summary is 51.38 bytes. Figure 1 shows the statistics of the corpus. When we train a standard encoder-decoder model and perform the standard beam search decoding on the corpus, the average length of its output sequence is 38.02 byte.

However, there are other situations where we want summaries with other lengths. For example, DUC2004 is a shared task where the maximum length of summaries is set to 75 bytes, and summarization systems would benefit from generating sentences up to this length limit.

While recent NSS models themselves cannot control their output length, Rush et al. (2015) and others following use an ad-hoc method, in which the system is inhibited from generating the end-of-sentence (EOS) tag by assigning a score of $-\infty$ to the tag and



(a) first sentence (181.87) **(b)** article headline (51.38) **(c)** ratio (0.30)

Figure 1: Histograms of first sentence length, headline length, and their ratio in Annotated Gigaword English Gigaword corpus. Bracketed values in each subcaption are averages.

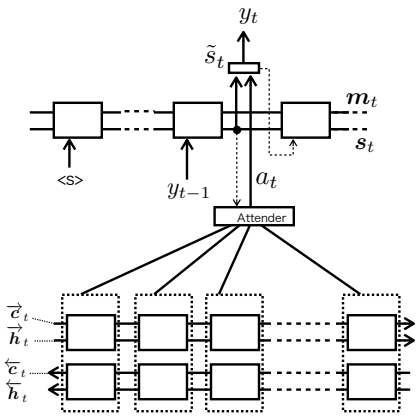


Figure 2: The encoder-decoder architecture we used as a base model in this paper.

generating a fixed number of words[2], and finally the output summaries are truncated to 75 bytes. Ideally, the models should be able to change the output sequence depending on the given output length, and to output the EOS tag at the appropriate time point in a natural manner.

3 Network Architecture: Encoder-Decoder with Attention

In this section, we describe the model architecture used for our experiments: an encoder-decoder consisting of bi-directional RNNs and an attention mechanism. Figure 2 shows the architecture of the model.

Suppose that the source sentence is represented as a sequence of words $\boldsymbol{x} = (x_1, x_2, x_3, ..., x_N)$. For a given source sentence, the summarizer generates a shortened version of the input (i.e. $N > M$), as summary sentence $\boldsymbol{y} = (y_1, y_2, y_3, ..., y_M)$. The model estimates conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ using parameters trained on large training data consisting of sentence-summary pairs. Typically, this conditional probability is factorized as the product of conditional probabilities of the next word in the sequence:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{M} p(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}),$$

where $\boldsymbol{y}_{<t} = (y_1, y_2, y_3, ..., y_{t-1})$. In the following, we describe how to compute $p(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$.

3.1 Encoder

We use the bi-directional RNN (BiRNN) as encoder which has been shown effective in neural machine translation (Bahdanau et al., 2015) and speech recognition (Schuster and Paliwal, 1997; Graves et al., 2013).

A BiRNN processes the source sentence for both forward and backward directions with two separate RNNs. During the encoding process, the BiRNN computes both forward hidden states $(\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, ..., \overrightarrow{\boldsymbol{h}}_N)$ and backward hidden states $(\overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, ..., \overleftarrow{\boldsymbol{h}}_N)$ as follows:

$$\overrightarrow{\boldsymbol{h}}_t = g(\overrightarrow{\boldsymbol{h}}_{t-1}, x_t),$$
$$\overleftarrow{\boldsymbol{h}}_t = g(\overleftarrow{\boldsymbol{h}}_{t+1}, x_t).$$

While g can be any kind of recurrent unit, we use long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks that have memory cells for both directions ($\overrightarrow{\boldsymbol{c}}_t$ and $\overleftarrow{\boldsymbol{c}}_t$).

[2] According to the published code (https://github.com/facebook/NAMAS), the default number of words is set to 15, which is too long for the DUC2004 setting. The average number of words of human summaries in the evaluation set is 10.43.

After encoding, we set the initial hidden states $\boldsymbol{s}_0$ and memory-cell $\boldsymbol{m}_0$ of the decoder as follows:

$$\begin{aligned} \boldsymbol{s}_0 &= \overleftarrow{\boldsymbol{h}}_1, \\ \boldsymbol{m}_0 &= \overleftarrow{\boldsymbol{c}}_1. \end{aligned}$$

3.2 Decoder and Attender

Our decoder is based on an RNN with LSTM g:

$$\boldsymbol{s}_t = g(\boldsymbol{s}_{t-1}, x_t).$$

We also use the attention mechanism developed by Luong et al. (2015), which uses $\boldsymbol{s}_t$ to compute contextual information $\boldsymbol{d}_t$ of time step t. We first summarize the forward and backward encoder states by taking their sum $\bar{\boldsymbol{h}}_i = \overrightarrow{\boldsymbol{h}}_i + \overleftarrow{\boldsymbol{h}}_i$, and then calculate the context vector $\boldsymbol{d}_t$ as the weighted sum of these summarized vectors:

$$\boldsymbol{d}_t = \sum_i \boldsymbol{a}_{ti} \bar{\boldsymbol{h}}_i,$$

where $\boldsymbol{a}_t$ is the weight at the t-th step for $\bar{\boldsymbol{h}}_i$ computed by a softmax operation:

$$a_{ti} = \frac{\exp(\boldsymbol{s}_t \cdot \bar{\boldsymbol{h}}_i)}{\sum_{\bar{\boldsymbol{h}}'} \exp(\boldsymbol{s}_t \cdot \bar{\boldsymbol{h}}')}.$$

After context vector $\boldsymbol{d}_t$ is calculated, the model updates the distribution over the next word as follows:

$$\begin{aligned} \tilde{\boldsymbol{s}}_t &= \tanh(W_{hs}[\boldsymbol{s}_t; \boldsymbol{d}_t] + \boldsymbol{b}_{hs}), \\ p(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}) &= \text{softmax}(W_{so}\tilde{\boldsymbol{s}}_t + \boldsymbol{b}_{so}). \end{aligned}$$

Note that $\tilde{\boldsymbol{s}}_t$ is also provided as input to the LSTM with y_t for the next step, which is called the *input feeding* architecture (Luong et al., 2015).

3.3 Training and Decoding

The training objective of our models is to maximize log likelihood of the sentence-summary pairs in a given training set D:

$$\begin{aligned} L_t(\theta) &= \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} \log p(\boldsymbol{y}|\boldsymbol{x}; \theta), \\ p(\boldsymbol{y}|\boldsymbol{x}; \theta) &= \prod_t p(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}). \end{aligned}$$

Once models are trained, we use beam search to find the output that maximizes the conditional probability.

4 Controlling Length in Encoder-decoders

In this section, we propose our four methods that can control the length of the output in the encoder-decoder framework. In the first two methods, the decoding process is used to control the output length without changing the model itself. In the other two methods, the model itself has been changed and is trained to obtain the capability of controlling the length. Following the evaluation dataset used in our experiments, we use bytes as the unit of length, although our models can use either words or bytes as necessary.

4.1 $fixLen$: Beam Search without EOS Tags

The first method we examine is a decoding approach similar to the one taken in many recent NSS methods that is slightly less ad-hoc. In this method, we inhibit the decoder from generating the EOS tag by assigning it a score of $-\infty$. Since the model cannot stop the decoding process by itself, we simply stop the decoding process when the length of output sequence reaches the desired length. More specifically, during beam search, when the length of the sequence generated so far exceeds the desired length, the last word is replaced with the EOS tag and also the score of the last word is replaced with the score of the EOS tag (*EOS replacement*).

4.2 $fixRng$: Discarding Out-of-range Sequences

Our second decoding method is based on discarding out-of-range sequences, and is not inhibited from generating the EOS tag, allowing it to decide when to stop generation. Instead, we define the legitimate range of the sequence by setting minimum and maximum lengths. Specifically, in addition to the normal beam search procedure, we set two rules:

- If the model generates the EOS tag when the output sequence is shorter than the minimum length, we discard the sequence from the beam.
- If the generated sequence exceeds the maximum length, we also discard the sequence from the beam. We then replace its last word with the EOS tag and add this sequence to the beam

(EOS replacement in Section 4.1).[3]

In other words, we keep only the sequences that contain the EOS tag and are in the defined length range. This method is a compromise that allows the model some flexibility to plan the generated sequences, but only within a certain acceptable length range.

It should be noted that this method needs a larger beam size if the desired length is very different from the average summary length in the training data, as it will need to preserve hypotheses that have the desired length.

4.3 *LenEmb*: Length Embedding as Additional Input for the LSTM

Our third method is a learning-based method specifically trained to control the length of the output sequence. Inspired by previous work that has demonstrated that additional inputs to decoder models can effectively control the characteristics of the output (Wen et al., 2015; Li et al., 2016b), this model provides information about the length in the form of an additional input to the net. Specifically, the model uses an embedding $e_2(l_t) \in \mathbb{R}^D$ for each potential desired length, which is parameterized by a length embedding matrix $W_{le} \in \mathbb{R}^{D \times L}$ where L is the number of length types. In the decoding process, we input the embedding of the *remaining length* l_t as additional input to the LSTM (Figure 3). l_t is initialized after the encoding process and updated during the decoding process as follows:

$$
\begin{aligned}
l_1 &= length, \\
l_{t+1} &= \begin{cases} 0 & (l_t - byte(y_t) \leq 0) \\ l_t - byte(y_t) & (\text{otherwise}), \end{cases}
\end{aligned}
$$

where $byte(y_t)$ is the length of output word y_t and $length$ is the desired length. We learn the values of the length embedding matrix W_{le} during training. This method provides additional information about the amount of length remaining in the output sequence, allowing the decoder to "plan" its output based on the remaining number of words it can generate.

[3]This is a workaround to prevent the situation in which all sequences are discarded from a beam.

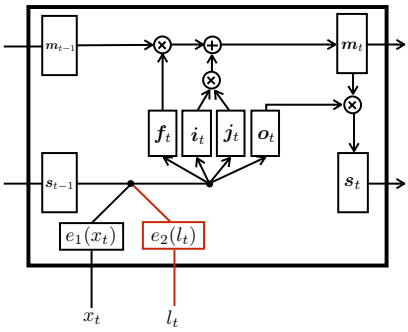


Figure 3: *LenEmb*: *remaining length* is used as additional input for the LSTM of the decoder.

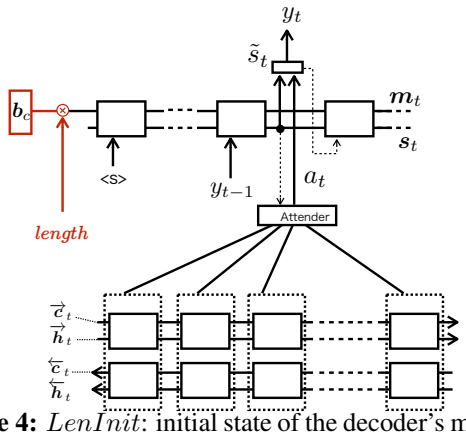


Figure 4: *LenInit*: initial state of the decoder's memory cell m_0 manages output length.

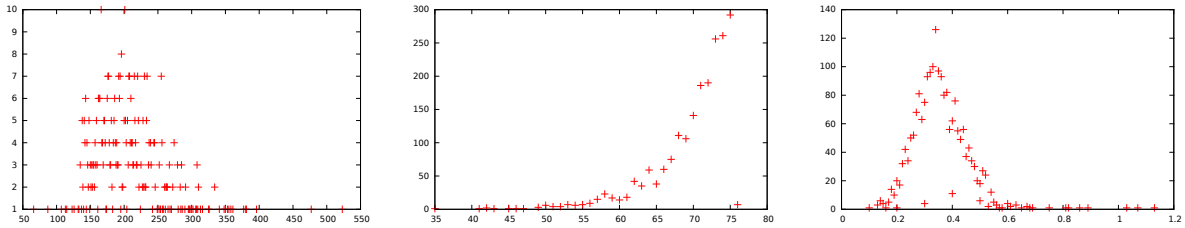
4.4 *LenInit*: Length-based Memory Cell Initialization

While *LenEmb* inputs the *remaining length* l_t to the decoder at each step of the decoding process, the *LenInit* method inputs the desired length once at the initial state of the decoder. Figure 4 shows the architecture of *LenInit*. Specifically, the model uses the memory cell $\boldsymbol{m}_t$ to control the output length by initializing the states of decoder (hidden state $\boldsymbol{s}_0$ and memory cell $\boldsymbol{m}_0$) as follows:

$$
\begin{aligned}
\boldsymbol{s}_0 &= \overleftarrow{\boldsymbol{h}}_1, \\
\boldsymbol{m}_0 &= \boldsymbol{b}_c * length, \qquad (1)
\end{aligned}
$$

where $\boldsymbol{b}_c \in \mathbb{R}^H$ is a trainable parameter and $length$ is the desired length.

While the model of *LenEmb* is guided towards the appropriate output length by inputting the remaining length at each step, this *LenInit* attempts to provide the model with the ability to manage the output length on its own using its inner state. Specifically, the memory cell of LSTM networks is suitable for this endeavour, as it is possible for LSTMs



(a) first sentence (206.91) **(b)** summary (70.00) **(c)** ratio (0.35)

Figure 5: Histograms of first sentence length, summary length, and their ratio in DUC2004.

to learn functions that, for example, subtract a fixed amount from a particular memory cell every time they output a word. Although other ways for managing the length are also possible,[4] we found this approach to be both simple and effective.

5 Experiment

5.1 Dataset

We trained our models on a part of the Annotated English Gigaword corpus (Napoles et al., 2012), which Rush et al. (2015) constructed for sentence summarization. We perform preprocessing using the standard script for the dataset[5]. The dataset consists of approximately 3.6 million pairs of the first sentence from each source document and its headline. Figure 1 shows the length histograms of the summaries in the training set. The vocabulary size is 116,875 for the source documents and 67,564 for the target summaries including the beginning-of-sentence, end-of-sentence, and unknown word tags. For $LenEmb$ and $LenInit$, we input the length of each headline during training. Note that we do not train multiple summarization models for each headline length, but a single model that is capable of controlling the length of its output.

We evaluate the methods on the evaluation set of DUC2004 task-1 (generating very short single-document summaries). In this task, summarization systems are required to create a very short summary for each given document. Summaries over the length limit (75 bytes) will be truncated and there is no bonus for creating a shorter summary. The evaluation set consists of 500 source documents and 4 human-written (reference) summaries for each source document. Figure 5 shows the length histograms of the summaries in the evaluation set. Note that the human-written summaries are not always as long as 75 bytes. We used three variants of ROUGE (Lin, 2004) as evaluation metrics: ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (longest common subsequence). The two-sided permutation test (Chinchor, 1992) was used for statistical significance testing ($p \leq 0.05$).

5.2 Implementation

We use Adam (Kingma and Ba, 2015) (α=0.001, β_1=0.9, β_2=0.999, eps=10^{-8}) to optimize parameters with a mini-batch of size 80. Before every 10,000 updates, we first sampled 800,000 training examples and made groups of 80 examples with the same source sentence length, and shuffled the 10,000 groups.

We set the dimension of word embeddings to 100 and that of the hidden state to 200. For LSTMs, we initialize the bias of the forget gate to 1.0 and use 0.0 for the other gate biases (Józefowicz et al., 2015). We use Chainer (Tokui et al., 2015) to implement our models. For $LenEmb$, we set L to 300, which is larger than the longest summary lengths in our dataset (see Figure 1-(b) and Figure 5-(b)).

For all methods except $fixRng$, we found a beam size of 10 to be sufficient, but for $fixRng$ we used a beam size of 30 because it more aggressively discards candidate sequences from its beams during decoding.

6 Result

6.1 ROUGE Evaluation

Table 1 shows the ROUGE scores of each method with various length limits (30, 50 and 75 byte). Regardless of the length limit set for the summariza-

[4]For example, we can also add another memory cell for managing the length.

[5]https://github.com/facebook/NAMAS

	30 byte			50 byte			75 byte		
model	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
$fixLen$	**14.34**	3.10*	**13.23**	20.00*	5.98	18.26*	25.87*	7.93	23.07*
$fixRng$	13.83*	3.08*	12.88	20.08*	5.74	18.19*	26.01	7.69*	22.77*
$LenEmb_{(0,L)}$	14.23	3.21	13.02	20.78	5.97	18.57	**26.73**	**8.39**	**23.88**
$LenInit_{(0,L)}$	14.31	3.27	13.19	**20.87**	6.16	19.00	25.87	8.27	23.24
$LenEmb_{(0,\infty)}$	13.75	3.30	12.68	20.62	**6.22**	18.64	26.42	8.26	23.59
$LenInit_{(0,\infty)}$	13.92	**3.49**	12.90	20.87	6.19	**19.09**	25.29*	8.00	22.71*

Table 1: ROUGE scores with various length limits. The scores with $*$ are significantly worse than the best score in the column (bolded).

source	five-time world champion michelle kwan withdrew from the #### us figure skating championships on wednesday , but will petition us skating officials for the chance to compete at the #### turin olympics .
reference	injury leaves kwan 's olympic hopes in limbo
$fixLen$ (30)	kwan withdraws from us gp
(50)	kwan withdraws from us skating championships
(75)	kwan pulls out of us figure skating championships for turin olympics
$fixRng$ (30)	kwan withdraws from us gp
(50)	kwan withdraws from figure skating championships
(75)	kwan pulls out of us figure skating championships for turin olympics bid
$LenEmb$ (30)	kwan withdraws from us skating
(50)	kwan withdraws from us figure skating championships
(75)	world champion kwan withdraws from #### olympic figure skating championships
$LenInit$ (30)	kwan quits us figure skating
(50)	kwan withdraws from #### us figure skating worlds
(75)	kwan withdraws from #### us figure skating championships for #### olympics

Table 2: Examples of the output of each method with various specified lengths.

tion methods, we use the same reference summaries. Note that, $fixLen$ and $fixRng$ generate the summaries with a hard constraint due to their decoding process, which allows them to follow the hard constraint on length. Hence, when we calculate the scores of $LenEmb$ and $LenInit$, we impose a hard constraint on length to make the comparison fair (i.e. $LenEmb_{(0,L)}$ and $LenInit_{(0,L)}$ in the table). Specifically, we use the same beam search as that for $fixRng$ with minimum length of 0.

For the purpose of showing the length control capability of $LenEmb$ and $LenInit$, we show at the bottom two lines the results of the standard beam search without the hard constraints on the length[6]. We will use the results of $LenEmb_{(0,\infty)}$ and $LenInit_{(0,\infty)}$ in the discussions in Sections 6.2 and 6.3.

The results show that the learning-based methods ($LenEmb$ and $LenInit$) tend to outperform decoding-based methods ($fixLen$ and $fixRng$) for the longer summaries of 50 and 75 bytes. However, in the 30-byte setting, there is no significant difference between these two types of methods. We hypothesize that this is because average compression rate in the training data is 30% (Figure 1-(c)) while the 30-byte setting forces the model to generate summaries with 15.38% in average compression rate, and thus the learning-based models did not have enough training data to learn compression at such a steep rate.

6.2 Examples of Generated Summaries

Tables 2 and 3 show examples from the validation set of the Annotated Gigaword Corpus. The tables show that all models, including both learning-based methods and decoding-based methods, can often generate well-formed sentences.

We can see various paraphrases of "#### us figure

[6] $fixRng$ is equivalence to the standard beam search when we set the range as $(0, \infty)$.

source	at least two people have tested positive for the bird flu virus in eastern turkey , health minister recep akdag told a news conference wednesday .
reference	two test positive for bird flu virus in turkey
$fixLen$ (30)	two infected with bird flu
(50)	two infected with bird flu in eastern turkey
(75)	two people tested positive for bird flu in eastern turkey says minister
$fixRng$ (30)	two infected with bird flu
(50)	two more infected with bird flu in eastern turkey
(75)	two people tested positive for bird flu in eastern turkey says minister
$LenEmb$ (30)	two bird flu cases in turkey
(50)	two confirmed positive for bird flu in eastern turkey
(75)	at least two bird flu patients test positive for bird flu in eastern turkey
$LenInit$ (30)	two cases of bird flu in turkey
(50)	two people tested positive for bird flu in turkey
(75)	two people tested positive for bird flu in eastern turkey health conference

Table 3: More examples of the output of each method.

championships"[7] and "withdrew". Some examples are generated as a single noun phrase ($LenEmb$(30) and $LenInit$(30)) which may be suitable for the short length setting.

6.3 Length Control Capability of Learning-based Models

Figure 6 shows histograms of output length from the standard encoder-decoder, $LenEmb$, and $LenInit$. While the output lengths from the standard model disperse widely, the lengths from our learning-based models are concentrated to the desired length. These histograms clearly show the length controlling capability of our learning-based models.

Table 4-(a) shows the final state of the beam when $LenInit$ generates the sentence with a length of 30 bytes for the example with standard beam search in Table 3. We can see all the sentences in the beam are generated with length close to the desired length. This shows that our method has obtained the ability to control the output length as expected. For comparison, Table 4-(b) shows the final state of the beam if we perform standard beam search in the standard encoder-decoder model (used in $fixLen$ and $fixRng$). Although each sentence is well-formed, the lengths of them are much more varied.

6.4 Comparison with Existing Methods

Finally, we compare our methods to existing methods on standard settings of the DUC2004 shared task-1. Although the objective of this paper is not to obtain state-of-the-art scores on this evaluation set, it is of interest whether our length-controllable models are competitive on this task. Table 5 shows that the scores of our methods, which are copied from Table 1, in addition to the scores of some existing methods. ABS (Rush et al., 2015) is the most standard model of neural sentence summarization and is the most similar method to our baseline setting ($fixLen$). This table shows that the score of $fixLen$ is comparable to those of the existing methods. The table also shows the $LenEmb$ and the $LenInit$ have the capability of controlling the length without decreasing the ROUGE score.

7 Conclusion

In this paper, we presented the first examination of the problem of controlling length in neural encoder-decoder models, from the point of view of summarization. We examined methods for controlling length of output sequences: two decoding-based methods ($fixLen$ and $fixRng$) and two learning-based methods ($LenEmb$ and $LenInit$). The results showed that learning-based methods generally outperform the decoding-based methods, and the learning-based methods obtained the capability of controlling the output length without losing ROUGE score compared to existing summarization methods.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP26280080. We are grateful to have the

[7] Note that "#" is a normalized number and "us" is "US" (United States).

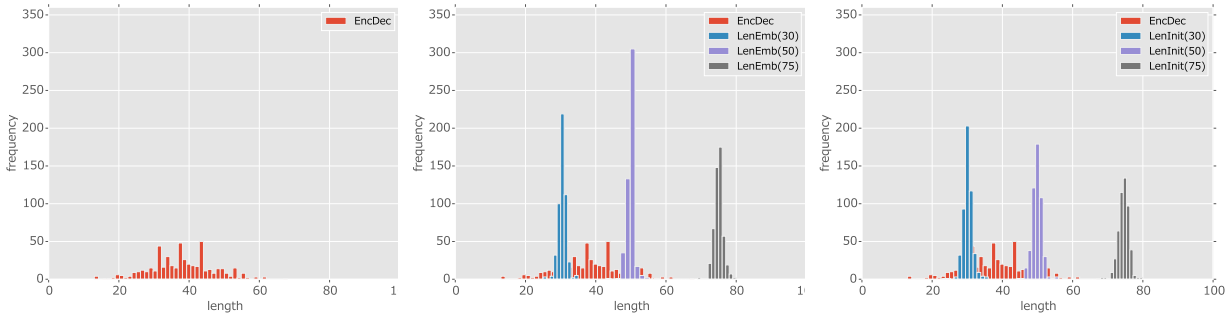
$\log p(\boldsymbol{y}\|\boldsymbol{x})$	byte	candidate summary
-4.27	31	two cases of bird flu in turkey
-4.41	28	two bird flu cases in turkey
-4.65	30	two people tested for bird flu
-5.25	30	two people tested in e. turkey
-5.27	31	two bird flu cases in e. turkey
-5.51	29	two bird flu cases in eastern
-5.55	32	two people tested in east turkey
-5.72	30	two bird flu cases in turkey :
-6.04	30	two people fail bird flu virus

(a) the beam of $LenInit$

$\log p(\boldsymbol{y}\|\boldsymbol{x})$	byte	candidate summary
-5.05	57	two people tested positive for bird flu in eastern turkey
-5.13	50	two tested positive for bird flu in eastern turkey
-5.30	39	two people tested positive for bird flu
-5.49	51	two people infected with bird flu in eastern turkey
-5.52	32	two tested positive for bird flu
-5.55	44	two infected with bird flu in eastern turkey
-6.00	49	two more infected with bird flu in eastern turkey
-6.04	54	two more confirmed cases of bird flu in eastern turkey
-6.50	49	two people tested positive for bird flu in turkey

(b) the beam of the standard encoder-decoder

Table 4: Final state of the beam when the learning-based model is instructed to output a 30 byte summary for the source document in Table 3.



(a) encoder-decoder **(b)** $LenEmb$ **(c)** $LenInit$

Figure 6: Histograms of output lengths generated by (a) the standard encoder-decoder , (b) $LenEmb$, and (c) $LenInit$. For $LenEmb$ and $LenInit$, the bracketed numbers in each region are the desired lengths we set.

model	R-1	R-2	R-L
$fixLen$	25.88	7.93	23.07
$fixRng$	26.02	7.69	22.78
$LenEmb$	26.73	8.40	23.88
$LenInit$	25.87	8.28	23.25
ABS(Rush et al., 2015)	26.55	7.06	22.05
ABS+(Rush et al., 2015)	28.18	**8.49**	23.81
RAS-Elman(Chopra et al., 2016)	**28.97**	8.26	**24.06**
RAS-LSTM(Chopra et al., 2016)	27.41	7.69	23.06

Table 5: Comparison with existing studies for DUC2004. Note that top four rows are reproduced from Table 1.

opportunity to use the Kurisu server of Dwango Co., Ltd. for our experiments.

References

Ayana, S. Shen, Z. Liu, and M. Sun. 2016. Neural Headline Generation with Minimum Risk Training. *CoRR*, abs/1604.01904.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR15*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of ACL00*, pages 318–325.

Nancy Chinchor. 1992. The statistical significance of the muc-4 results. In *Proceedings MUC4 '92*, pages 30–50.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the EMNLP14*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL-HLT16*, pages 93–98.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING08*, pages 137–144.

Trevor Cohn and Mirella Lapata. 2013. An abstrac-

tive approach to sentence compression. *ACM TIST13*, 4(3):41:1–41:35, July.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP13*, pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of INLG08*, pages 25–32.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of EMNLP15*, pages 360–368.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of NAACL-HLT10*, pages 885–893.

A. Graves, N. Jaitly, and A. r. Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Proceedings of IEEE Workshop on ASRU13*, pages 273–278.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL16*, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of ACL16*, pages 140–149.

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

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML15*, pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP13*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR15*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT16*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of ACL16*, pages 994–1003.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL04 Workshop*, pages 74–81.

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *CoRR*, abs/1512.01712.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP15*, pages 1412–1421.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. In *Foundations and Trends® in Information Retrieval*, volume 2-3, pages 103–233.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech & Language*, 15(1):55–73.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP15*, pages 379–389.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of NAACL-HLT16*, pages 35–40.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI16*, pages 3776–3784.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS14*, pages 3104–3112.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of NIPS15 Workshop on LearningSys*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015a. Grammar as a foreign language. In *Proceedings of NIPS15*, pages 2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP15*, pages 1711–1721, Lisbon, Portugal, September. Association for Computational Linguistics.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the EMNLP10*, pages 513–523.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of ACL12*, pages 1015–1024.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of ICML15*, pages 2048–2057. JMLR Workshop and Conference Proceedings.

David Zajic, Bonnie J Dorr, and R. Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of NAACL-HLT04 Document Understanding Workshop*, pages 112 – 119.

Poet Admits // Mute Cypher: Beam Search to find Mutually Enciphering Poetic Texts

Cole Peterson and **Alona Fyshe**
University of Victoria
cpeterso@uvic.ca, afyshe@uvic.ca

Abstract

The Xenotext Experiment implants poetry into an extremophile's DNA, and uses that DNA to generate new poetry in a protein form. The molecular machinery of life requires that these two poems encipher each other under a symmetric substitution cipher. We search for ciphers which permit writing under the Xenotext constraints, incorporating ideas from cipher-cracking algorithms, and using n-gram data to assess a cipher's "writability". Our algorithm, Beam Verse, is a beam search which uses new heuristics to navigate the cipher-space. We find thousands of ciphers which score higher than successful ciphers used to write Xenotext constrained texts.

1 Introduction

For over a decade, poet Christian Bök has been working on The Xenotext (Bök, 2008), a literary experiment which aims to insert poetry into the DNA of an extremophile organism. In contrast to popular modern data storage mediums like paper and hard disks, DNA is more robust to accidents, and requires minimal maintenance to last hundreds or even thousands of years (Cox, 2001). Many groups are actively pursuing efficient and stable ways to use DNA to encode information (Shimanovsky et al., 2002; Goldman et al., 2013). With his project, Bök aims to leave a lasting cultural contribution inside of an organism, which, as a result of DNA's durability and self-replicating properties, could conceivably survive longer than all other existing works of art.

Furthermore, Bök aims to craft his poem so the protein it instructs the cell to produce is yet another English poem. In a sense, Bök not only turns a microbe into a genetic book, he also engineers the microbe to be, in a sense, a poet. The organism's molecular machinery powers this translation between the two texts, which, at a high level, is a symmetric substitution cipher between the letters, and is described in more detail in Section 2. The two poems (the poet's and the organism's) must both play by the rules we refer to as the Xenotext Constraints:

- Each text is valid natural language.
- The substitution cipher function applied to one text, results in the other text, and vice versa. In other words, the cipher function must be symmetric.
- Whitespace characters (space, new line) encipher themselves, and are the only characters allowed identity mappings.

After four years of work, Bök successfully wrote two English poems which satisfied the Xenotext constraints[1], becoming the first person to do so. The first challenge was finding a cipher which allows for two valid English texts to be written. Finding "writable" ciphers is difficult and is the focus of this paper.

We present Beam Verse, a language-agnostic algorithm driven by the target language's n-gram data, that searches for "writable" ciphers, and suggests words and phrases which can be written under them.

[1]The two poems used the cipher $\binom{\texttt{abcdefghijlmq}}{\texttt{tvukyspnoxrwz}}$

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1339–1347,
Austin, Texas, November 1-5, 2016.

We do not concern ourselves with the full biochemical constraints of The Xenotext (eg. the actual impact of the protein and the cell's reaction to it, or its viability after folding) and instead only consider the Xenotext constraints listed above. This problem sits at the intersection of natural language processing and cryptography, and is a prerequisite to the genetic engineering required to fully realize a living xenotext. Our algorithm uncovers new ciphers which make satisfying the Xenotext constraints in English possible, and makes it easier for a poet of any language to investigate the feasibility of writing two mutually enciphering texts in their own tongue.

2 Genetic Background

DNA provides instructions to a living cell to produce a protein. DNA has a double helix structure (reminiscent of a twisted ladder) and is made up of four nucleotides: adenine, cytosine, guanine, and thymine, commonly abbreviated as A, T, C, and G. Each nucleotide always appears paired with another across the "rung" of the ladder, A with T, C with G, and vice versa. To transfer the data in the DNA to the protein-producing ribosome, the double helix is "unzipped", separating the ladder at the rungs, and a copy of the exposed DNA sequence called an *mRNA transcript* is synthesized, pairing in the same way the DNA did, with the exception that adenine in the DNA strand pairs with uracil (U) in the mRNA. The ribosome reads the mRNA as instructions to produce a specific protein. A protein is a sequence of amino acids and each triplet of nucleotides in the mRNA (called a *codon*) represents one of the twenty amino acids (see Table 2) (Campbell and Reece, 2008).

We can write in the DNA of an organism by having a codon represent a letter. When this sequence of codons (the full poem) is read by the organism, it then writes a sequence of amino acids, each of which represent a letter, in reply[2]. The letters in each poem have a bijective relationship determined by the biochemical processes that link them. For example, as shown in Table 2, the letters E and T are mutually linked, as wherever the poet uses the letter E, the cell uses the letter T, and vice versa. If the poet was to write "mute" instead of "poet", the cell would write "poet" instead of "mute".

Poet's Letter	P	O	E	T
DNA Codon	AAC	GAG	CCG	GGC
mRNA Codon	UUG	CUC	GGC	CCG
Amino Acid	phenylalanine	leucine	glycine	proline
Cell's Letter	M	U	T	E

Table 1: Sample translation from text through DNA to new text

This view of DNA is extremely simplistic, and serves only to motivate and provide context for the rest of this paper. When using a codon to represent a poet's letter and an amino acid to represent one of the organism's letters, many complexities arise which add further constraints to the text, which we ignore in the remainder of the paper. When actually inserting his poetry into a cell, Bök struggled to get the organism to express the correct protein, because he failed to account for the full complexity of the cell and caused the cell to "censor" itself (Wershler, 2012). However, we consider these additional constraints to be future work.

3 Substitution Ciphers

Substitution ciphers are among the earliest known forms of cryptography, having existed since at least the days of Caesar (Sinkov, 1966). They work by replacing one letter in the plaintext with another to form the ciphertext. However, they are never used in modern cryptography because, despite a large keyspace, substitution ciphers do not change the letter frequency between plaintext and ciphertext. When properties of the plaintext are known (like the language it is written in), letter or word frequency data from that language can be used to quickly crack the cipher and uncover the key.

Every word has a deterministic encryption under a cipher. The encrypted word could be nonsense, or it could be another word. The word "poet", for example, can encrypt to many other words, including "mute". The "poet↔mute" word-pair forms what we call a *partial cipher*, and notate as $\binom{\texttt{poe}}{\texttt{mut}}$. We

[2]This makes the writing lipogrammatic, as there are only 20 amino acids, and one of them (Serine) must be used as the space character. Serine is used for the space because it is the only amino acid to encipher itself, as both codons AGT and TCA produce it, mirroring the constraint that the space is mapped to itself in our ciphers.

say this partial cipher has a *cardinality* of three, as it defines three letter pairings. A *full cipher* in a 26-letter language has a cardinality of 13. We also refer to $\binom{\texttt{poe}}{\texttt{mut}}$ as a *primary* cipher, because it is the lowest cardinality cipher to contain the word-pairing "poet↔mute".

As no characters except whitespace are allowed identity mappings a word-pair like "eat↔cat" is not valid, as both *a* and *t* would have to map to them selves. The symmetric Xenotext constraint prohibits "admits" from pairing with "cipher", as the letter *i* would require a mapping to both *d* and *h*. However, "admits" can pair with the alternative spelling "cypher", forming the primary cipher $\binom{\texttt{admits}}{\texttt{cypher}}$. We can combine this cipher with $\binom{\texttt{poe}}{\texttt{mut}}$, as none of the letter pairs conflict with each other – they are *compatible* with each other. Together, they form $\binom{\texttt{poeadis}}{\texttt{mutcyhr}}$. As the letter-pairs in $\binom{\texttt{poe}}{\texttt{mut}}$ and $\binom{\texttt{admits}}{\texttt{cypher}}$ are subsets of the letter-pairs in $\binom{\texttt{poeadis}}{\texttt{mutcyhr}}$, we call $\binom{\texttt{poe}}{\texttt{mut}}$ and $\binom{\texttt{admits}}{\texttt{cypher}}$ *subciphers* of $\binom{\texttt{poeadis}}{\texttt{mutcyhr}}$, and say that $\binom{\texttt{poeadis}}{\texttt{mutcyhr}}$ *extends* $\binom{\texttt{poe}}{\texttt{mut}}$ and $\binom{\texttt{admits}}{\texttt{cypher}}$. For any two ciphers ϕ_1 and ϕ_2, we use the notation $\phi_1 \subset \phi_2$ to denote that ϕ_1 is a subcipher of ϕ_2.

If we applied $\binom{\texttt{poeadis}}{\texttt{mutcyhr}}$ to "Poet Admits" (the first part of this paper's title), it would result "Mute Cypher" (the second part of the paper's title). The title refers to the difficulty of writing under the Xenotext constraint, as it is hard to find a cipher where writing is possible, most of the ciphers are mute. Once all of the possible word pairs of a target language have been discovered (Section 7) the challenge becomes navigating the tradeoffs of including a letter pair, as each letter pair eliminates the possibility of using some word-pairs, while including other word-pairs.

If a language has an odd number of characters a symmetric substitution cipher is not possible using every letter. We must decide which letter to leave out of our texts. This is accomplished by inserting a null letter (which appears nowhere in the language) into our letter set, thus giving the language an even number of characters. At the conclusion of Beam Verse the letter paired with null is the character to leave out.

4 Scoring a Cipher's "Writability"

When scoring a cipher, an important consideration is what makes one cipher more "writable" than another. We might score a cipher on the number of valid words under it, as having more words at your disposal makes it easier to write, but this is not necessarily so if all the words are all rare and useless. To combat this, we weight words based upon their frequency in language, so that better, more frequent words contribute more to a ciphers overall score. This values highly frequent and syntactically important words, like "the" or "and", while also allowing a large number of infrequent words to also contribute significantly to the score. However, a word's usefulness when writing mutually enciphering texts is explicitly tied to its sister word under the cipher. "The" looses its usefulness if every time it is used in one poem, a less frequent word like "nag' must be used in the other. We propose that since a word pair is only as good as its weakest word, that ciphers be scored by taking the sum of all available word pairs, counting the minimum frequency of the two words. This means that the word pair "the↔nag" would count the frequency of "nag".

Multiple different word pairings can form the same primary cipher. For example, $\binom{\texttt{th}}{\texttt{ea}}$ is formed by both "the↔eat" and "he↔at", and would count the score of both word-pairs. As there are always less or equal primary ciphers than word-pairs, it is more memory efficient to store the score of all the primary ciphers than to store the scores of all the word-pairs. We count the score of a primary cipher ϕ_p towards a full cipher ϕ_f if it is a subcipher of ϕ_f. Formally, if P is the set of all primary ciphers and $\phi_p \in P$, the score of ϕ_f is $\sum_{\phi_p \subset \phi_f} score(\phi_p)$.

Alternatively, this could be expressed as a dot product between a vector where every element is the score of a primary (the score vector, s), and a vector indicating whether a primary cipher is a subcipher (the heuristic vector, h), as seen in equation 1. In section 8 we show how h can be calculated efficiently, and also how it can be use to provide an upper and lower bound the score of a full cipher extended from a partial cipher.

$$score = s \cdot h \tag{1}$$

The concept of a word-pair can easily be extended

to include longer word-level n-grams. Like words, every n-gram either enciphers to nonsense or has a sister n-gram it is locked to under a cipher. All n-gram pairs also have an associated frequency in the language, and so can contribute to the score of a cipher in the same way as single words do: by the minimum of the two n-gram's frequency counting as the weight of the pair. Using n-grams also indirectly captures some syntactic structure, and allows for generation of sample phrase and sentence generation from the cipher by chaining together n-grams. These small phrases can be used to quickly prototype poetry. For our word list and frequency data, we use Google's n-grams (Michel et al., 2011), but any dataset could be used, and would give different ciphers depending on the data's source.

5 Graphical and Combinatoric Representation

There are 7,905,853,580,625 possible symmetric substitution ciphers in a 26 letter language like English. Even with the efficient means of scoring ciphers shown in section 8 (which can calculate a full cipher's score in $\sim$ 300 microseconds) the brute force solution would take over 75 years of computing time. To avoid this expensive full calculation, we formulate the problem as a graph of partial ciphers and use beam search to navigate the graph to high valued full solutions. We regret that the smallest non-trivial graph (of a 6 letter language) is too large to be included here; it requires 75 nodes arranged in three layers which takes up an entire page, but it can be found on our website[3]. An incomplete graph is shown in Figure 1. As we search the cipher space we trace edges up through the graph to a full cipher solution.

The size of the m^{th} layer of a n letter language is defined by equations 2-4. The counts for a 26-letter language and a derivation of this equation can be seen on our website.

$$f(m,0) = 1 \quad (2)$$

$$f(1,n) = n \times (n-1)/2 \quad (3)$$

$$f(m,n) = f(m-1,n) \times f(1, n-2\times(m-1))/m \quad (4)$$

[3] http://www.langlearnlab.cs.uvic.ca/beamverse

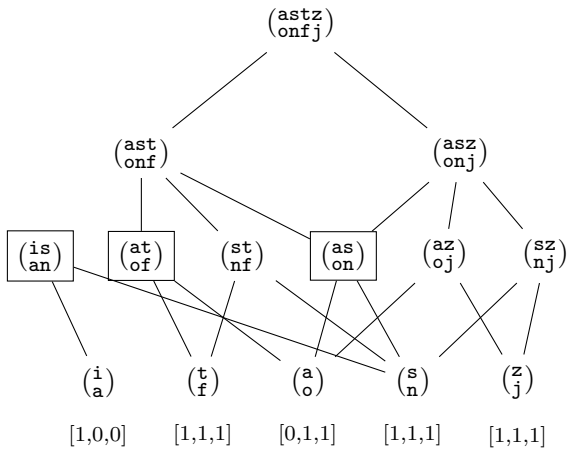


Figure 1: An incomplete cipher-graph, showing some of the partial ciphers which descend from $\binom{\text{astz}}{\text{onfj}}$. Three primary ciphers are shown in boxes, and are the same example primary ciphers used in Section 8. Compatibility vectors (discussed in Section 8.1) are shown for every letter-pair. Edges in this graph represent a difference of a letter-pair between ciphers. Each cardinality of partial cipher has a layer in the graph, which is our beam in the search.

6 Beam Search

A beam search algorithm does not require us to store all of this graph in memory, as we only examine a subset of the ciphers anticipated to be the best. Beam search works by enumerating all possibilities for one step of a solution (one layer of the cipher graph), sorting those options by a heuristic, keeping the best n partial solutions, and pruning the rest (Edelkamp and Schroedl, 2011). We fully expand one step further on the best n partial solutions, repeating the pruning and expanding process until a set of full solutions are reached. Beam search can effectively crack substitution ciphers, as shown by Nuhn et al. (2013).

A static beam size could be used (keeping the best thousand partial ciphers at each step, for example), however, the lower cardinality the partial, the more possibilities it generates. Every cardinality-1 partial cipher generates 276 possible cardinality-2 partial ciphers, whereas a cardinality-12 partial cipher only generates one possible full cipher (as there are only two unpaired letters remaining, therefore they must pair together). A constant beam size will limit the algorithm's performance in later stages of the search

if this is not accounted for.

We can rearrange our recursive definition in Equations 2 to 4 to determine the beam size which will generate exactly as many partial ciphers as we can hold in memory. If we want to generate B ciphers in an n letter language, and are at layer m, we should prune the beam to $b(m, n)$. This can be found by replacing replacing $f(m, n)$ for B, and $f(m-1, n)$ for $b(m, n)$ in equation 4 and rearranging to produce equation 5, removing m from equation 4 because we cannot assume duplicates will be generated.

$$b(m, n) = \frac{B}{f(1, n - 2 \times (m-1))} \tag{5}$$

7 Generating Primary Ciphers

In order to generate the primary ciphers, we must find all words which can encipher with each other, and record their score. Rather than checking every word against every other word, many useless checks can be avoided by hashing each word or n-gram according to the pattern of its letters, a concept which Hauer et al. (2014) called "pattern-equivalence" and used to crack substitution ciphers. We use a key which represents each letter in the n-gram with a number or symbol, ordered by first occurrence of the letter, while maintaining whitespace (eg. "and we are" $\rightarrow$ "123 45 165"). Another trigram like "his or her" would also generate the same key, and so the two trigrams would be checked against each other to see if they validly encipher, which they do, forming a primary partial cipher $\binom{\texttt{andwr}}{\texttt{hisoe}}$.

A match on the hash key does not guarantee that the words form a valid pairing. Many words which share a key form invalid word-pairings due to the symmetric or identity restrictions of the Xenotext constraint (eg. "cat↔eat", which share the key "123", or "admits↔cipher", which share the key "123456" are both invalid word-pairings). The score of a primary cipher is the sum of the score of all word-pairs which generate the primary. The algorithm is shown in Algorithm 1.

Algorithm 1 Generating Primary Ciphers

```
function GENERATE PRIMARIES(ngrams)
  for word_1 ← ngrams do
    key ← pattern(word_1)
    for word_2 ← patternDict[key] do
      if mutually-encipher(word, word_2) then
        primaries[encipher(word_1, word_2)] +=
  minScore(word_1, word_2)
      end if
    end for
    patternDict[key].add(word_1)
  end for
  return primaries
end function
```

8 Beam Search Heuristics

Recall from Section 3 that, in a full cipher, all letters have a defined mapping (the cipher has a cardinality of 13), while in a partial cipher some letters have undefined defined mappings, and that a primary cipher is the minimal cardinality partial cipher to contain a particular word-pair and is the building block of a cipher's score. We explore three different heuristics which calculate the true score for full ciphers, and estimate the score of full ciphers extended from a partial cipher by forming an upper and lower bound. All heuristics produce a vector h, which forms the score for a cipher when dotted with the score vector s (Equation 1). For full ciphers this vector will be the same regardless of the heuristic used, and the score from Equation 1 will be the true score of the full cipher, whereas different heuristics will give different values for a partial cipher, and thus guide Beam Verse in different directions. We implement these heuristics using bitwise operations, making them both memory and CPU efficient.

To demonstrate the calculation of the heuristics, we use the following primary ciphers as a running example, $\binom{\texttt{is}}{\texttt{an}}$ (score: 100), $\binom{\texttt{at}}{\texttt{of}}$ (score: 82), and $\binom{\texttt{on}}{\texttt{as}}$ (score: 76), which are the same primary ciphers as are shown in Figure 1. These three primaries would form the score vector, which is shared amongst all heuristics, is $s = [100, 82, 76]$. Thus $P = \{\binom{\texttt{is}}{\texttt{an}}, \binom{\texttt{at}}{\texttt{of}}, \binom{\texttt{on}}{\texttt{as}}\}$, and $|P| = 3$. We show the heuristic calculation for all three heuristics on the partial cipher $\binom{\texttt{asz}}{\texttt{onj}}$.

8.1 Upper Bound: Compatibility Vector

Recall that two ciphers are compatible if they have no conflicting letter-pairs. If two ciphers are compatible, it is possible to combine them. Every cipher ϕ has a vector representing compatibility with the

primary ciphers P. This vector is $|P|$ long, and contains a 1 in the i^{th} element if ϕ is compatible with the i^{th} primary cipher, and a 0 if it is not.

We use a superscript c on a cipher to notate its compatibility vector. Here are the compatibility vectors for four letter-pairs, using the primary ciphers outlined above, and are shown in Figure 1:

$(\substack{\texttt{i}\\\texttt{a}})^c = [1,0,0]$, $(\substack{\texttt{s}\\\texttt{n}})^c = [1,1,1]$,
$(\substack{\texttt{a}\\\texttt{o}})^c = [0,1,1]$, $(\substack{\texttt{z}\\\texttt{j}})^c = [1,1,1]$.

This is an upper-bound because the primary ciphers compatible with ϕ may not be compatible with each other. For example, the null cipher, which has no letter pairings, is compatible with all primary ciphers, but no full cipher contains all primaries. When we combine two ciphers ϕ_1 and ϕ_2, which have compatibility vectors ϕ_1^c and ϕ_2^c, the resulting cipher ϕ_3 has a compatibility vector $\phi_3^c = \phi_1^c \wedge \phi_2^c$, where $\wedge$ is the bitwise AND operation. We calculate the compatibility vector for every letter-pair, and can combine those vectors to determine the compatibility vector for any cipher. The heuristic's score for $(\substack{\texttt{asz}\\\texttt{onj}})$ follows.

$h = (\substack{\texttt{asz}\\\texttt{onj}})^c = (\substack{\texttt{a}\\\texttt{o}})^c \wedge (\substack{\texttt{s}\\\texttt{n}})^c \wedge (\substack{\texttt{z}\\\texttt{j}})^c = [0,1,1]$
$score(\substack{\texttt{asz}\\\texttt{onj}}) = 100 \cdot 0 + 82 \cdot 1 + 76 \cdot 1 = 158$

8.2 Lower Bound: Guarantee Vector

We can calculate another vector, g for every cipher ϕ which represents whether each primary cipher is a subcipher of ϕ. This forms a lower bound guarantee because any cipher which extends from ϕ will also contain the primary ciphers in g, plus potentially more. The null cipher in this case would have a heuristic vector g of all zeros, as it does not contain any of the primary ciphers. Likewise, in this P, all of the individual letter pairs $((\substack{\texttt{a}\\\texttt{o}}), (\substack{\texttt{s}\\\texttt{n}}), (\substack{\texttt{z}\\\texttt{j}}))$ would have a heuristic vector of all zeros, as all of the primaries require at least two letter-pairs.

Efficiently implementing this heuristic is slightly more challenging than the compatibility heuristic. Our method, which uses bitwise operations and is cacheable like the compatibility vector. Using this heuristic, g of $(\substack{\texttt{asz}\\\texttt{onj}})$ is $[0,0,1]$, as $(\substack{\texttt{is}\\\texttt{an}}) \not\subset (\substack{\texttt{asz}\\\texttt{onj}})$, $(\substack{\texttt{at}\\\texttt{of}}) \not\subset (\substack{\texttt{asz}\\\texttt{onj}})$, and $(\substack{\texttt{as}\\\texttt{on}}) \subset (\substack{\texttt{asz}\\\texttt{onj}})$.

This heuristic therefore scores $(\substack{\texttt{asz}\\\texttt{onj}})$ as follows:

$score(\substack{\texttt{asz}\\\texttt{onj}}) = 100 \cdot 0 + 82 \cdot 0 + 76 \cdot 1 = 76$

8.3 Middle Ground: Medium Vector

Both of the two aforementioned heuristics have weaknesses. The optimistic, max heuristic does not differentiate between a primary cipher it already has and one that it potentially has, and the conservative min heuristic is greedy and short-sighted. Our third heuristic incorporates elements from the first two, to ideally form a measure that is neither overly optimistic, or overly short-sighted. Unlike the lower bound heuristic in Section 8.2, which requires all letter pairings to count a primary cipher, this medium heuristic counts some of the primary cipher's score if some of the letter-pairs are present. For example, if a partial cipher has $3/4$ of the required letter pairings for a primary, it would count 75% of the score.

For example, $(\substack{\texttt{asz}\\\texttt{onj}})$ has one of the two letter pairings of the first primary, $(\substack{\texttt{is}\\\texttt{an}})$; one of the two letter pairings of the second primary, $(\substack{\texttt{at}\\\texttt{of}})$; and two of the two letter pairings of the third primary, $(\substack{\texttt{on}\\\texttt{as}})$. We represent this as [0.5,0.5,1]. However, we know from Section 8.2 that the first primary is incompatible with $(\substack{\texttt{asz}\\\texttt{onj}})$, and so we do not count its score. That makes the heuristic vector $h = [0, 0.5, 1]$, and $score(\substack{\texttt{asz}\\\texttt{onj}}) = 100 \cdot 0 + 82 \cdot .5 + 76 \cdot 1 = 117$.

We have now evaluated the same cipher using three different heuristics, all which produce a different score. These scores are guaranteed to converge to the same value at the full cipher.

8.4 Speed improvements

Table 8.4 shows the massive performance gains of the heuristics, which are over 3000 times faster than the simplistic means of scoring by iterating over every word and checking if it enciphers to anything useful.

9 Related Work

Nuhn et al. (2013) use a beam search algorithm to crack substitution ciphers. Our work differs from their's in several key ways: in Nuhn et al. (2013) there are two distinct symbol spaces, that of the ciphertext and that of the plaintext and so there is no concept of symmetry. Each step of Nuhn et al.'s beam search explores pairing a ciphertext character with a plaintext character, and decides upon the "extension-order" to pair the ciphertext letters, whereas each step of our search pairs two characters

Heuristic	Time	Memory
word	$1 \times 10^6 \mu s$	all words +1int/word
med 8.3	$3 \times 10^3 \mu s$	n bits/primary + 1 int/primary
min 8.2	$2 \times 10^3 \mu s$	n bits/primary + 1 int/primary
max 8.1	$3 \times 10^2 \mu s$	1 bit/primary + 1 int/primary

Table 2: Time to score a cipher using different means, and each mean's memory requirements. The word method stores the strings of all words and enciphers them and checks if they are valid words. It will produce the same value as the min heuristic.

together. As such, we make 13 decisions, not 26.

Additionally, the search space of the non-symmetric and symmetric ciphers are characteristically different. If the "extension-order" is predetermined as in Nuhn et al.'s work, there is only one path from the bottom of the graph to a full solution. In contrast, our graph has 13! different paths to any full solution, as all the permutations of the 13 different letter pairs are valid paths. On the one hand, this highly connected property of the graph means that we can prune our beam to a smaller size, as failing to expand a partial cipher does not eliminate it from appearing as a subcipher in a future solution like it does for Nuhn et al..

However, the connectedness of our cipher graph does present new challenges. As the Xenotext constraints are not satisfied by one cipher, we want to maximize the number of different solutions presented to the writer which each allow for unique expressive potential. We observe, however, that the connectedness property results in a final beam which is smaller and less diverse than would be anticipated. This is caused by multiple partial ciphers in a beam sharing the same "propensity" to become an identical full cipher. We solve this by enforcing that every partial cipher in the beam be incompatible with every other, thereby guaranteeing that no two partial ciphers can share the same "propensity", and that all possibilities generated from them in all future layers will be unique. As there are many ($\mathcal{O}(n^2)$) compatibility comparisons to be made at every beam, we limit only enforce compatibility for the first thousand ciphers in the beam.

Our scoring function is also entirely different from what would be used to crack a substitution cipher. Unlike Beam Verse, cracking software is tied to a ciphertext, and typically uses character-level n-gram data to calculate the probability that a decipherment is valid. Beam Verse, on the other hand, uses word-level n-grams as the basis of scoring, and is not tied to any given text, but suggests fragments of text which satisfy the Xenotext constraint.

10 Results

The raw score of a cipher changes according to the dataset, and so we report the score divided by highest scored cipher across all of the heuristics. Table 10 shows results using unigrams, while Table 10 shows results for primary ciphers generated from bigrams.

Heuristic	High	Low	End Beam Size
max	0.74	0.53	12160
min	0.98	0.97	4223
med	0.93	0.73	13043
max incomp	0.71	0.59	12160
min incomp	1.00	0.97	4181
med incomp	0.85	0.74	13043
Bök	0.39		

Table 3: Normalized scores for three different heuristics on highest 2^{16} **unigram** primary ciphers, and a variable beam aiming for 2^{15} ciphers. "Incomp" means that we enforce that all partial ciphers in the beam be incompatible with each other. The low value is the normalized score of the index of the shortest end beam. This is a better comparison that the last cipher in the beam, as the length of the beams is variable across heuristics.

Heuristic	High	Low	End Beam Size
max	0.81	0.73	9631
min	1.00	0.94	5777
med	0.97	0.88	13291
max incomp	0.81	0.73	9631
min incomp	0.97	0.88	13301
med incomp	0.97	0.84	13291
Bök	0.23		

Table 4: Normalized scores for different heuristics on highest 2^{20} **bigram** primary ciphers, and a variable beam aiming for 2^{15} ciphers.

We note that all ciphers we generate, regardless

of heuristic, score higher than the cipher Bök used to write his poems. This suggests that there are many ciphers other than Bök's which can be used to write Xenotext constrained poetry. Poems written using ciphers generated from Beam Verse can be found on our website. However, attempting to write with some high-scoring ciphers has revealed that our scoring metric may be only loosely correlated with the true "writability", as some ciphers which score higher that Bök's we find more difficult to write with.

Bök's cipher also scores relatively worse than the top ciphers using a bigram model (Table 10). Many of the bigrams Bök uses in his poems are not frequent enough to be in the Google bigrams. Anecdotally, we find ciphers generated using bigram data to be more writable, as bigram models begin to capture syntactic structure of the language.

Enforcing that each cipher in the beam be incompatible with every other showed minimal gains with some heuristics and minimal losses in others. It does, however, guarantee that more ciphers will be generated. Enforcing incompatibility is probably not worth the processing time if memory permits increasing the size of the beam instead.

The top scoring cipher[4] according to the unigram model performs similarly to the Bök cipher when scored against the bigram model, accumulating only 24% of the points the highest scoring bigram cipher does. The top bigram cipher[5] scores 68% of the top unigram cipher's score when using unigram scoring, not as drastic of a difference, but still low enough to be pruned by Beam Verse and not be discovered. The discrepancy in scores between models suggests that "writable" ciphers are excluded from our final results, and also encourages running Beam Verse on additional datasets to find new ciphers. A score which incorporates elements from multiple models of language might be explored, searching for ciphers which perform well across all datasets.

11 Further Work

Work in Kao (2011) sets out to quantify good poetic style and techniques. We note that some poetic techniques, like alliteration and anaphora, are preserved through the substitution cipher. We could boost alliterative n-grams to encourage Beam Verse to include alliterative n-grams.

As Beam Verse is language agnostic, all of the work here is applicable to other languages. The Xenotext constraints might be more easily satisfied in a different language than English, perhaps a language with a smaller character set like Hawaiian (which only consists of thirteen letters). Additionally, The Xenotext project as defined here only minimally uses the organism to actively write – the organism does not have any degree of freedom to express itself as its poem is precisely determined by the author's poem. However, DNA possesses computational power (Paun et al., 2005), which could be leveraged to generate natural language. By taking advantage of the complexity of the cell, its output could be more loosely defined, and change according to mutations in the DNA.

Further investigation can also be done into quantifying the "writability" of a limited vocabulary (perhaps using semantic and grammar data), and constrained text generation under constraint. Poetic endeavours with rigid mathematical constraints are not only attempted by Bök. Any work in the traditions of the Oulipo, a primarily French-speaking group of writers who explore the creative potential of mathematical and logical constraints, would stand to benefit immensely from software tools designed to aid constrained writing. Whereas visual artists and musicians have been quick to use computers to produce images and sounds which would have been impossible by traditional means, writers have been slow to use computers to produce works which would have been impossible to create otherwise.

12 Conclusion

In this paper we present a new metric to quantify "writability" of a symmetric substitution cipher. We experiment using three different heuristics in a beam search, an algorithm we call Beam Verse. We find that our score for "writability", which takes the minimum frequency of a word or n-gram pair, is effective at finding candidate ciphers, but is not a perfect metric of "writability" in this constrained environment. "Writability" is highly subjective, and possibly requires more data than just n-gram frequency

[4] $\begin{pmatrix} \texttt{abcdegijkmnqv} \\ \texttt{fhlryutpswozx} \end{pmatrix}$

[5] $\begin{pmatrix} \texttt{abcdefjklnpqx} \\ \texttt{ightomuvrswzy} \end{pmatrix}$

(eg. semantic and grammar information). Luckily, beam search is highly flexible, and any scoring function, perhaps using a more sophisticated model of writability, could be used in place of the one used here.

Source code and highly scoring ciphers are available for download[6].

References

Christian Bök. 2008. The xenotext experiment. *SCRIPTed*, 5:228–231.

Neil A. Campbell and Jane B. Reece. 2008. *Biology*. Pearson, 8th edition.

Jonathan PL Cox. 2001. Long-term data storage in dna. *TRENDS in Biotechnology*, 19(7):247–250.

Stefan Edelkamp and Stefan Schroedl. 2011. *Heuristic search: theory and applications*. Elsevier.

Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M LeProust, Botond Sipos, and Ewan Birney. 2013. Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, 494(7435):77–80.

Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. 2014. Solving substitution ciphers with combined language models. pages 2314–2325.

Justine T Kao. 2011. A computational analysis of poetic craft in contemporary professional and amateur poetry.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, and Jon Orwant. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. Citeseer.

Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. 2005. *DNA computing: new computing paradigms*. Springer Science & Business Media.

Boris Shimanovsky, Jessica Feng, and Miodrag Potkonjak. 2002. Hiding data in dna. pages 373–386. Springer.

Abraham Sinkov. 1966. Elementary cryptanalysis: A mathematical approach, mathematical association of america, 1966. *Additional Reading*.

Darren Wershler. 2012. The xenotext experiment, so far. *Canadian Journal of Communication*, 37(1):43.

[6]http://www.langlearnlab.cs.uvic.ca/beamverse

All Fingers are not Equal: Intensity of References in Scientific Articles

Tanmoy Chakraborty
Dept. of Computer Science & UMIACS
University of Maryland, College Park, USA
tanchak@umiacs.umd.edu

Ramasuri Narayanam
IBM Research, India
ramasurn@in.ibm.com

Abstract

Research accomplishment is usually measured by considering all citations with equal importance, thus ignoring the wide variety of purposes an article is being cited for. Here, we posit that measuring the intensity of a reference is crucial not only to perceive better understanding of research endeavor, but also to improve the quality of citation-based applications. To this end, we collect a rich annotated dataset with references labeled by the intensity, and propose a novel graph-based semi-supervised model, `GraLap` to label the intensity of references. Experiments with AAN datasets show a significant improvement compared to the baselines to achieve the true labels of the references (46% better correlation). Finally, we provide four applications to demonstrate how the knowledge of reference intensity leads to design better real-world applications.

1 Introduction

With more than one hundred thousand new scholarly articles being published each year, there is a rapid growth in the number of citations for the relevant scientific articles. In this context, we highlight the following interesting facts about the process of citing scientific articles: (i) the most commonly cited paper by Gerard Salton, titled "A Vector Space Model for Information Retrieval" (alleged to have been published in 1975) does not actually exist in reality (Dubin, 2004), (ii) the scientific authors read only 20% of the works they cite (Simkin and Roychowdhury, 2003), (iii) one third of the references in a paper are redundant and 40% are perfunctory (Moravcsik and Murugesan, 1975), (iv) 62.7% of the references could not be attributed a specific function (definition, tool etc.) (Teufel et al., 2006). Despite these facts, the existing bibliographic metrics consider that all citations are *equally significant*.

In this paper, we would emphasize the fact that *all the references of a paper are not equally influential*. For instance, we believe that for our current paper, (Wan and Liu, 2014) is more influential reference than (Garfield, 2006), although the former has received lower citations (9) than the latter (1650) so far[1]. Therefore the influence of a cited paper completely depends upon the context of the citing paper, not the overall citation count of the cited paper. We further took the opinion of the original authors of few selective papers and realized that around 16% of the references in a paper are highly influential, and the rest are trivial (Section 4). This motivates us to design a prediction model, `GraLap` to automatically label the influence of a cited paper with respect to a citing paper. Here, we label paper-reference pairs rather than references alone, because a reference that is influential for one citing paper may not be influential with equal extent for another citing paper.

We experiment with ACL Anthology Network (AAN) dataset and show that `GraLap` along with the novel feature set, quite efficiently, predicts the intensity of references of papers, which achieves (Pearson) correlation of 0.90 with the human annotations. Finally, we present four interesting appli-

[1]The statistics are taken from Google Scholar on June 2, 2016.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1348–1358,
Austin, Texas, November 1-5, 2016.

cations to show the efficacy of considering unequal intensity of references, compared to the uniform intensity.

The contributions of the paper are four-fold: (i) we acquire a rich annotated dataset where paper-reference pairs are labeled based on the influence scores (Section 4), which is perhaps the first gold-standard for this kind of task; (ii) we propose a graph-based label propagation model `GraLap` for semi-supervised learning which has tremendous potential for any task where the training set is less in number and labels are non-uniformly distributed (Section 3); (iii) we propose a diverse set of features (Section 3.3); most of them turn out to be quite effective to fit into the prediction model and yield improved results (Section 5); (iv) we present four applications to show how incorporating the reference intensity enhances the performance of several state-of-the-art systems (Section 6).

2 Defining Intensity of References

All the references of a paper usually do not carry equal intensity/strength with respect to the citing paper because some papers have influenced the research more than others. To pin down this intuition, here we discretize the reference intensity by numerical values within the range of 1 to 5, (5: most influential, 1: least influential). The appropriate definitions of different labels of reference intensity are presented in Figure 1, which are also the basis of building the annotated dataset (see Section 4):

Note that "reference intensity" and "reference similarity" are two different aspects. It might happen that two similar reference are used with different intensity levels in a citing paper – while one is just mentioned somewhere in the paper and other is used as a baseline. Here, we address the former problem as a semi-supervised learning problem with clues taken from content of the citing and cited papers.

3 Reference Intensity Prediction Model

In this section, we formally define the problem and introduce our prediction model.

- **Label-1:** The reference is related to the citing article with *very limited extent* and can be *removed* without compromising the competence of the references (e.g., (Garfield, 2006) for this paper).
- **Label-2:** The reference is *little mentioned* in the citing article and can be *replaced* by others without compromising the adequacy of the references (e.g., (Zhu et al., 2015) for this paper).
- **Label-3:** The reference occurs separately in a sentence within the citing article and has *no significant impact on the current problem* (e.g., references to metrics, tools) (e.g., (Porter, 1997) for this paper).
- **Label-4:** The reference is *important* and highly related to the citing article. It is usually mentioned several times in the article with long reference context (e.g., (Singh et al., 2015) for this paper).
- **Label-5:** The reference is *extremely important* and occurs (is emphasized) multiple times within the citing article. It generally points to the cited article from where the citing article borrows main ideas (and can be treated as a baseline) (e.g., (Wan and Liu, 2014) for this paper).

Figure 1: Definitions of the intensity of references.

3.1 Problem Definition

We are given a set of papers $\mathbb{P} = \{P_1, P_2, ..., P_M\}$ and a sets of references $\mathbb{R} = \{R_1, R_2, ..., R_M\}$, where R_i corresponds to the set of references (or cited papers) of P_i. There is a set of papers $P_L \in \mathbb{P}$ whose references $R_L \in \mathbb{R}$ are already labeled by $\ell \in L = \{1, ..., 5\}$ (each reference is labeled with exactly one value). Our objective is to define a predictive function f that labels the references $R_U \in \{\mathbb{R} \setminus R_L\}$ of the papers $P_U \in \{\mathbb{P} \setminus P_L\}$ whose reference intensities are unknown, i.e., $f : (\mathbb{P}, \mathbb{R}, P_L, R_L, P_U, R_L) \longrightarrow L$.

Since the size of the annotated (labeled) data is much smaller than unlabeled data ($|P_L| \ll |P_U|$), we consider it as a semi-supervised learning problem.

Definition 1. (Semi-supervised Learning) *Given a set of entries X and a set of possible labels Y_L, let us assume that (x_1, y_1), (x_2, y_2),..., (x_l, y_l) be the set of labeled data where x_i is a data point and $y_i \in Y_L$ is its corresponding label. We assume that at least one instance of each class label*

is present in the labeled dataset. Let (x_{l+1}, y_{l+1}), (x_{l+2}, y_{l+2}),..., (x_{l+n}, y_{l+u}) be the unlabeled data points where $Y_U = \{y_{l+1}, y_{l+2}, ...y_{l+u}\}$ are unknown. Each entry $x \in X$ is represented by a set of features $\{f_1, f_2, ..., f_D\}$. The problem is to determine the unknown labels using X and Y_L.

3.2 `GraLap`: A Prediction Model

We propose `GraLap`, a variant of label propagation (LP) model proposed by (Zhu et al., 2003) where a node in the graph propagates its associated label to its neighbors based on the proximity. We intend to assign same label to the vertices which are closely connected. However unlike the traditional LP model where the original values of the labels continue to fade as the algorithm progresses, we systematically handle this problem in `GraLap`. Additionally, we follow a post-processing in order to handle "class-imbalance problem".

Graph Creation. The algorithm starts with the creation of a *fully connected weighted graph* $G = (X, E)$ where nodes are data points and the weight w_{ij} of each edge $e_{ij} \in E$ is determined by the radial basis function as follows:

$$w_{ij} = exp\left(- \frac{\sum_{d=1}^{D}(x_i^d - x_j^d)^2}{\sigma^2} \right) \quad (1)$$

The weight is controlled by a parameter σ. Later in this section, we shall discuss how σ is selected. Each node is allowed to propagate its label to its neighbors through edges (the more the edge weight, the easy to propagate).

Transition Matrix. We create a probabilistic transition matrix $T_{|X|\times|X|}$, where each entry T_{ij} indicates the probability of jumping from j to i based on the following: $T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{|X|} w_{kj}}$.

Label Matrix. Here, we allow a soft label (interpreted as a distribution of labels) to be associated with each node. We then define a label matrix $Y_{|X|\times|L|}$, where ith row indicates the label distribution for node x_i. Initially, Y contains only the values of the labeled data; others are zero.

Label Propagation Algorithm. This algorithm works as follows:

After initializing Y and T, the algorithm starts by disseminating the label from one node to its neighbors (including self-loop) in one step (Step 3). Then we normalize each entry of Y by the sum of its corresponding row in order to maintain the interpretation of label probability (Step 4). Step 5 is crucial; here we want the labeled sources X_L to be persistent. During the iterations, the initial labeled nodes X_L may fade away with other labels. Therefore we forcefully restore their actual label by setting $y_{il} = 1$ (if $x_i \in X_L$ is originally labeled as l), and other entries ($\forall_{j \neq l} y_{ij}$) by zero. We keep on "pushing" the labels from the labeled data points which in turn pushes the class boundary through high density data points and settles in low density space. In this way, our approach intelligently uses the unlabeled data in the intermediate steps of the learning.

1: Initialize T and Y
2: **while** (Y does not converge) **do**
3: $\quad Y \leftarrow TY$
4: $\quad$ Normalize rows of Y, $y_{ij} = \frac{y_{ij}}{\sum_k y_{ik}}$
5: $\quad$ Reassign original labels to X_L

Assigning Final Labels. Once Y_U is computed, one may take the most likely label from the label distribution for each unlabeled data. However, this approach does not guarantee the label proportion observed in the annotated data (which in this case is not well-separated as shown in Section 4). Therefore, we adopt a *label-based normalization* technique. Assume that the label proportions in the labeled data are $c_1, ..., c_{|L|}$ (s.t. $\sum_{i=1}^{|L|} c_i = 1$). In case of Y_U, we try to balance the label proportion observed in the ground-truth. The label mass is the column sum of Y_U, denoted by $Y_{U_{.1}}, ..., Y_{U_{.|L|}}$, each of which is scaled in such a way that $Y_{U_{.1}} : ... : Y_{U_{.|L|}} = c_1 : ... : c_{|L|}$. The label of an unlabeled data point is finalized as the label with maximum value in the row of Y.

Convergence. Here we briefly show that our algorithm is guaranteed to converge. Let us combine Steps 3 and 4 as $Y \leftarrow \hat{T}Y$, where $\hat{T} = T_{ij}/\sum_k T_{ik}$. Y is composed of $Y_{L_{l\times|L|}}$ and $Y_{U_{u\times|L|}}$, where Y_U never changes because of the reassignment. We can split $\hat{T}$ at the boundary of labeled and unlabeled data as follows:

$$\hat{F} = \begin{bmatrix} \hat{T}_{ll} & \hat{T}_{lu} \\ \hat{T}_{ul} & \hat{T}_{uu} \end{bmatrix}$$

Therefore, $Y_U \leftarrow \hat{T}_{uu}Y_U + \hat{T}_{ul}Y_L$, which can lead to $Y_U = \lim_{n\rightarrow\infty} \hat{T}_{uu}^n Y^0 + [\sum_{i=1}^{n} \hat{T}_{uu}^{(i-1)}]\hat{T}_{ul}Y_L$, where Y^0 is the shape of Y at iteration 0. We need

to show $\hat{T}^n_{uu_{ij}} Y^0 \leftarrow 0$. By construction, $\hat{T}_{ij} \geq 0$, and since $\hat{T}$ is row-normalized, and $\hat{T}_{uu}$ is a part of $\hat{T}$, it leads to the following condition: $\exists \gamma < 1$, $\sum_{j=1}^{u} \hat{T}_{uu_{ij}} \leq \gamma$, $\forall i = 1, ..., u$. So,

$$\begin{aligned}\sum_j \hat{T}^n_{uu_{ij}} &= \sum_j \sum_k \hat{T}^{(n-1)}_{uu_{ik}} \hat{T}_{uu_{kj}} \\ &= \sum_k \hat{T}^{(n-1)}_{uu_{ik}} \sum_j \hat{T}_{uu_{ik}} \\ &\leq \sum_k \hat{T}^{(n-1)}_{uu_{ik}} \gamma \\ &\leq \gamma^n\end{aligned}$$

Therefore, the sum of each row in $\hat{T}^n_{uu_{ij}}$ converges to zero, which indicates $\hat{T}^n_{uu_{ij}} Y^0 \leftarrow 0$.

Selection of σ. Assuming a spatial representation of data points, we construct a minimum spanning tree using Kruskal's algorithm (Kruskal, 1956) with distance between two nodes measured by Euclidean distance. Initially, no nodes are connected. We keep on adding edges in increasing order of distance. We choose the distance (say, d_f) of the first edge which connects two components with different labeled points in them. We consider d_f as a heuristic to the minimum distance between two classes, and arbitrarily set $\sigma = d_0/3$, following 3σ rule of normal distribution (Pukelsheim, 1994).

3.3 Features for Learning Model

We use a wide range of features that suitably represent a paper-reference pair (P_i, R_{ij}), indicating P_i refers to P_j through reference R_{ij}. These features can be grouped into six general classes.

3.3.1 Context-based Features (CF)

The "reference context" of R_{ij} in P_i is defined by three-sentence window (sentence where R_{ij} occurs and its immediate previous and next sentences). For multiple occurrences, we calculate its average score. We refer to "reference sentence" to indicate the sentence where R_{ij} appears.

(i) *CF:Alone*. It indicates whether R_{ij} is mentioned alone in the reference context or together with other references.

(ii) *CF:First*. When R_{ij} is grouped with others, this feature indicates whether it is mentioned first (e.g., "[2]" is first in "[2,4,6]").

Next four features are based on the occurrence of words in the corresponding lists created manually (see Table 1) to understand different aspects.

(iii) *CF:Relevant*. It indicates whether R_{ij} is explicitly mentioned as relevant in the reference context (`Rel` in Table 1).

(iv) *CF:Recent*. It tells whether the reference context indicates that R_{ij} is new (`Rec` in Table 1).

(v) *CF:Extreme*. It implies that R_{ij} is extreme in some way (`Ext` in Table 1).

(vi) *CF:Comp*. It indicates whether the reference context makes some kind of comparison with R_{ij} (`Comp` in Table 1).

Note we do not consider any sentiment-based features as suggested by (Zhu et al., 2015).

3.3.2 Similarity-based Features (SF)

It is natural that the high degree of semantic similarity between the contents of P_i and P_j indicates the influence of P_j in P_i. We assume that although the full text of P_i is given, we do not have access to the full text of P_j (may be due to the subscription charge or the unavailability of the older papers). Therefore, we consider only the title of P_j as a proxy of its full text. Then we calculate the cosine-similarity[2] between the title (*T*) of P_j and (i) *SF:TTitle.* the title, (ii) *SF:TAbs*. the abstract, *SF:TIntro*. the introduction, (iv) *SF:TConcl*. the conclusion, and (v) *SF:TRest*. the rest of the sections (sections other than abstract, introduction and conclusion) of P_i.

We further assume that the "reference context" (*RC*) of P_j in P_i might provide an alternate way of summarizing the usage of the reference. Therefore, we take the same similarity based approach mentioned above, but replace the title of P_j with its *RC* and obtain five more features: (vi) *SF:RCTitle*, (vii) *SF:RCAbs*, (viii) *SF:RCIntro*, (ix) *SF:RCConcl* and (x) *SF:RCRest*. If a reference appears multiple times in a citing paper, we consider the aggregation of all RCs together.

3.3.3 Frequency-based Feature (FF)

The underlying assumption of these features is that a reference which occurs more frequently in a citing paper is more influential than a single occurrence (Singh et al., 2015). We count the frequency of R_{ij} in (i) *FF:Whole*. the entire content, (ii) *FF:Intro*. the introduction, (iii) *FF:Rel*. the related work, (iv) *FF:Rest*. the rest of the sections (as

[2]We use the vector space based model (Turney and Pantel, 2010) after stemming the words using Porter stammer (Porter, 1997).

Rel	pivotal, comparable, innovative, relevant, relevantly, inspiring, related, relatedly, similar, similarly, applicable, appropriate, pertinent, influential, influenced, original, originally, useful, suggested, interesting, inspired, likewise
Rec	recent, recently, latest, later, late, latest, up-to-date, continuing, continued, upcoming, expected, update, renewed, extended subsequent, subsequently, initial, initially, sudden, current, currently, future, unexpected, previous, previously, old, ongoing, imminent, anticipated, unprecedented, proposed, startling, preliminary, ensuing, repeated, reported, new, earlier, earliest, early, existing, further, revised, improved
Ext	greatly, awfully, drastically, intensely, acutely, almighty, exceptionally, excessively, exceedingly, tremendously, importantly significantly, notably, outstandingly
Comp	easy, easier, easiest, vague, vaguer, vaguest, weak, weaker, weakest, strong, stronger, strongest, bogus, unclear

Table 1: Manually curated lists of words collected from analyzing the reference contexts. The lists are further expanded using the Wordnet:Synonym with different lexical variations. Note that while searching the occurrence of these words in reference contexts, we use different lexical variations of the words instead of exact matching.

mentioned in Section 3.3.2) of P_i. We also introduce (v) *FF:Sec.* to measure the fraction of different sections of P_i where R_{ij} occurs (assuming that appearance of R_{ij} in different sections is more influential). These features are further normalized using the number of sentences in P_i in order to avoid unnecessary bias on the size of the paper.

3.3.4 Position-based Features (PF)

Position of a reference in a paper might be a predictive clue to measure the influence (Zhu et al., 2015). Intuitively, the earlier the reference appears in the paper, the more important it seems to us. For the first two features, we divide the entire paper into two parts equally based on the sentence count and then see whether R_{ij} appears (i) *PF:Begin.* in the beginning or (ii) *PF:End.* in the end of P_i. Importantly, if R_{ij} appears multiple times in P_i, we consider the fraction of times it occurs in each part.

For the other two features, we take the entire paper, consider sentences as atomic units, and measure position of the sentences where R_{ij} appears, including (iii) *PF:Mean.* mean position of appearance, (iv) *PF:Std.* standard deviation of different appearances. These features are normalized by the total length (number of sentences) of P_i. , thus ranging from 0 (indicating beginning of P_i) to 1 (indicating the end of P_i).

3.3.5 Linguistic Features (LF)

The linguistic evidences around the context of R_{ij} sometimes provide clues to understand the intrinsic influence of P_j on P_i. Here we consider word level and structural features.

(i) *LF:NGram.* Different levels of n-grams (1-grams, 2-grams and 3-grams) are extracted from the reference context to see the effect of different word combination (Athar and Teufel, 2012).

(ii) *LF:POS.* Part-of-speech (POS) tags of the words in the reference sentence are used as features (Jochim and Schütze, 2012).

(iii) *LF:Tense.* The main verb of the reference sentence is used as a feature (Teufel et al., 2006).

(iv) *LF:Modal.* The presence of modal verbs (e.g., "can", "may") often indicates the strength of the claims. Hence, we check the presence of the modal verbs in the reference sentence.

(v) *LF:MainV.* We use the main-verb of the reference sentence as a direct feature in the model.

(vi) *LF:hasBut.* We check the presence of conjunction "but", which is another clue to show less confidence on the cited paper.

(vii) *LF:DepRel.* Following (Athar and Teufel, 2012) we use all the dependencies present in the reference context, as given by the dependency parser (Marneffe et al., 2006).

(viii) *LF:POSP.* (Dong and Schfer, 2011) use seven regular expression patterns of POS tags to capture syntactic information; then seven boolean features mark the presence of these patterns. We also utilize the same regular expressions as shown below [3] with the examples (the empty parenthesis in each example indicates the presence of a reference token R_{ij} in the corresponding sentence; while few examples are complete sentences, few are not):

- ".*\\(\\) VV[DPZN].*": *Chen () showed that cohesion is held in the vast majority of cases for English-French.*
- ".*(VHP|VHZ) VV.*": *while Cherry and Lin () have shown it to be a strong feature for word alignment...*
- ".*VH(D|G|N|P|Z) (RB)*VBN.*": *Inducing features for taggers by clustering has been tried by several researchers ().*
- ".*MD (RB)*VB(RB)* VVN.*": *For example, the likelihood of those generative procedures can be accumulated to get the likelihood of the phrase pair ().*

[3] The meaning of each POS tag can be found in `http://nlp.stanford.edu/software/tagger.shtml`(Toutanova and Manning, 2000).

- "[IW.]*VB(D|P|Z) (RB)*VV[ND].*": *Our experimental set-up is modeled after the human evaluation presented in ().*
- "(RB)*PP (RB)*V.*": *We use CRF () to perform this tagging.*
- ".*VVG (NP)*(CC)*(NP).*": *Following (), we provide the annotators with only short sentences: those with source sentences between 10 and 25 tokens long.*

These are all considered as Boolean features. For each feature, we take all the possible evidences from all paper-reference pairs and prepare a vector. Then for each pair, we check the presence (absence) of tokens for the corresponding feature and mark the vector accordingly (which in turn produces a set of Boolean features).

3.3.6 Miscellaneous Features (MS)

This group provides other factors to explain why is a paper being cited. (i) *MS:GCount*. To answer whether a highly-cited paper has more academic influence on the citing paper than the one which is less cited, we measure the number of other papers (except P_i) citing P_j.
(ii) *MS:SelfC*. To see the effect of self-citation, we check whether at least one author is common in both P_i and P_j.
(iii) *MG:Time*. The fact that older papers are rarely cited, may not stipulate that these are less influential. Therefore, we measure the difference of the publication years of P_i and P_j.
(iv) *MG:CoCite*. It measures the co-citation counts of P_i and P_j defined by $\frac{|R_i \cap R_j|}{|R_i \cup R_j|}$, which in turn answers the significance of reference-based similarity driving the academic influence (Small, 1973).

Following (Witten and Frank, 2005), we further make one step normalization and divide each feature by its maximum value in all the entires.

4 Dataset and Annotation

We use the AAN dataset (Radev et al., 2009) which is an assemblage of papers included in ACL related venues. The texts are preprocessed where sentences, paragraphs and sections are properly separated using different markers. The filtered dataset contains 12,843 papers (on average 6.21 references per paper) and 11,092 unique authors.

Next we use *Parscit* (Councill et al., 2008) to identify the reference contexts from the dataset and then extract the section headings from all the papers. Then each section heading is mapped into one of the following broad categories using the method proposed by (Liakata et al., 2012): Abstract, Introduction, Related Work, Conclusion and Rest.

Dataset Labeling. The hardest challenge in this task is that there is no publicly available dataset where references are annotated with the intensity value. Therefore, we constructed our own annotated dataset in two different ways. (i) *Expert Annotation*: we requested members of our research group[4] to participate in this survey. To facilitate the labeling process, we designed a portal where all the papers present in our dataset are enlisted in a drop-down menu. Upon selecting a paper, its corresponding references were shown with five possible intensity values. The citing and cited papers are also linked to the original texts so that the annotators can read the original papers. A total of 20 researchers participated and they were asked to label as many paper-reference pairs as they could based on the definitions of the intensity provided in Section 2. The annotation process went on for one month. Out of total 1640 pairs annotated, 1270 pairs were taken such that each pair was annotated by at least two annotators, and the final intensity value of the pair was considered to be the average of the scores. The Pearson correlation and Kendell's τ among the annotators are 0.787 and 0.712 respectively. (ii) *Author Annotation*: we believe that the authors of a paper are the best experts to judge the intensity of references present in the paper. With this intension, we launched a survey where we requested the authors whose papers are present in our dataset with significant numbers. We designed a web portal in similar fashion mentioned earlier; but each author was only shown her own papers in the drop-down menu. Out of 35 requests, 22 authors responded and total 196 pairs are annotated. This time we made sure that each paper-reference pair was annotated by only one author. The percentages of labels in the overall annotated dataset are as follows: 1: 9%, 2: 74%, 3: 9%, 4: 3%, 5: 4%.

5 Experimental Results

In this section, we start with analyzing the importance of the feature sets in predicting the reference

[4] All were researchers with the age between 25-45 working on document summarization, sentiment analysis, and text mining in NLP.

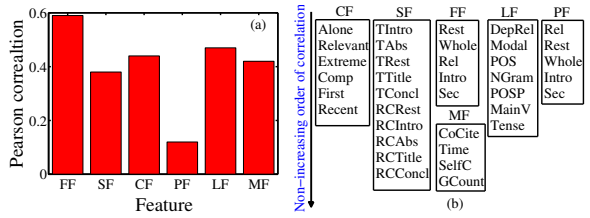


Figure 2: Pearson correlation coefficient between the features and the gold-standard annotations. (a) Group-wise average correlation, and (b) ranking of features in each group based on the correlation.

intensity, followed by the detailed results.

Feature Analysis. In order to determine which features highly determine the gold-standard labeling, we measure the Pearson correlation between various features and the ground-truth labels. Figure 2(a) shows the average correlation for each feature group, and in each group the rank of features based on the correlation is shown in Figure 2(b). Frequency-based features (*FF*) turn out to be the best, among which *FF:Rest* is mostly correlated. This set of features is convenient and can be easily computed. Both *CF* and *LF* seem to be equally important. However, PF tends to be less important in this task.

Model	*RMSE*	ρ	R^2
Uniform	2.09	-0.05	3.21
SVR+W	1.95	0.54	1.34
SVR+O	1.92	0.56	1.29
C4.5SSL	1.99	0.46	2.46
GLM	1.98	0.52	1.35

(a) Baselines

No.	Model	*RMSE*	ρ	R^2
(1)	GraLap+ FF	1.10	0.79	1.05
(2)	(1) + LF	0.98	0.84	0.95
(3)	(2) + CF	0.90	0.87	0.87
(4)	(3) + MF	0.95	0.89	0.84
(5)	(4) + SF	0.92	0.90	0.82
(6)	(5) + PF	0.91	0.90	0.80

(b) Our model

Table 2: Performance of the competing models. The features are added greedily into the GraLap model.

Results of Predictive Models. For the purpose of evaluation, we report the average results after 10-fold cross-validation. Here we consider five baselines to compare with GraLap: (i) Uniform: assign 3 to all the references assuming equal intensity, (ii) SVR+W: recently proposed Support Vector Regression (SVR) with the feature set mentioned in (Wan and Liu, 2014), (iii) SVR+O: SVR model with our feature set, (iv) C4.5SSL: C4.5 semi-supervised algorithm with our feature set (Quinlan, 1993), and (v) GLM: the traditional graph-based LP model with our feature set (Zhu et al., 2003). Three metrics are used to compare the results of the competing models with the annotated labels: *Root Mean Square Error* (*RMSE*), *Pearson's correlation coefficient* (ρ), and *coefficient of determination* (R^2)[5].

Table 2 shows the performance of the competing models. We incrementally include each feature set into GraLap greedily on the basis of ranking shown in Figure 2(a). We observe that GraLap with only FF outperforms SVR+O with 41% improvement of ρ. As expected, the inclusion of PF into the model improves the model marginally. However, the overall performance of GraLap is significantly higher than any of the baselines ($p < 0.01$).

6 Applications of Reference Intensity

In this section, we provide four different applications to show the use of measuring the intensity of references. To this end, we consider all the labeled entries for training and run GraLap to predict the intensity of rest of the paper-reference pairs.

6.1 Discovering Influential Articles

Influential papers in a particular area are often discovered by considering *equal weights* to all the citations of a paper. We anticipate that considering the reference intensity would perhaps return more meaningful results. To show this, Here we use the following measures individually to compute the influence of a paper: (i) RawCite: total number of citations per paper, (ii) RawPR: we construct a citation network (nodes: papers, links: citations), and measure PageRank (Page et al., 1998) of each node n: $PR(n) = \frac{1-q}{N} + q\sum_{m\in M(n)} \frac{PR(m)}{|L(m)|}$; where, q, the damping factor, is set to 0.85, N is the total number of nodes, $M(n)$ is the set of nodes that have edges to n, and $L(m)$ is the set of nodes that m has an edge to, (iii) InfCite: the weighted version of RawCite, measured by the sum of intensities of all citations of a paper, (iv) InfPR: the weighted version of RawPR: $PR(n) = \frac{1-q}{N} + q\sum_{m\in M(n)} \frac{Inf(m\to n)PR(m)}{\sum_{a\in L(m)Inf(m\to a)}}$, where Inf indicates the influence of a reference. We rank all the articles based on these four measures separately. Table 3(a) shows the Spearman's rank correlation between pair-wise measures. As expected, (i) and (ii) have high correlation (same for (iii) and (iv)), whereas across two types of measures the correlation is less. Further, in order to know which mea-

[5]The less (*resp.* more) the value of $RMSE$ and R^2 (*resp.* ρ), the better the performance of the models.

sure is more relevant, we conduct a subjective study where we select top ten papers from each measure and invite the experts (not authors) who annotated the dataset, to make a binary decision whether a recommended paper is relevant. [6]. The average pair-wise inter-annotator's agreement (based on Cohen's kappa (Cohen, 1960)) is 0.71. Table 3(b) presents that out of 10 recommendations of `InfPR`, 7 (5) papers are marked as influential by majority (all) of the annotators, which is followed by `InfCite`. These results indeed show the utility of measuring reference intensity for discovering influential papers. Top three papers based on `InfPR` from the entire dataset are shown in Table 4.

	RowCite	RowPR	InfCite	InfPR
RowCite	1	0.82	0.61	0.54
RowPR	0.82	1	0.52	0.63
InfCite	0.61	0.52	1	0.84
InfPR	0.54	0.63	0.84	1

(a)

Metric	All	Majority
RowCite	2	5
RowPR	2	4
InfCite	4	5
InfPR	5	7

(b)

Table 3: (a) Spearman's rank correlation among influence measures and (b) expert evaluation of the ranked results (for top 10 recommendations).

6.2 Identifying Influential Authors

H-index, a measure of impact/influence of an author, considers each citation with equal weight (Hirsch, 2005). Here we incorporate the notion of reference intensity into it and define `hif-index`.

Definition 2. *An author A with a set of papers $P(A)$ has an* `hif-index` *equals to h, if h is the largest value such that $|\{p \in P(A) | Inf(p) \geq h\}| \geq h$; where $Inf(p)$ is the sum of intensities of all citations of p.*

We consider 37 ACL fellows as the list of gold-standard influential authors. For comparative evaluation, we consider the total number of papers (`TotP`), total number of citations (`TotC`) and average citations per paper (`AvgC`) as three competing measures along with `h-index` and `hif-index`. We arrange all the authors in our dataset in decreasing order of each measure. Figure 3(a) shows the Spearman's rank correlation among the common elements across pair-wise rankings. Figure 3(b) shows the $Precision@k$ for five competing measures at identifying ACL fellows. We observe that `hif-index` performs significantly well with an overall precision of 0.54, followed by `AvgC` (0.37), h-index (0.35), `TotC` (0.32) and `TotP` (0.34). This result is an encouraging evidence that the reference-intensity could improve the identification of the influential authors. Top three authors based on `hif-index` are shown in Table 4.

	TopP	TopC	AvgC	H–index	Hif–index
TopP	1	0.27	0.34	0.24	0.21
TopC	0.27	1	0.35	0.32	0.28
AvgC	0.34	0.35	1	0.24	0.38
H–index	0.24	0.32	0.24	1	0.31
Hif–index	0.21	0.28	0.38	0.31	1

(a)

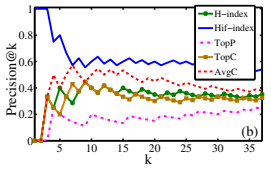


Figure 3: (a) Sprearman's rank correlation among pair-wise ranks, and (b) the performance of all the measures.

6.3 Effect on Recommendation System

Here we show the effectiveness of reference-intensity by applying it to a real paper recommendation system. To this end, we consider `FeRoSA`[7] (Chakraborty et al., 2016), a new (probably the first) framework of faceted recommendation for scientific articles, where given a query it provides facet-wise recommendations with each facet representing the purpose of recommendation (Chakraborty et al., 2016). The methodology is based on random walk with restarts (RWR) initiated from a query paper. The model is built on AAN dataset and considers both the citation links and the content information to produce the most relevant results. Instead of using the unweighted citation network, here we use the weighted network with each edge labeled by the intensity score. The final recommendation of `FeRoSA` is obtained by performing RWR with the transition probability proportional to the edge-weight (we call it `Inf-FeRoSA`). We observe that `Inf-FeRoSA` achieves an average precision of 0.81 at top 10 recommendations, which is 14% higher then `FeRoSA` while considering the flat version and 12.34% higher than `FeRoSA` while considering the faceted version.

6.4 Detecting Citation Stacking

Recently, Thomson Reuters began screening for journals that exchange large number of anomalous citations with other journals in a cartel-like arrangement, often known as "citation stacking" (Jump, 2013; Hardcastle, 2015). This sort of citation stacking is much more pernicious and difficult to detect.

[6] We choose papers from the area of "sentiment analysis" on which experts agree on evaluating the papers.

[7] `www.ferosa.org`

No	Paper	Author
1.	Lexical semantic techniques for corpus analysis (Pustejovsky et al., 1993)	Mark Johnson
2.	An unsupervised method for detecting grammatical errors (Chodorow and Leacock, 2000)	Christopher D. Manning
3.	A maximum entropy approach to natural language processing (Berger et al., 1996)	Dan Klein

Table 4: Top three papers and authors based on `InfPR` and `Hif-index` respectively.

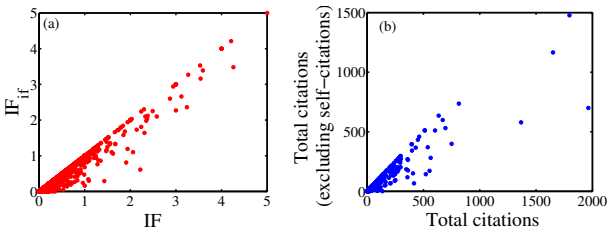


Figure 4: Correlation between (a) IF and IF_{if} and (b) number of citations before and after removing self-journal citations.

We anticipate that this behavior can be detected by the reference intensity. Since the AAN dataset does not have journal information, we use DBLP dataset (Singh et al., 2015) where the complete metadata information (along with reference contexts and abstract) is available, except the full content of the paper (559,338 papers and 681 journals; more details in (Chakraborty et al., 2014)). From this dataset, we extract all the features mentioned in Section 3.3 except the ones that require full text, and run our model using the existing annotated dataset as training instances. We measure the traditional impact factor (IF) of the journals and impact factor after considering the reference intensity (IF_{if}). Figure 4(a) shows that there are few journals whose IF_{if} significantly deviates (3σ from the mean) from IF; out of the suspected journals 70% suffer from the effect of self-journal citations as well (shown in Figure 4(b)), example including *Expert Systems with Applications* (current IF of 2.53). One of the future work directions would be to predict such journals as early as possible after their first appearance.

7 Related Work

Although the citation count based metrics are widely accepted (Garfield, 2006; Hirsch, 2010), the belief that mere counting of citations is dubious has also been a subject of study (Chubin and Moitra, 1975). (Garfield, 1964) was the first who explained the reasons of citing a paper. (Pham and Hoffmann, 2003) introduced a method for the rapid development of complex rule bases for classifying text segments. (Dong and Schfer, 2011) focused on a less manual approach by learning domain-insensitive features from textual, physical, and syntactic aspects To address concerns about h-index, different alternative measures are proposed (Waltman and van Eck, 2012). However they too could benefit from filtering or weighting references with a model of influence. Several research have been proposed to weight citations based on factors such as the prestige of the citing journal (Ding, 2011; Yan and Ding, 2010), prestige of an author (Balaban, 2012), frequency of citations in citing papers (Hou et al., 2011). Recently, (Wan and Liu, 2014) proposed a SVR based approach to measure the intensity of citations. Our methodology differs from this approach in at lease four significant ways: (i) they used six very shallow level features; whereas we consider features from different dimensions, (ii) they labeled the dataset by the help of independent annotators; here we additionally ask the authors of the citing papers to identify the influential references which is very realistic (Gilbert, 1977); (iii) they adopted SVR for labeling, which does not perform well for small training instances; here we propose `GraLap` , designed specifically for small training instances; (iv) four applications of reference intensity mentioned here are completely new and can trigger further to reassessing the existing bibliometrics.

8 Conclusion

We argued that the equal weight of all references might not be a good idea not only to gauge success of a research, but also to track follow-up work or recommending research papers. The annotated dataset would have tremendous potential to be utilized for other research. Moreover, `GraLap` can be used for any semi-supervised learning problem. Each application mentioned here needs separate attention. In future, we shall look into more linguistic evidences to improve our model.

References

Awais Athar and Simone Teufel. 2012. Context-enhanced citation sentiment detection. In *NAACL*, pages 597–601, Stroudsburg, PA, USA. ACL.

Alexandru T. Balaban. 2012. Positive and negative aspects of citation indices and journal impact factors. *Scientometrics*, 92(2):241–247.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March.

Tanmoy Chakraborty, Suhansanu Kumar, Pawan Goyal, Niloy Ganguly, and Animesh Mukherjee. 2014. Towards a stratified learning approach to predict future citation counts. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '14, pages 351–360, Piscataway, NJ, USA. IEEE Press.

Tanmoy Chakraborty, Amrith Krishna, Mayank Singh, Niloy Ganguly, Pawan Goyal, and Animesh Mukherjee, 2016. *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part II*, chapter FeRoSA: A Faceted Recommendation System for Scientific Articles, pages 528–541. Springer International Publishing, Cham.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *NAACL*, pages 140–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. E. Chubin and S. D. Moitra. 1975. Content-Analysis of References Adjunct or Alternative to Citation Counting. *Social studies of science*, 5(4):423–441.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–41.

Isaac G Councill, C Lee Giles, and Min-Yen Kan. 2008. Parscit: an open-source crf reference string parsing package. In *LREC*, pages 28–30, Marrakech, Morocco.

Ying Ding. 2011. Applying weighted pagerank to author citation networks. *JASIST*, 62(2):236–245.

Cailing Dong and Ulrich Schfer. 2011. Ensemble-style self-training on citation classification. In *IJCNLP*, pages 623–631. ACL, 11.

David Dubin. 2004. The most influential paper gerard salton never wrote. *Library Trends*, 52(4):748–764.

Eugene Garfield. 1964. Can citation indexing be automated? *Statistical association methods for mechanized documentation, Symposium proceedings*, pages 188–192.

Eugene Garfield. 2006. The History and Meaning of the Journal Impact Factor. *JAMA*, 295(1):90–93.

G. N. Gilbert. 1977. Referencing as persuasion. *Social Studies of Science*, 7(1):113–122.

James Hardcastle. 2015. Citations, self-citations, and citation stacking, `http://editorresources.taylorandfrancisgroup.com/citations-self-citations\-and-citation-stacking/`.

J. E. Hirsch. 2005. An index to quantify an individual's scientific research output. *PNAS*, 102(46):16569–16572.

J. E. Hirsch. 2010. An index to quantify an individual's scientific research output that takes into account the effect of multiple coauthorship. *Scientometrics*, 85(3):741–754, December.

Wen-Ru Hou, Ming Li, and Deng-Ke Niu. 2011. Counting citations in texts rather than reference lists to improve the accuracy of assessing scientific contribution. *BioEssays*, 33(10):724–727.

Charles Jochim and Hinrich Schütze. 2012. Towards a generic and flexible citation classifier based on a faceted classification scheme. In *COLING*, pages 1343–1358, Bombay, India.

Paul Jump. 2013. Journal citation cartels on the rise, `https://www.timeshighereducation.com/news/journal-citation-cartels-on-the-rise/2005009.article`.

J. B. Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin R. Batchelor, and Dietrich Rebholz-Schuhmann. 2012. Automatic recognition of conceptualization zones in scientific articles and two life science applications. *Bioinformatics*, 28(7):991–1000.

M. Marneffe, B. Maccartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*, pages 449–454, Genoa, Italy, May. European Language Resources Association (ELRA).

M. J. Moravcsik and P. Murugesan. 1975. Some results on the function and quality of citations. *Social studies of science*, 5(1):86–92.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. In *WWW*, pages 161–172, Brisbane, Australia.

Son Bao Pham and Achim Hoffmann. 2003. A new approach for scientific citation classification using cue phrases. In Tamas Domonkos Gedeon and Lance Chun Che Fung, editors, *Advances in Artificial Intelligence: 16th Australian Conference on AI*, pages 759–771. Springer Berlin Heidelberg.

M. F. Porter. 1997. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Friedrich Pukelsheim. 1994. The Three Sigma Rule. *The American Statistician*, 48(2):88–91.

James Pustejovsky, Peter Anick, and Sabine Bergler. 1993. Lexical semantic techniques for corpus analysis. *Comput. Linguist.*, 19(2):331–358, June.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The acl anthology network corpus. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, NLPIR4DL, pages 54–61, Stroudsburg, PA, USA. ACL.

Mikhail V. Simkin and V. P. Roychowdhury. 2003. Read Before You Cite! *Complex Systems*, 14:269–274.

Mayank Singh, Vikas Patidar, Suhansanu Kumar, Tanmoy Chakraborty, Animesh Mukherjee, and Pawan Goyal. 2015. The role of citation context in predicting long-term citation profiles: An experimental study based on a massive bibliographic text dataset. In *CIKM*, pages 1271–1280, New York, NY, USA. ACM.

Henry Small. 1973. Co-citation in the scientific literature: A new measure of the relationship between two documents. *JASIST*, 24(4):265–269.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *EMNLP*, pages 103–110, Stroudsburg, PA, USA. ACL.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, pages 63–70, Stroudsburg, PA, USA. ACL.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

Ludo Waltman and Nees Jan van Eck. 2012. The inconsistency of the h-index. *JASIST*, 63(2):406–415, February.

Xiaojun Wan and Fang Liu. 2014. Are all literature citations equally important? automatic citation strength estimation and its applications. *JASIST*, 65(9):1929–1938.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Erjia Yan and Ying Ding. 2010. Weighted citation: An indicator of an article's prestige. *JASIST*, 61(8):1635–1643.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, Washington D.C.

Xiaodan Zhu, Peter Turney, Daniel Lemire, and Andr Vellino. 2015. Measuring academic influence: Not all citations are equal. *JASIST*, 66(2):408–427.

Improving Users' Demographic Prediction via the Videos They Talk about

Yuan Wang, Yang Xiao, Chao Ma, and Zhen Xiao
Department of Computer Science, Peking University, Beijing 100871, China
{wangyuan, xiaoyang, machao, xiaozhen}@net.pku.edu.cn

Abstract

In this paper, we improve microblog users' demographic prediction by fully utilizing their video related behaviors. First, we collect the describing words of currently popular videos, including video names, actor names and video keywords, from video websites. Secondly, we search these describing words in users' microblogs, and build the direct relationships between users and the appeared words. After that, to make the sparse relationship denser, we propose a Bayesian method to calculate the probability of connections between users and other video describing words. Lastly, we build two models to predict users' demographics with the obtained direct and indirect relationships. Based on a large real-world dataset, experiment results show that our method can significantly improve these words' demographic predictive ability.

1 Introduction

Recent studies have indicated that users' demographics can be predicted from their linguistic characteristics. A typical practice is cutting the text into a bag of words and training a linear classifier. Although this practice can achieve an acceptable result in simple tasks such as predicting gender and age, it loses some important information about the text structure and does not fully use the relationship between words.

Nowadays, people spend a lot of time on videos and social media which provide them with access to post views and comments. Weibo is one of the biggest microblogging platforms in China. More than one third of the "Weibo Trends"[1] are about videos. Generally, people with different demographic attributes usually have different tastes for videos (Abisheva et al., 2014). For example, in China people who watch English drama tend to be well-educated. Here is a question: if the video related information in users' weibo messages can be fully used, will the users' demographic prediction be improved?

One challenge is that many users do not directly mention the video names in their weibo messages. Instead, they make comments on the actors or the plots. If a person likes "Big Bang Theory", he may post "*Will the Big Bang Theory last into the next century?*" where the sitcom's name is mentioned directly, or "*Sheldon is so cool, I love him!*" which talks about an actor of the sitcom. Both posts indicate the user is interested in "Big Bang Theory". When involving the demographic prediction, however, the traditional "bag of words based" model cannot extract the above information effectively. Some previous works use topic models such as LIWC (Pennebaker et al., 2001) or LDA (Blei et al., 2003) to detect the relations among users' words. Usually, they suffer from the short length of weibo messages and the number of topics. In addition, the lifespan of most popular video programs is not very long, which renders traditional topic models inefficient.

Fortunately, there exist some third-party video websites, such as *youtube.com* and *youku.com*, from which we can get the most popular videos. For each video, there is usually a homepage with a actor list

[1] http://d.weibo.com/100803

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1359–1368,
Austin, Texas, November 1-5, 2016.

and also a comments section, and we can calculate the video's Top TF-IDF words (keywords) based on these comments. Here we define the *video name*, *actor name* and *keyword* to be three different kinds of "video describing words". The relationships among these words can be used to better understand weibo users' video related behaviors. This approach can be applied to other kinds of words, such as describing words on books and music. This paper focuses on the video as an example.

After obtaining the video describing words, we build three matrices to represent the direct and indirect relationships between weibo users and these words. They are User-Video Matrix, User-Actor Matrix and User-Keyword Matrix, respectively. At beginning, these three matrices are sparse because they only represent the direct relationships, which means that only when the words appear in user's weibos, the corresponding position will be set. After that, we propose a "hidden layer" to detect the indirect relationships, making them denser.

With these indirect relationships, we can improve users' demographic predictions, including *gender*, *age*, *education background*, and *marital status*. This paper makes the followings three contributions:

1. By construct three matrices, we detect the direct and indirect relationships between weibo users and video describing words.

2. Two models are proposed to predict users' demographics by using both direct and indirect relationships.

3. Experiment results prove that our efforts can significantly improve the predictive accuracy, compared with the existing research.

The rest of this paper is organized as follows. Section 2 introduces the dataset and demographics. Section 3 introduces how to make full use of video related behaviors. Section 4 presents experimental results. Finally, we review related work in Section 5, and draw conclusions in Section 6.

2 Dataset and Demographics

2.1 Dataset

We collected 2,970,642 microblog users from Weibo (http://weibo.com), the largest microblog service in China, as our dataset. To avoid spam users (sometimes called robot users), we only collected *verified users* and *users followed by verified user*. Weibo conducts manual verifications to make sure the verified users provide real and authentic profile information. Table 1 presents four target demographic attributes and the completion rates (ratio of effective users). All data is either through Open API or publicly available. No private data is used in the experiment.

We also collected 847 popular video programs from YISQ (4 popular video websites in China: *youku*, *iqiyi*, *sohu*, *qq*). These videos mainly fall into three types: movie, tv play, and variety shows. We downloaded these videos' Homepages and extracted their actors and TOP20 TF-IDF words. The statistics are shown in Table 2.

2.2 Ground Truth

One problem of our dataset is it contains celebrities, while our model mainly targets ordinary weibo users. We implement a filter to exclude celebrities based on their large numbers of followers (>50000 as default), making the ground truth more representative. Besides, users with less than 100 messages are discarded. At last, we obtain 742,323 accounts with both their demographics and messages.

2.3 Demographics

As Table 1 shows, the demographic attributes concerned in this paper include gender, age, education background, and marital status:

Gender (Binary): the gender prediction is a typical binary classification task: male, female.

Age (4-Class): because there is only a handful of(<1%) user older than 45, we classify users into the following four age groups: Teenage (<18), Youngster (18-24), Young (25-34), Mid-age (>34).

Education Background (Binary): we categorize users' education background into two groups: university, non-university.

Marital Status (Binary): marital status is also simplified to a binary classification task: single, non-single.

3 Our Model

In this section, we introduce the framework, which contains four steps.

Attribute	Completion Rate	Categories
Gender	95.019%	Male, Female
Age	18.604%	Teenage (<18), Youngster (18-24), Young (25-34), Mid-age(>34)
Education BG	17.443%	University, Non-University
Marital Status	2.203%	Single, Non-Single

Table 1: Demographic attributes and corresponding categories

	Video	Actor	Keyword
Variety show	344	1007	2925
Movie	306	741	2049
TV	197	515	1302
Total	**847**	1422	4094

Table 2: Statics of video relevant information (There is an overlap between the three collections of actors and keywords.)

The first step generates the "Video describing words" and represents user as two vectors (V_v, V_o). V_v consists of user's "video describing words" and V_o consists of user's "other words". At first, V_v only contains user's direct relationships.

> *V_v: video describing words (direct)*
> *V_o: other words*
> *V_a: V_v + V_o*

The second step detects the indirect relationships between users and videos. For example, if a user mentioned "Robert Downey Jr", we believe he has an indirect relationships with "Iron Man" movie. By doing so, we add user's indirect relationships into his V_v, getting a denser vector V_v'.

> *V_v': video describing words (direct+indirect)*
> *V_a': V_v' + V_o*

The third step proposes two models respectively to evaluate whether those indirect relationships, discovered in second step, can be used to develop a more accurate prediction model.

The fourth step represents weibo user with the combination of V_v' and V_o, and use the combination to train a linear SVM to evaluate whether this effort can make the prediction better.

3.1 Discover Indirect Relationships

If a user mentioned a video's name directly, we believe there is a direct relationship between them. The rests are unobvious relationships. In this part, we calculate whether these unobvious relationships can be transformed into indirect ones.

3.1.1 User-Video Matrix

Firstly, we detect whether a user directly mentioned a video program in his weibo messages. There are two scenarios: the first is this user posts a message containing the video's name directly, and the other is this user reposts a message containing the video's name. In this paper, we believe these two scenarios both indicate there is a direct relationship between the user and the video, and do not make a distinction between them. Till now, we construct a Direct User-Video Matrix (DUVM) to denote all the direct relationships between users and videos.

Step 1: We know each video program v_n contains some actors a_{nj} and keywords w_{ni}. We can calculate $P(v_n)$, $P(a_{nj}|v_n)$ and $P(w_{ni}|v_n)$ in Step 1. $P(v_n)$ represents the probability that a person has watched the n_{th} video. $P(w_{ni}|v_n)$ represents the probability that a person, who has watched the n_{th} video, mention the ni_{th} keyword. $P(a_{nj}|v_n)$ is the probability that a person, who has watched the n_{th} video, mention the nj_{th} actor.

$P(v_n)$ = num (users watched the n_{th} video) / num (users)

$P(w_{ni}|v_n)$ = num (users watched the n_{th} video and mentioned the ni_{th} keyword) / num (users watched the n_{th} video)

$P(a_{nj}|v_n)$ = num (users watched the n_{th} video and mentioned the nj_{th} actor) / num (users watched the n_{th} video)

Step 2: In step 2, If a user doesn't mention a video's name directly, but mentions the video's related actors (A_k) and keywords (W_m), we can update his unobvious user-video relationships according to a Bayesian framework.

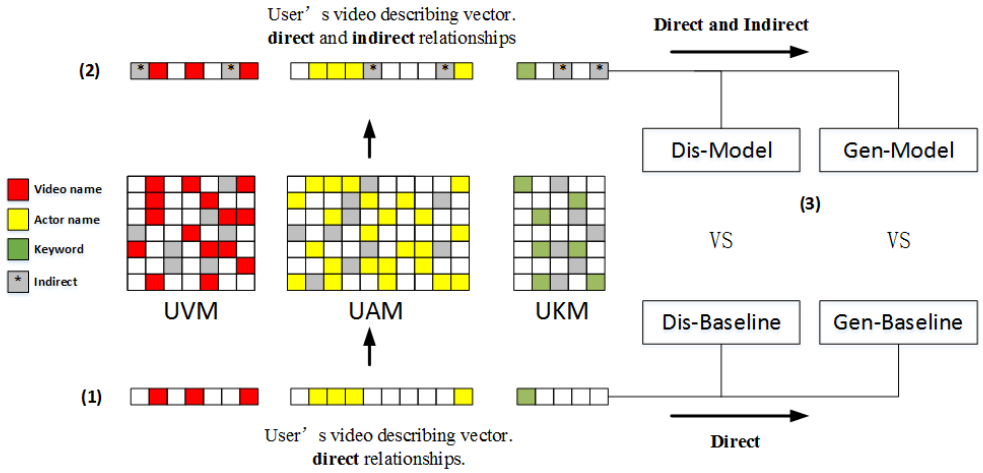


Figure 1: (1) At first, identify the describing words from users microblogs, which builds the direct relationships between users and these words. (2) By construct three matrices, we detect the indirect relationships between weibo users and video describing words. (3)Two models are proposed to predict users demographics by using both direct and indirect relationships.

$$P(v_n|W_m, A_k) = \frac{P(W_m, A_k|v_n) * P(v_n)}{P(W_m, A_k)}$$
$$= \frac{\prod_{w_{ni} \in W_m} P(w_{ni}|v_n) * \prod_{a_{nj} \in A_k} P(a_{nj}|v_n) * P(v_n)}{P(W_m, A_k)} \quad (1)$$

Through Step 2, we can discover some new indirect relationships and update UVM. Go back to Step 1 and iterate until converges, we can get the Final UVM at last.

3.1.2 User-Actor Matrix

Every video program has several actors, and the relationships between weibo users and actors may contribute to the demographic prediction either. So we build the UAM, where each row represents a weibo user and each column represents an actor.

There are two case that the element of UAM will be set to true: (1) the user 'i' directly mentioned actor 'j' in his weibo messages (including post and repost); (2) the user 'i' has watched video 'v', and actor 'j' participate in video 'v'. The second case needs UVM's help. We suppose these two cases affect the value equally in this paper.

3.1.3 User-Keyword Matrix

We can find several keywords to describe each video from their Homepages. For instance, we get "Paul Walker", "fight", and "car" to describe "Furious 7".

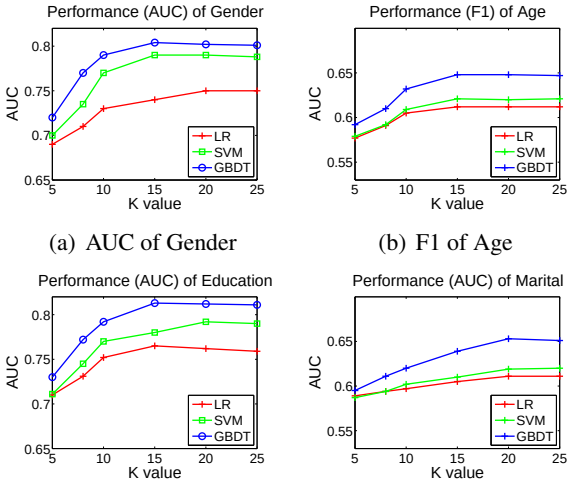
Each row of UKM represents a weibo user and each column represents a keyword of a certain video. (1) If we find a user has watched the "Furious 7", no matter direct or indirect relationship, we can set the columns of user's "Furious 7" keywords to true. (2) The value can be set to true either if the user directly mentioned these keywords.

3.2 Two Indirect Relationship Based Models

In this part, two models are proposed to predict users' demographics by using both direct and indirect relationships.

3.2.1 Discriminant Model (Dis-Model)

Given three matrices, the intuitive way to predict users' demographics is using Collaborative Filtering. However, finding the similar users directly based on the vector similarity is not a good idea, because a substantial part of users have ever watched no more than 10 videos. Matrix Factorization has been proven useful to address data sparsity, for the reduced orthogonal dimensions are less noisy than the original data and can capture the latent associations between users and videos. In our Dis-Model, we utilize the factorization machines (Rendle, 2010) to deal with UVM, UAM, and UKM, reducing the length of user's dimensionality from videos' number (actors' number, keywords'



(a) AUC of Gender (b) F1 of Age

(c) AUC of Education BG (d) AUC of Marital Status

Figure 2: Performance of different classifiers (LR, SVM, GBDT) for Dis-Model with varying K.

number) to a smaller value K. Every weibo user can be represented by the combination of these three K-length vectors.

Over the last several decades, many kinds of discriminant classifier have been created. For our four tasks, we compared Logistic Regression (LR), Support Vector Machine (SVM), and Gradient Boosted Decision Tree (GBDT). Figure 2 illustrates their performance, where GBDT performs the best in all K values. When K increases from 5 to 20, all classifiers' results are all getting better and tend to be stable when K is bigger than 20. So we choose GBDT as our default base classifier and K=20 as default value.

3.2.2 Generative Model (Gen-Model)

We start with introducing an important concept: video demographic tendency, which means to what extent a video belongs to a specified demographic group. For example, if 90% audiences of a movie are males, we define its demographic tendency on male as 90%. The actor tendency and keyword tendency can be calculated in the same way.

In the Gen-Model, (1) we firstly calculate each video's (actor, keyword) demographic tendency according to its audiences (known demographics). (2) Based on the demographic tendency of videos (actors, keywords), we predict user's (unknown) demographics via a Bayesian method. (3) At last, we propose a smooth step to adjust the result.

(1) Calculate video demographic tendency

At first, we calculate every video demographic tendency as Equation 2:

$$p(c|v_j) = \frac{\sum_{i=1}^{n}(r_{ij} * u_i(c))}{\sum_{i=1}^{n} r_{ij}} \tag{2}$$

$P(c|v_j)$ represents the j_{th} video's demographic tendency on c, where c is the demographic attribute. r_{ij} will be set to 1 if the i_{th} user has watched the j_{th} video, otherwise set to 0. $u_i(c)$ is a boolean, representing whether the i_{th} user has the attribute c.

(2) Calculate user demographic attribute

In this step, we predict users' demographics according to the demographic tendency of the videos they has watched. Suppose user's viewing habits are independent, we can calculate the probability of $P(c|u_i)$ as Equation 3:

$$\begin{aligned} P(c|u_i) &\propto P(c|\{V\}) \\ &\propto P(\{V\}|c) * P(c) \\ &\propto \prod_{v_j \in \{V\}} P(v_j|c) * P(c) \\ &= \frac{\prod_{v_j \in \{V\}} P(c|v_j) * P(v_j)}{P(c)} * P(c) \\ &\propto \prod_{v_j \in \{V\}} P(c|v_j) \end{aligned} \tag{3}$$

$\{V\}$ represents the collection of videos watched by u_i. $P(c|v_j)$ is the j_{th} video's demographic tendency on c, as the previous part described.

(3) Smooth the result

Based on the fact that people in same demographic group may have similar behaviors, we deploy a smooth component to adjust the value of $P(c|v_j)$ and $P(c|u_i)$ according to their top *n* neighbors. As mentioned above, we use factorization machines to transform the user and video vectors into low-dimensional (K=20) ones. The distance is calculated by Euclidean Distance. The video, actor, and word have the same treating process, so we introduce the video as representative.

Smooth the Video's Demographic Tendency: Base on video v_j's top n neighbors, we can calculate its neighbors' average demographic tendency $P(c|nbr(v_j))$, where $P(c|v_{nbj})$ is v_j's nbj_{th} neighbor's demographic tendency.

$$p(c|nbr(v_j)) = \frac{\sum_{j=1}^{n} P(c|v_{nbj})}{n} \quad (4)$$

Therefore, we can smooth v_j's demographic tendency by:

$$P(c|v_j) = \alpha * P(c|v_j) + (1-\alpha) * P(c|nbr(v_j)) \quad (5)$$

α is the parameter to control the top n neighbors' influence. In this paper, we compared ten values of α and chose 0.7 as default. With the same process, n is set to 10 as default.

Smooth the User's Demographic Result: The user side smooth procedure is similar to the video side, except user's $P(c|nbr(u_i))$ is affected by three kinds of neighbors (u_{nbvi}, u_{nbai}, u_{nbwi}).

$$p(c|nbr(u_i)) = \frac{\sum_{i=1}^{n} P(c|u_{nbvi})}{3n} + \frac{\sum_{i=1}^{n} P(c|u_{nbai})}{3n} + \frac{\sum_{i=1}^{n} P(c|u_{nbwi})}{3n} \quad (6)$$

Just like video's smooth process, we adjust u_i's demographic attributes by:

$$P(c|u_i) = \alpha * P(c|u_i) + (1-\alpha) * P(c|nbr(u_i)) \quad (7)$$

The smooth component is deployed as an iterative procedure, and keeps running until each $P(c|u_i)$ became stable.

Two Baselines: To validate whether those indirect relationships can improve the predictions, we build two baseline models: Dis-Baseline and Gen-Baseline. While our two models use the V_v' as input, these two baseline models use the raw V_v. These two baseline models adopt the same architecture with our proposed two models. The only difference is the input data.

3.3 Fusion Model

As described above, we discovered the indirect relationships between users and video describing words, and demonstrated this effort can leading a better result than directly train the classifier.

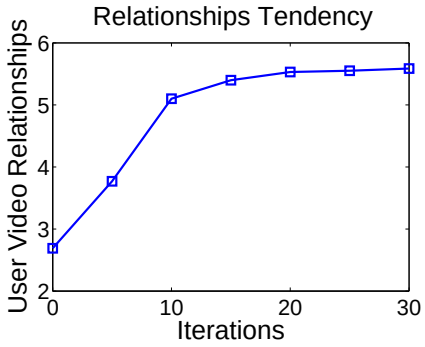


Figure 3: Tendency of User-Video relationship number.

But pre-existing models commonly utilize all the words in user's weibo messages. So we need to find out whether our hard-earned improvement would be submerged by those "Non video describing words". We train a Fusion Model using all the words in weibo messages and indirect relationships together, and compare it with a baseline model, who only use all the words (without indirect relationships).

Fusion Baseline: Many pre-existing methods (Burger et al., 2011; Tu et al., 2015) chose linear model as their text classifier, for linear model is suitable for text categorization tasks. We choose L1-regularized linear SVM as our Fusion Model and Fusion-Baseline's classifier. The only difference between them is the input data (V_v' + V_o **vs** V_v + V_o).

4 Experiment Results

We conducted a 10-fold cross validation to demonstrate our framework's effectiveness, where 8 parts for training, 1 parts for validation and 1 parts for testing by default. The performance of presented methods were evaluated using the Precision, Recall and Macro-F1 measures. Binary classification tasks were also measured by Area Under the ROC Curve (AUC).

4.1 Indirect Relationships Evaluation

In our dataset, each user directly mention 2.6 video programs on average and only 0.7% has more than 10 direct relationships. As shown in Figure 3, more and more indirect relationships arise along with the iterations. User's relationship number (direct + indirect) stabilized at 5.7 on average and 13% of them is bigger than 10.

To answer whether these indirect relationships

		Precision	Recall	F1	AUC
Gender	Dis-Baseline	0.720	0.714	0.717	0.730
	Dis-Model	0.786	0.779	0.783	**0.812** ↑ 11.2%
	Gen-Baseline	0.701	0.687	0.694	0.707
	Gen-Model	0.799	0.802	0.801	**0.825** ↑ 16.7%
Age	Dis-Baseline	0.569	0.541	0.554	*
	Dis-Model	0.642	0.653	**0.648** ↑ 16.8%	*
	Gen-Baseline	0.529	0.504	0.516	*
	Gen-Model	0.663	0.645	**0.654** ↑ 26.7%	*
Education BG	Dis-Baseline	0.707	0.716	0.711	0.730
	Dis-Model	0.788	0.801	0.795	**0.809** ↑ 11.1%
	Gen-Baseline	0.680	0.659	0.669	0.690
	Gen-Model	0.790	0.808	0.799	**0.812** ↑ 17.7%
Marital Status	Dis-Baseline	0.565	0.549	0.557	0.571
	Dis-Model	0.657	0.640	0.648	**0.659** ↑ 15.4%
	Gen-Baseline	0.572	0.550	0.560	0.581
	Gen-Model	0.682	0.691	0.687	**0.696** ↑ 19.8%

Table 3: Prediction accuracy based on users' video describing words. Classes have been balanced.

can make the prediction better, we compared our two models (Dis-Model & Gen-Model) with two baseline models. We also compared their performance on different user groups categorized by user-video relationship number.

Gender: As Table 3 shows, our two models both have a significant improvement compared to the baseline models. The Gen-Model achieve the best performance (AUC 0.825) in terms of all the measurement. As Figure 4(a) shows, with the number growth, our two models' AUC scores are both getting better. Surprisingly, when the number is bigger than 10, the Gen-Model even get a similar performance of the model using all of the user's words.

Age: In the age task, our two models both outperformed the baseline models significantly, and the generative model performs better (F1 0.654) too. We analyzed the result and found the "youngster" and "young" share the similar watching habits in Weibo. It's hard to pick out a 23 years old user from the 28 years old group. As Figure 4(b) shows, our two models' F1 scores are both getting better along with the growth of user-video relationship number.

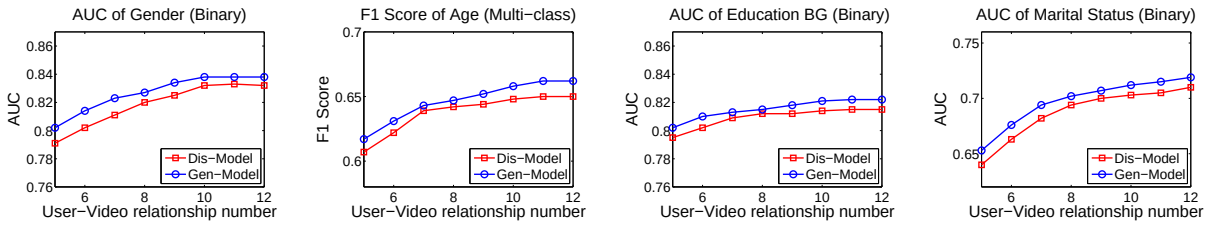
Education Background: Not surprisingly, our two models obviously outperform the result over two baseline models. This result indicates that people in different education background has visible different tastes on video programs.

Marital Status: Table 3 presents the results of marital status. We notice that the performance of our two model is still reasonable, but is worse than gender and education tasks. In addition to that this task is more difficulty, another reason is when a user gets married, he might not update the information in his online profile.

Remark: Experiment results show that our method can significantly improve these words' demographic predictive ability by more than 15% on average. 10 videos is good enough to portray a weibo user, and can achieve reasonable results in these 4 inference tasks. The video related behavior is efficient on predicting gender and education, for people on these two tasks have visible different inclinations. Inferring age and marital status is not easy, but our two models still achieve reasonable improvements. In general, our two models both get significantly better results than baselines. The Gen-Model is a better choice by contrast.

4.2 Fusion Model Evaluation

After we obtained the potential predictive ability of indirect relationships, we also need to find out whether it can help pre-existing model perform



(a) AUC of Gender (b) F1 Score of Age (c) AUC of Education BG (d) AUC of Marital Status

Figure 4: Prediction result with varying User-Video relationship numbers.

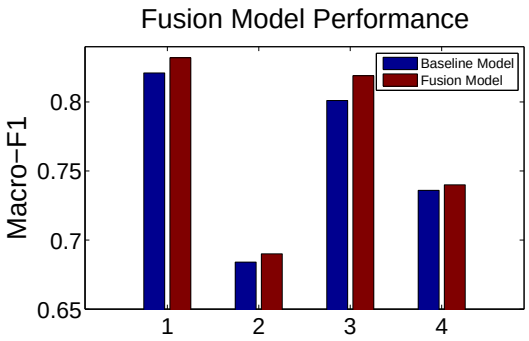


Figure 5: Results of Fusion Model evaluation (Macro-F1).

better. We compare the Fusion Baseline (V_v+V_o) with our Fusion Model (V_v'+V_o). As Figure 5 shows, Fusion Model's performance is better than Fusion Baseline's in all four tasks. The improvement is about 2-3% on average. As above mentioned, our approach can be applied to other kinds of words, such as describing words on books and music. So there is some room for improvement.

5 Related work

In this section, we briefly review the research works related to our work.

Many researches (Kumar and Tomkins, 2010; Goel et al., 2012) found users belong to different demographic groups behave differently. (Hu et al., 2007; Murray and Durrell, 2000; Goel et al., 2012; Kosinski et al., 2012) showed that age, gender, education level, and even personality can be predicted from people's webpage browsing logs. (Kosinski et al., 2013; Schwartz et al., 2013; Youyou et al., 2015) showed computers' judgments of people's personalities based on their Facebook Likes are more accurate and valid than judgments made by their close acquaintances. (Malmi and Weber, 2016) showed users' demographics also can be predicted based on theirs apps. Apart from the browsing behaviors, there also exist some works based on user's linguistic characteristics. (Schler et al., 2006) analyzed tens of thousands of blogs and indicated significant differences in writing style and word usage between different gender and age groups. The similar result also showed in (Luyckx and Daelemans, 1998; Oberlander and Nowson, 2006; Mairesse et al., 2007; Nowson, 2007; Gill et al., 2009; Rosenthal and McKeown, 2011). There are some works (Bi et al., 2013; Weber and Jaimes, 2011; Weber and Castillo, 2010) on predicting search engine user's demographics based on their search queries. (Hovy, 2015) investigated the influence of user's demographics on better understanding their online reviews. (Otterbacher, 2010) used logistic regression model to infer users gender based on the content of movie reviews.

Many researches focused on the twitter users. In the Author Profiling task at PAN 2015 (Rangel et al., 2015), participants approached the task of identifying age, gender and personality traits from Twitter. (Nguyen et al., 2013) explored users' age prediction task based on their tweets, achieving better performance than humans. (Burger et al., 2011) studied the gender predictive ability of twitter linguistic characteristics, reached 92% accuracy. (Pennacchiotti and Popescu, 2011) proposed a GB-DT model to predict users' age, gender, political orientation and ethnicity by leveraging their observable information. (Culotta et al., 2015) predicted the demographics of Twitter users based on whom they follow, and (Zhong et al., 2015) predicted the microblog user's demographic attributes only by their chick-ins. In (Li et al., 2014), job and education attributes are extracted by combining a rule based approach with a probabilistic system. There are also some works based on users' social relationships

(Mislove et al., 2010; Henderson et al., 2012; Zhao et al., 2013).

6 Conclusion

Our motivation on writing this paper is user's video related behavior is usually under-utilized on demographic prediction tasks. With the help of third-party video sites, we detect the direct and calculate the indirect relationships between users and video describing words, and demonstrate this effort can improve the accuracy of users' demographic predictions. To our knowledge, this is the first work which explores demographic prediction by fully using users' video describing words. This framework has good scalability and can be applied on other concrete features, such as user's book reading behaviors and music listening behaviors.

Acknowledgments

This work was supported by the National Grand Fundamental Research 973 Program of China under Grant No.2014CB340405 and the National Natural Science Foundation of China under Grant No.61572044.

References

Adiya Abisheva, Venkata Rama Kiran Garimella, David Garcia, and Ingmar Weber. 2014. Who watches (and shares) what on youtube? and when?: using twitter to understand youtube viewership. In *Proceedings of WSDM*, pages 593–602. ACM.

Bin Bi, Milad Shokouhi, Michal Kosinski, and Thore Graepel. 2013. Inferring the demographics of search users: Social data meets search queries. In *Proceedings of WWW*, pages 131–140. International World Wide Web Conferences Steering Committee.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the EMNLP*, pages 1301–1309. Association for Computational Linguistics.

Aron Culotta, Nirmal Ravi Kumar, and Jennifer Cutler. 2015. Predicting the demographics of twitter users from website traffic data. In *Proceedings of AAAI*, pages 72–78.

Alastair J Gill, Scott Nowson, and Jon Oberlander. 2009. What are they blogging about? personality, topic and motivation in blogs. In *Proceedings of ICWSM*.

Sharad Goel, Jake M Hofman, and M Irmak Sirer. 2012. Who does what on the web: A large-scale study of browsing behavior. In *Proceedings of ICWSM*.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *Proceedings of SIGKDD*, pages 1231–1239. ACM.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of ACL*.

Jian Hu, Hua-Jun Zeng, Hua Li, Cheng Niu, and Zheng Chen. 2007. Demographic prediction based on user's browsing behavior. In *Proceedings of WWW*, pages 151–160. ACM.

Michal Kosinski, David Stillwell, Pushmeet Kohli, Yoram Bachrach, and Thore Graepel. 2012. Personality and website choice.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Ravi Kumar and Andrew Tomkins. 2010. A characterization of online browsing behavior. In *Proceedings of WWW*, pages 561–570. ACM.

Jiwei Li, Alan Ritter, and Eduard H Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of ACL*, pages 165–174.

Kim Luyckx and Walter Daelemans. 1998. Using syntactic features to predict author personality from text. *Science*, 22:319–346.

Francois Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*, pages 457–500.

Eric Malmi and Ingmar Weber. 2016. You are what apps you use: Demographic prediction based on user's apps. *arXiv preprint arXiv:1603.00059*.

Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. 2010. You are who you know: inferring user profiles in online social networks. In *Proceedings of WSDM*, pages 251–260. ACM.

Dan Murray and Kevan Durrell. 2000. Inferring demographic attributes of anonymous internet users. In *Web Usage Analysis and User Profiling*, pages 7–20. Springer.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "how old do you think i am?"; a study of language and age in twitter. In *Proceedings of ICWSM*. AAAI Press.

Scott Nowson. 2007. Identifying more bloggers: Towards large scale personality classification of personal weblogs. In *Proceedings of ICWSM*. Citeseer.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway?: classifying author personality from weblog text. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 627–634. Association for Computational Linguistics.

Jahna Otterbacher. 2010. Inferring gender of movie reviewers: exploiting writing style, content and metadata. In *Proceedings of CIKM*, pages 369–378. ACM.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A machine learning approach to twitter user classification. In *Proceedings of ICWSM*, pages 281–288.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Francisco Rangel, Fabio Celli, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd Author Profiling Task at PAN 2015. In Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan, editors, *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France*. CEUR-WS.org, September.

Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre-and post-social media generations. In *Proceedings of ACL*, pages 763–772. Association for Computational Linguistics.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, pages 199–205.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

Cunchao Tu, Zhiyuan Liu, and Maosong Sun, 2015. *Social Media Processing: 4th National Conference, SMP 2015, Guangzhou, China, November 16-17, 2015, Proceedings*, chapter PRISM: Profession Identification in Social Media with Personal Information and Community Structure, pages 15–27. Springer Singapore, Singapore.

Ingmar Weber and Carlos Castillo. 2010. The demographics of web search. In *Proceedings of SIGIR*, pages 523–530. ACM.

Ingmar Weber and Alejandro Jaimes. 2011. Who uses web search for what: and how. In *Proceedings of WSDM*, pages 15–24. ACM.

Wu Youyou, Michal Kosinski, and David Stillwell. 2015. Computer-based personality judgments are more accurate than those made by humans. *Proceedings of the National Academy of Sciences*, 112(4):1036–1040.

Yuchen Zhao, Guan Wang, Philip S Yu, Shaobo Liu, and Simon Zhang. 2013. Inferring social roles and statuses in social networks. In *Proceedings of SIGKDD*, pages 695–703. ACM.

Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. 2015. You are where you go: Inferring demographic attributes from location checkins. In *Proceedings of WSDM*, pages 295–304. ACM.

AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding

Xiang Ren[†*] **Wenqi He**[†*] **Meng Qu**[†] **Lifu Huang**[♯] **Heng Ji**[♯] **Jiawei Han**[†]

[†] University of Illinois at Urbana-Champaign, Urbana, IL, USA

[♯] Computer Science Department, Rensselaer Polytechnic Institute, USA

[†]{xren7, wenqihe3, mengqu2, hanj}@illinois.edu [♯]{huangl7, jih}@rpi.edu

Abstract

Distant supervision has been widely used in current systems of fine-grained entity typing to automatically assign categories (entity types) to entity mentions. However, the types so obtained from knowledge bases are often incorrect for the entity mention's local context. This paper proposes a novel embedding method to separately model "clean" and "noisy" mentions, and incorporates the given type hierarchy to induce loss functions. We formulate a joint optimization problem to learn embeddings for mentions and type-paths, and develop an iterative algorithm to solve the problem. Experiments on three public datasets demonstrate the effectiveness and robustness of the proposed method, with an average 15% improvement in accuracy over the next best compared method[1].

1 Introduction

Assigning types (*e.g.*, `person`, `organization`) to mentions of entities in context is an important task in natural language processing (NLP). The extracted entity type information can serve as primitives for relation extraction (Mintz et al., 2009) and event extraction (Ji and Grishman, 2008), and assists a wide range of downstream applications including knowledge base (KB) completion (Dong et al., 2014), question answering (Lin et al., 2012) and entity recommendation (Yu et al., 2014). While traditional named entity recognition systems (Ratinov and Roth, 2009; Nadeau and Sekine, 2007) focus on a small set of coarse types (typically fewer than 10), recent studies (Ling and Weld, 2012; Yosef et al., 2012) work on a much larger set of fine-grained types (usually over 100) which form a tree-structured hierarchy (see the blue region of Fig. 1). Fine-grained typing allows one mention to have multiple types, which together constitute a *type-path* (not necessarily ending in a leaf node) in the given type hierarchy, *depending on the local context* (*e.g.*, sentence). Consider the example in Fig. 1, "*Arnold Schwarzenegger*" could be labeled as {`person`, `businessman`} in S3 (investment). But he could also be labeled as {`person`, `politician`} in S1 or {`person`, `artist`, `actor`} in S2. Such fine-grained type representation provides more informative features for other NLP tasks. For exam-

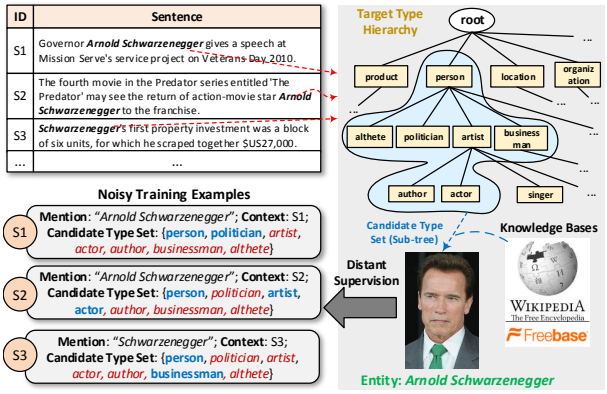


Figure 1: Current systems may detect *Arnold Schwarzenegger* in sentences S1-S3 and assign the same types to all (listed within braces), when only some types are correct for context (blue labels within braces).

[*]Equal contribution.

[1]Codes and datasets used in this paper can be downloaded at `https://github.com/shanzhenren/AFET`.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1369–1378,
Austin, Texas, November 1-5, 2016.

ple, since relation and event extraction pipelines rely on entity recognizer to identify possible arguments in a sentence, fine-grained argument types help distinguish hundreds or thousands of different relations and events (Ling and Weld, 2012).

Traditional named entity recognition systems adopt manually annotated corpora as training data (Nadeau and Sekine, 2007). But the process of manually labeling a training set with large numbers of fine-grained types is too expensive and error-prone (hard for annotators to distinguish over 100 types consistently). Current fine-grained typing systems annotate training corpora automatically using knowledge bases (*i.e.*, *distant supervision*) (Ling and Weld, 2012; Ren et al., 2016a). A typical workflow of distant supervision is as follows (see Fig. 1): (1) identify entity mentions in the documents; (2) link mentions to entities in KB; and (3) assign, to the candidate type set of each mention, all KB types of its KB-linked entity. However, existing distant supervision methods encounter the following limitations when doing automatic fine-grained typing.

• **Noisy Training Labels.** Current practice of distant supervision may introduce *label noise* to training data since it fails to take a mention's local contexts into account when assigning type labels (*e.g.*, see Fig. 1). Many previous studies ignore the label noises which appear in a majority of training mentions (see Table. 1, row (1)), and assume *all* types obtained by distant supervision are "correct" (Yogatama et al., 2015; Ling and Weld, 2012). The noisy labels may mislead the trained models and cause negative effect. A few systems try to denoise the training corpora using simple pruning heuristics such as deleting mentions with conflicting types (Gillick et al., 2014). However, such strategies significantly reduce the size of training set (Table 1, rows (2a-c)) and lead to performance degradation (later shown in our experiments). The larger the target type set, the more severe the loss.

• **Type Correlation.** Most existing methods (Yogatama et al., 2015; Ling and Weld, 2012) treat every type label in a training mention's candidate type set *equally* and *independently* when learning the classifiers but ignore the fact that types in the given hierarchy are semantically correlated (*e.g.*, `actor` is more relevant to `singer` than to `politician`). As a consequence, the learned classifiers may bias toward popular types but perform poorly on infrequent types since training data on infrequent types is scarce. Intuitively, one should pose smaller penalty on types which are semantically more relevant to the true types. For example, in Fig. 1 `singer` should receive a smaller penalty than `politician` does, by knowing that `actor` is a true type for "*Arnold Schwarzenegger*" in S2. This provides classifiers with additional information to distinguish between two types, especially those infrequent ones.

Dataset	Wiki	OntoNotes	BBN	NYT
# of target types	113	89	47	446
(1) noisy mentions (%)	27.99	25.94	22.32	51.81
(2a) sibling pruning (%)	23.92	16.09	22.32	39.26
(2b) min. pruning (%)	28.22	8.09	3.27	32.75
(2c) all pruning (%)	45.99	23.45	25.33	61.12

Table 1: A study of label noise. (1): %mentions with multiple *sibling types* (*e.g.*, `actor`, `singer`); (2a)-(2c): %mentions deleted by the three pruning heuristics (2014) (see Sec. 4), for three experiment datasets and New York Times annotation corpus (2014).

In this paper, we approach the problem of *automatic fine-grained entity typing* as follows: (1) Use different objectives to model training mentions with correct type labels and mentions with noisy labels, respectively. (2) Design a novel *partial-label loss* to model true types within the noisy candidate type set which requires only the "*best*" candidate type to be relevant to the training mention, and progressively estimate the best type by leveraging various text features extracted for the mention. (3) Derive type correlation based on two signals: (i) the given type hierarchy, and (ii) the shared entities between two types in KB, and incorporate the correlation so induced by enforcing *adaptive margins* between different types for mentions in the training set. To integrate these ideas, we develop a novel embedding-based framework called **AFET**. First, it uses distant supervision to obtain candidate types for each mention, and extract a variety of text features from the mentions themselves and their local contexts. Mentions are partitioned into a "clean" set and a "noisy" set based on the given type hierarchy. Second, we embed mentions and types jointly into a low-dimensional space, where, in that space, objects (*i.e.*, features and types) that are semantically close to each other also have similar representations. In the proposed objective, an adaptive margin-based rank loss is pro-

posed to model the set of clean mentions to capture type correlation, and a partial-label rank loss is formulated to model the "best" candidate type for each noisy mention. Finally, with the learned embeddings (*i.e.*, mapping matrices), one can predict the type-path for each mention in the test set in a top-down manner, using its text features. The major contributions of this paper are as follows:

1. We propose an automatic fine-grained entity typing framework, which reduces label noise introduced by distant supervision and incorporates type correlation in a principle way.
2. A novel optimization problem is formulated to jointly embed entity mentions and types to the same space. It models noisy type set with a partial-label rank loss and type correlation with adaptive-margin rank loss.
3. We develop an iterative algorithm for solving the joint optimization problem efficiently.
4. Experiments with three public datasets demonstrate that AFET achieves significant improvement over the state of the art.

2 Automatic Fine-Grained Entity Typing

Our task is to automatically uncover the type information for entity mentions (*i.e.*, token spans representing entities) in natural language sentences. The task takes a document collection $\mathcal{D}$ (automatically labeled using a KB Ψ in conjunction with a target type hierarchy $\mathcal{Y}$) as input and predicts a type-path in $\mathcal{Y}$ for each mention from the test set $\mathcal{D}_t$.

Type Hierarchy and Knowledge Base. Two key factors in distant supervision are the target type hierarchy and the KB. A *type hierarchy*, $\mathcal{Y}$, is a tree where nodes represent types of interests from Ψ. Previous studies manually create several clean type hierarchies using types from Freebase (Ling and Weld, 2012) or WordNet (Yosef et al., 2012). In this study, we adopt the existing hierarchies constructed using Freebase types[2]. To obtain types for entities $\mathcal{E}_\Psi$ in Ψ, we use the human-curated entity-type facts in Freebase, denoted as $\mathcal{F}_\Psi = \{(e, y)\} \subset \mathcal{E}_\Psi \times \mathcal{Y}$.

[2]We use the Freebase dump as of 2015-06-30.

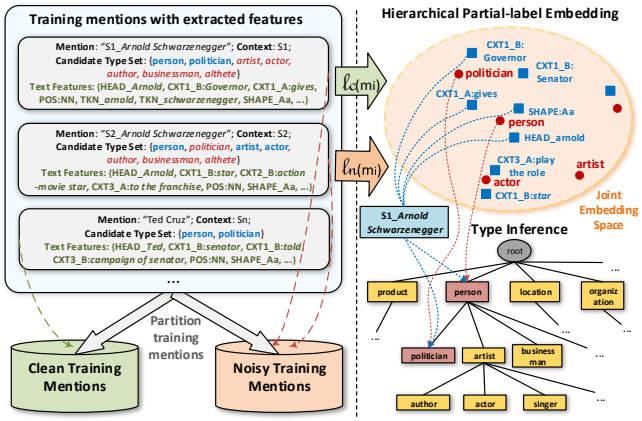


Figure 2: Framework Overview of AFET.

Automatically Labeled Training Corpora. There exist publicly available labeled corpora such as Wikilinks (Singh et al., 2012) and ClueWeb (Gabrilovich et al., 2013). In these corpora, entity mentions are identified and mapped to KB entities using anchor links. In specific domains (*e.g.*, product reviews) where such public corpora are unavailable, one can utilize distant supervision to automatically label the corpus (Ling and Weld, 2012). Specifically, an entity linker will detect mentions m_i and map them to one or more entity e_i in $\mathcal{E}_\Psi$. Types of e_i in KB are then associated with m_i to form its type set $\mathcal{Y}_i$, *i.e.*, $\mathcal{Y}_i = \{y \mid (e_i, y) \in \mathcal{F}_\Psi,\ y \in \mathcal{Y}\}$. Formally, a training corpus $\mathcal{D}$ consists of a set of extracted *entity mentions* $\mathcal{M} = \{m_i\}_{i=1}^N$, the *context* (*e.g.*, sentence, paragraph) of each mention $\{c_i\}_{i=1}^N$, and the candidate *type sets* $\{\mathcal{Y}_i\}_{i=1}^N$ for each mention. We represent $\mathcal{D}$ using a set of triples $\mathcal{D} = \{(m_i, c_i, \mathcal{Y}_i)\}_{i=1}^N$.

Problem Description. For each test mention, we aim to predict the correct type-path in $\mathcal{Y}$ based on the *mention's context*. More specifically, the test set $\mathcal{T}$ is defined as a set of mention-context pairs (m, c), where mentions in $\mathcal{T}$ (denoted as $\mathcal{M}_t$) are extracted from their sentences using existing extractors such as named entity recognizer (Finkel et al., 2005). We denote the gold type-path for a test mention m as $\mathcal{Y}^*$. This work focuses on learning a typing model from the noisy training corpus D, and estimating $\mathcal{Y}^*$ from $\mathcal{Y}$ for each test mention m (in set $\mathcal{M}_t$), based on mention m, its context c, and the learned model.

Framework Overview. At a high level, the AFET framework (see also Fig. 2) learns low-dimensional representations for entity types and text features, and

infers type-paths for test mentions using the learned embeddings. It consists of the following steps:

1. Extract text features for entity mentions in training set $\mathcal{M}$ and test set $\mathcal{M}_t$ using their surface names as well as the contexts. (Sec. 3.1).
2. Partition training mentions $\mathcal{M}$ into a clean set (denoted as $\mathcal{M}_c$) and a noisy set (denoted as $\mathcal{M}_n$) based on their candidate type sets (Sec. 3.2).
3. Perform joint embedding of entity mentions $\mathcal{M}$ and type hierarchy $\mathcal{Y}$ into the same low-dimensional space where, in that space, close objects also share similar types (Secs. 3.3-3.6).
4. For each test mention m, estimate its type-path $\mathcal{Y}^*$ (on the hierarchy $\mathcal{Y}$) in a top-down manner using the learned embeddings (Sec. 3.6).

3 The AFET Framework

This section introduces the proposed framework and formulates an optimization problem for learning embeddings of text features and entity types jointly.

3.1 Text Feature Generation

We start with a representation of entity mentions. To capture the shallow syntax and distributional semantics of a mention $m_i \in \mathcal{M}$, we extract various features from both m_i itself and its context c_i. Table 2 lists the set of text features used in this work, which is similar to those used in (Yogatama et al., 2015; Ling and Weld, 2012). We denote the set of M unique features extracted from $\mathcal{D}$ as $\mathcal{F} = \{f_j\}_{j=1}^{M}$.

3.2 Training Set Partition

A training mention m_i (in set $\mathcal{M}$) is considered as a "clean" mention if its candidate type set obtained by distant supervision (*i.e.*, $\mathcal{Y}_i$) is not ambiguous, *i.e.*, candidate types in $\mathcal{Y}_i$ can form a *single* path in tree $\mathcal{Y}$. Otherwise, a mention is considered as "noisy" mention if its candidate types form *multiple* type-paths in $\mathcal{Y}$. Following the above hypothesis, we judge each mention m_i (in set $\mathcal{M}$) and place it in either the "clean" set $\mathcal{M}_c$, or the "noisy" set $\mathcal{M}_n$. Finally, we have $\mathcal{M} = \mathcal{M}_c \cup \mathcal{M}_n$.

3.3 The Joint Mention-Type Model

We propose to learn mappings into low-dimensional vector space, where, both entity mentions and type

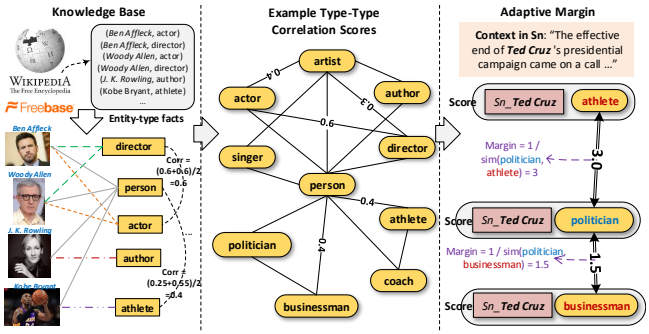


Figure 3: An illustration of KB-based type correlation computation, and the proposed adaptive margin.

labels (in the training set) are represented, and in that space, *two objects are embedded close to each other if and only if they share similar types*. In doing so, we later can derive the representation of a test mention based on its text features and the learned mappings. Mapping functions for entity mentions and entity type labels are different as they have different representations in the *raw* feature space, but are jointly learned by optimizing a global objective of interests to handle the aforementioned challenges.

Each entity mention $m_i \in M$ can be represented by a M-dimensional feature vector $\mathbf{m}_i \in \mathbb{R}^M$, where $m_{i,j}$ is the number of occurrences of feature f_j (in set $\mathcal{F}$) for m_i. Each type label $y_k \in \mathcal{Y}$ is represented by a K-dimensional binary indicator vector $\mathbf{y}_k \in \{0,1\}^K$, where $\mathbf{y}_{k,k} = 1$, and 0 otherwise.

Specifically, we aim to learn a mapping function from the mention's feature space to a low-dimensional vector space, *i.e.*, $\Phi_{\mathcal{M}}(\mathbf{m}_i) : \mathbb{R}^M \mapsto \mathbb{R}^d$ and a mapping function from type label space to the same low-dimensional space, *i.e.*, $\Phi_{\mathcal{Y}}(\mathbf{y}_k) : \mathbb{R}^K \mapsto \mathbb{R}^d$. In this work, we adopt linear maps, as similar to the mapping functions used in (Weston et al., 2011).

$$\Phi_{\mathcal{M}}(\mathbf{m}_i) = \mathbf{U}\mathbf{m}_i; \quad \Phi_{\mathcal{Y}}(\mathbf{y}_k) = \mathbf{V}\mathbf{y}_k, \qquad (1)$$

where $\mathbf{U} \in \mathbb{R}^{d \times M}$ and $\mathbf{V} \in \mathbb{R}^{d \times K}$ are the projection matrices for mentions and type labels, respectively.

3.4 Modeling Type Correlation

In type hierarchy (tree) $\mathcal{Y}$, types closer to each other (*i.e.*, shorter path) tend to be more related (*e.g.*, `actor` is more related to `artist` than to `person` in the right column of Fig. 2). In KB Ψ, types assigned to similar sets of entities should be more related to each other than those assigned to quite different entities (Jiang et al., 2015) (*e.g.*, `actor` is

Feature	Description	Example
Head	Syntactic head token of the mention	"HEAD_Turing"
Token	Tokens in the mention	"Turing", "Machine"
POS	Part-of-Speech tag of tokens in the mention	"NN"
Character	All character trigrams in the head of the mention	":tu", "tur", ..., "ng:"
Word Shape	Word shape of the tokens in the mention	"Aa" for "Turing"
Length	Number of tokens in the mention	"2"
Context	Unigrams/bigrams before and after the mention	"CXT_B:Maserati ,", "CXT_A:and the"
Brown Cluster	Brown cluster ID for the head token (learned using $\mathcal{D}$)	"4_1100", "8_1101111", "12_111011111111"
Dependency	Stanford syntactic dependency (Manning et al., 2014) associated with the head token	"GOV:nn", "GOV:turing"

Table 2: Text features used in this paper. "*Turing Machine*" is used as an example mention from "*The band's former drummer Jerry Fuchs—who was also a member of Maserati,* ***Turing Machine*** *and The Juan MacLean—died after falling down an elevator shaft.*".

more related to `director` than to `author` in the left column of Fig. 3). Thus, type correlation between y_k and $y_{k'}$ (denoted as $w_{kk'}$) can be measured either using the *one over the length of shortest path in* $\mathcal{Y}$, or using the *normalized number of shared entities in KB*, which is defined as follows.

$$w_{kk'} = \Big(|\mathcal{E}_k \cap \mathcal{E}_{k'}|/|\mathcal{E}_k| + |\mathcal{E}_k \cap \mathcal{E}_{k'}|/|\mathcal{E}_{k'}|\Big)/2. \quad (2)$$

Although a shortest path is efficient to compute, its accuracy is limited—It is not always true that a type (*e.g.*, `athlete`) is more related to its parent type (*i.e.*, `person`) than to its sibling types (*e.g.*, `coach`), or that all sibling types are equally related to each other (*e.g.*, `actor` is more related to `director` than to `author`). We later compare these two methods in our experiments.

With the type correlation computed, we propose to apply *adaptive* penalties on different negative type labels (for a training mention), instead of treating all of the labels *equally* as in most existing work (Weston et al., 2011). The hypothesis is intuitive: given the positive type labels for a mention, we force the negative type labels which are related to the positive type labels to receive smaller penalty. For example, in the right column of Fig. 3, negative label `businessman` receives a smaller penalty (*i.e.*, margin) than `athele` does, since `businessman` is more related to `politician`.

Hypothesis 1 (Adaptive Margin) *For a mention, if a negative type is correlated to a positive type, the margin between them should be smaller.*

We propose an adaptive-margin rank loss to model the set of "clean" mentions (*i.e.*, $\mathcal{M}_c$), based on the above hypothesis. The intuition is simple: for each mention, rank all the positive types ahead of negative types, where the ranking score is measured by similarity between mention and type. We denote

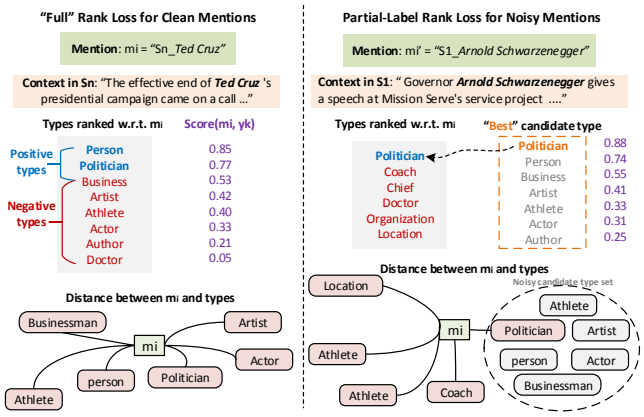


Figure 4: An illustration of the partial-label rank loss.

$f_k(m_i)$ as the similarity between (m_i, y_k) and is defined as the inner product of $\Phi_{\mathcal{M}}(\mathbf{m}_i)$ and $\Phi_{\mathcal{Y}}(\mathbf{y}_k)$.

$$\ell_c(m_i, \mathcal{Y}_i, \overline{\mathcal{Y}}_i) = \sum_{y_k \in \mathcal{Y}_i} \sum_{y_{\bar{k}} \in \overline{\mathcal{Y}}_i} L\Big\lfloor \mathrm{rank}_{y_k}\Big(f(m_i)\Big)\Big\rfloor \Theta_{i,k,\bar{k}};$$

$$\Theta_{i,k,\bar{k}} = \max\Big\{0, \gamma_{k,\bar{k}} - f_k(m_i) + f_{\bar{k}}(m_i)\Big\};$$

$$\mathrm{rank}_{y_k}\Big(f(m_i)\Big) = \sum_{y_{\bar{k}} \in \overline{\mathcal{Y}}_i} \mathbb{1}\Big(\gamma_{k,\bar{k}} + f_{\bar{k}}(m_i) > f_k(m_i)\Big).$$

Here, $\gamma_{k,\bar{k}}$ is the adaptive margin between positive type k and negative type $\bar{k}$, which is defined as $\gamma_{k,\bar{k}} = 1 + 1/(w_{k,\bar{k}} + \alpha)$ with a smooth parameter α. $L(x) = \sum_{i=1}^{x} \frac{1}{i}$ transforms rank to a weight, which is then multiplied to the max-margin loss $\Theta_{i,k,\bar{k}}$ to optimize precision at x (Weston et al., 2011).

3.5 Modeling Noisy Type Labels

True type labels for noisy entity mentions $\mathcal{M}_n$ (i.e., mentions with ambiguous candidate types in the given type hierarchy) in each sentence are not available in knowledge bases. To effectively model the set of noisy mentions, we propose not to treat *all*

candidate types (*i.e.*, $\{\mathcal{Y}_i\}$ as true labels. Instead, we model the "true" label among the candidate set as *latent value*, and try to infer that using text features.

Hypothesis 2 (Partial-Label Loss) *For a noisy mention, the maximum score associated with its candidate types should be greater than the scores associated with any other non-candidate types*

We extend the partial-label loss in (Nguyen and Caruana, 2008) (used to learn linear classifiers) to enforce Hypothesis 2, and integrate with the adaptive margin to define the loss for m_i (in set $\mathcal{M}_n$).

$$\ell_n(m_i, \mathcal{Y}_i, \overline{\mathcal{Y}}_i) = \sum_{\bar{k} \in \overline{\mathcal{Y}}_i} L\Big\lfloor \text{rank}_{y_{k^*}}\Big(f(m_i)\Big)\Big\rfloor \Omega_{i,\bar{k}};$$

$$\Omega_{i,k} = \max\Big\{0, \gamma_{k^*,\bar{k}} - f_{k^*}(m_i) + f_{\bar{k}}(m_i)\Big\};$$

$$\text{rank}_{y_{k^*}}\Big(f(m_i)\Big) = \sum_{y_{\bar{k}} \in \overline{\mathcal{Y}}_i} \mathbb{1}\Big(\gamma_{k^*,\bar{k}} + f_{\bar{k}}(m_i) > f_{k^*}(m_i)\Big)$$

where we define . $y_{k^*} = \text{argmax}_{y_k \in \mathcal{Y}_i} f_k(m_i)$ and $y_{\bar{k}^*} = \text{argmax}_{y_k \in \overline{\mathcal{Y}}_i} f_k(m_i)$.

Minimizing ℓ_n encourages a *large margin* between the maximum scores $\max_{y_k \in \mathcal{Y}_i} f_{y_k}(m_i)$ and $\max_{y_{\bar{k}} \in \overline{\mathcal{Y}}_i} f_{y_k}(m_i)$. This forces m_i to be embedded closer to the most "relevant" type in the noisy candidate type set, *i.e.*, $y^* = \text{argmax}_{y_k \in \mathcal{Y}_i} f_{y_k}(m_i)$, than to *any other* non-candidate types (*i.e.*, Hypothesis 2). This constrasts sharply with multi-label learning (Yosef et al., 2012), where a large margin is enforced between *all* candidate types and non-candidate types without considering noisy types.

3.6 Hierarchical Partial-Label Embedding

Our goal is to embed the heterogeneous graph G into a d-dimensional vector space, following the three proposed hypotheses in the section. Intuitively, one can *collectively* minimize the objectives of the two kinds of loss functions ℓ_c and ℓ_n, across all the training mentions. To achieve the goal, we formulate a joint optimization problem as follows.

$$\min_{\mathbf{U},\,\mathbf{V}} \mathcal{O} = \sum_{m_i \in \mathcal{M}_c} \ell_c(m_i, \mathcal{Y}_i, \overline{\mathcal{Y}}_i) + \sum_{m_i \in \mathcal{M}_n} \ell_n(m_i, \mathcal{Y}_i, \overline{\mathcal{Y}}_i).$$

We use an alternative minimization algorithm based on block-wise coordinate descent (Tseng, 2001) to *jointly* optimize the objective $\mathcal{O}$. One can also apply stochastic gradient descent to do online update.

Type Inference. With the learned mention embeddings $\{\mathbf{u}_i\}$ and type embeddings $\{\mathbf{v}_k\}$, we perform top-down search in the given type hierarchy $\mathcal{Y}$ to estimate the correct type-path $\mathcal{Y}_i^*$. Starting from the tree's root, we recursively find the best type among the children types by measuring the dot product of the corresponding mention and type embeddings, *i.e.*, $\text{sim}(\mathbf{u}_i, \mathbf{v}_k)$. The search process stops when we reach a leaf type, or the similarity score is below a pre-defined threshold $\eta > 0$.

Data sets	Wiki	OntoNotes	BBN
#Types	113	89	47
#Documents	780,549	13,109	2,311
#Sentences	1.51M	143,709	48,899
#Training mentions	2.69M	223,342	109,090
#Ground-truth mentions	563	9,604	121,001
#Features	644,860	215,642	125,637
#Edges in graph	87M	5.9M	2.9M

Table 3: Statistics of the datasets.

4 Experiments

4.1 Data Preparation

Datasets. Our experiments use three public datasets. (1) **Wiki** (Ling and Weld, 2012): consists of 1.5M sentences sampled from Wikipedia articles; (2) **OntoNotes** (Weischedel et al., 2011): consists of 13,109 news documents where 77 test documents are manually annotated (Gillick et al., 2014); (3) **BBN** (Weischedel and Brunstein, 2005): consists of 2,311 Wall Street Journal articles which are manually annotated using 93 types. Statistics of the datasets are shown in Table 3.

Training Data. We followed the process in (Ling and Weld, 2012) to generate training data for the Wiki dataset. For the BBN and OntoNotes datasets, we used DBpedia Spotlight[3] for entity linking. We discarded types which cannot be mapped to Freebase types in the BBN dataset (47 of 93).

Table 2 lists the set of features used in our experiments, which are similar to those used in (Yogatama et al., 2015; Ling and Weld, 2012) except for topics and ReVerb patterns. We discarded the features which occur only once in the corpus.

4.2 Evaluation Settings

For the Wiki and OntoNotes datasets, we used the provided test set. Since BBN corpus is fully annotated, we followed a 80/20 ratio to partition it into

[3] http://spotlight.dbpedia.org/

training/test sets. We report Accuracy (Strict-F1), Micro-averaged F1 (Mi-F1) and Macro-averaged F1 (Ma-F1) scores commonly used in the fine-grained type problem (Ling and Weld, 2012; Yogatama et al., 2015). Since we use the gold mention set for testing, the Accuracy (**Acc**) we reported is the same as the Strict F1.

Baselines. We compared the proposed method (AFET) and its variant with state-of-the-art typing methods, embedding methods and partial-label learning methods [4]: (1) **FIGER** (Ling and Weld, 2012); (2) **HYENA** (Yosef et al., 2012); (3) **FIGER/HYENA-Min** (Gillick et al., 2014): removes types appearing only once in the document; (4) **ClusType** (Ren et al., 2015): predicts types based on co-occurring relation phrases; (5) **HNM** (Dong et al., 2015): proposes a hybrid neural model without hand-crafted features; (6) **DeepWalk** (Perozzi et al., 2014): applies Deep Walk to a feature-mention-type graph by treating all nodes as the same type; (7) **LINE** (Tang et al., 2015b): uses a second-order LINE model on feature-type bipartite graph; (8) **PTE** (Tang et al., 2015a): applies the PTE joint training algorithm on feature-mention and type-mention bipartite graphs. (9) **WSABIE** (Yogatama et al., 2015): adopts WARP loss to learn embeddings of features and types; (10) **PL-SVM** (Nguyen and Caruana, 2008): uses a margin-based loss to handle label noise. (11) **CLPL** (Cour et al., 2011): uses a linear model to encourage large average scores for candidate types.

We compare AFET and its variant: (1) **AFET**: complete model with KB-induced type correlation; (2) **AFET-CoH**: with hierarchy-induced correlation (*i.e.*, shortest path distance); (3) **AFET-NoCo**: without type correlation (*i.e.*, all margin are "1") in the objective $\mathcal{O}$; and (4) **AFET-NoPa**: without label partial loss in the objective $\mathcal{O}$.

4.3 Performance Comparison and Analyses

Table 4 shows the results of AFET and its variants.

Comparison with the other typing methods. AFET outperforms both FIGER and HYENA systems, demonstrating the predictive power of the learned embeddings, and the effectiveness of modeling type correlation information and noisy candidate types. We also observe that pruning methods do not always improve the performance, since they aggressively filter out rare types in the corpus, which may lead to low Recall. ClusType is not as good as FIGER and HYENA because it is intended for coarse types and only utilizes relation phrases.

Comparison with the other embedding methods. AFET performs better than all other embedding methods. HNM does not use any linguistic features. None of the other embedding methods consider the label noise issue and treat the candidate type sets as clean. Although AFET adopts the WARP loss in WSABIE, it uses an adaptive margin in the objective to capture the type correlation information.

Comparison with partial-label learning methods. Compared with PL-SVM and CLPL, AFET obtains superior performance. PL-SVM assumes that only *one* candidate type is correct and does not consider type correlation. CLPL simply averages the model output for all candidate types, and thus may generate results biased to frequent false types. Superior performance of AFET mainly comes from modeling type correlation derived from KB.

Comparison with its variants. AFET always outperforms its variant on all three datasets. It gains performance from capturing type correlation, as well as handling type noise in the embedding process.

4.4 Case Analyses

Example output on news articles. Table 5 shows the types predicted by AFET, FIGER, PTE and WSABIE on two news sentences from OntoNotes dataset: AFET predicts fine-grained types with better accuracy (*e.g.*, `person_title`) and avoids overly-specific predictions (*e.g.*, `news_company`). Figure 5 shows the types estimated by AFET, PTE and WSABIE on a training sentence from OntoNotes dataset. We found AFET could discover the best type from noisy candidate types.

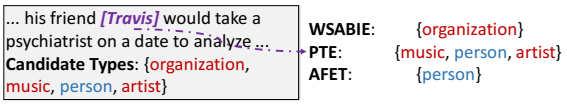


Figure 5: Example output of AFET and the compared methods on a training sentence from **OntoNotes** dataset.

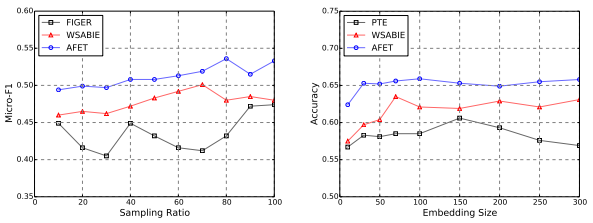
[4] We used the published code for FIGER, ClusType, HNM, LINE, PTE, and DeepWalk, and implemented other baselines which have no public code. Our implementations yield comparable performance as those reported in the original papers.

Typing Method	Wiki			OntoNotes			BBN		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
CLPL (Cour et al., 2011)	0.162	0.431	0.411	0.201	0.347	0.358	0.438	0.603	0.536
PL-SVM (Nguyen and Caruana, 2008)	0.428	0.613	0.571	0.225	0.455	0.437	0.465	0.648	0.582
FIGER (Ling and Weld, 2012)	0.474	0.692	0.655	0.369	0.578	0.516	0.467	0.672	0.612
FIGER-Min (Gillick et al., 2014)	0.453	0.691	0.631	0.373	0.570	0.509	0.444	0.671	0.613
HYENA (Yosef et al., 2012)	0.288	0.528	0.506	0.249	0.497	0.446	0.523	0.576	0.587
HYENA-Min	0.325	0.566	0.536	0.295	0.523	0.470	0.524	0.582	0.595
ClusType (Ren et al., 2015)	0.274	0.429	0.448	0.305	0.468	0.404	0.441	0.498	0.573
HNM (Dong et al., 2015)	0.237	0.409	0.417	0.122	0.288	0.272	0.551	0.591	0.606
DeepWalk (Perozzi et al., 2014)	0.414	0.563	0.511	0.479	0.669	0.611	0.586	0.638	0.628
LINE (Tang et al., 2015b)	0.181	0.480	0.499	0.436	0.634	0.578	0.576	0.687	0.690
PTE (Tang et al., 2015a)	0.405	0.575	0.526	0.436	0.630	0.572	0.604	0.684	0.695
WSABIE (Yogatama et al., 2015)	0.480	0.679	0.657	0.404	0.580	0.527	0.619	0.670	0.680
AFET-NoCo	0.526	0.693	0.654	0.486	0.652	0.594	0.655	0.711	0.716
AFET-NoPa	0.513	0.675	0.642	0.463	0.637	0.591	0.669	0.715	0.724
AFET-CoH	0.433	0.583	0.551	0.521	0.680	0.609	0.657	0.703	0.712
AFET	**0.533**	**0.693**	**0.664**	**0.551**	**0.711**	**0.647**	**0.670**	**0.727**	**0.735**

Table 4: Study of typing performance on the three datasets.

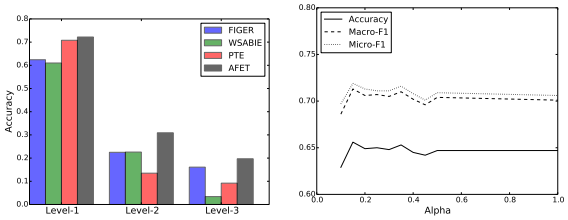
Text	"... going to be an imminent easing of monetary policy, " said Robert Dederick , chief economist at ***Northern Trust Co. in Chicago***.	...It's terrific for advertisers to know the reader will be paying more , " said Michael Drexler , ***national media director*** at Bozell Inc. ad agency.
Ground Truth	`organization`, `company`	`person`, `person_title`
FIGER	`organization`	`organization`
WSABIE	`organization`, `company`, `broadcast`	`organization`, `company`, `news_company`
PTE	`organization`	`person`
AFET	`organization`, `company`	`person`, `person_title`

Table 5: Example output of AFET and the compared methods on two news sentences from **OntoNotes** dataset.



(a) Varying training set size (b) Varying d

Figure 6: Performance change with respect to (a) sampling ratio of training mentions on the Wiki dataset; and (b) embedding dimension d on the BBN dataset.



(a) Test at different levels (b) Varying α

Figure 7: Performance change (a) at different levels of the type hierarchy on the OntoNotes dataset; and (b) with respect to smooth parameter α on the BBN dataset.

Testing the effect of training set size and dimension. Experimenting with the same settings for model learning, Fig. 6(a) shows the performance trend on the Wiki dataset when varying the sampling ratio (subset of mentions randomly sampled from the training set $\mathcal{D}$). Fig. 6(b) analyzes the performance sensitivity of AFET with respect to d—the embedding dimension on the BBN dataset. Accuracy of AFET improves as d becomes large but the gain decreases when d is large enough.

Testing sensitivity of the tuning parameter. Fig. 7(b) analyzes the sensitivity of AFET with respect to α on the BBN dataset. Performance increases as α becomes large. When α is large than 0.5, the performance becomes stable.

Testing at different type levels. Fig. 7(a) reports the Ma-F1 of AFET, FIGER, PTE and WSABIE at different levels of the target type hierarchy (*e.g.*, person and location on level-1, politician and artist on level-2, author and actor on level-3). The results show that it is more difficult to distinguish among more fine-grained types. AFET always outperforms the other two method, and achieves a 22.36% improvement in Ma-F1, compared to FIGER on level-3 types. The gain mainly comes from explicitly modeling the noisy candidate types.

Testing for frequent/infrequent types. We also

Type	animal	city
# of Training Mentions	1882	12421
# of Test Mentions	8	240
WSABIE	0.176	0.546
FIGER	0.167	0.648
PTE	0.222	0.677
AFET	**0.400**	**0.766**

Table 6: Example output of AFET and other methods on frequent/infrequent type from **OntoNotes** dataset.

evaluate the performance on frequent and rare types (Table 6). Note that we use a different evaluation metric, which is introduced in (Yosef et al., 2012) to calculate the F1 score for a type. We find **AFET** can always perform better than other baselines and it works for both frequent and rare types.

5 Related Work

There has been considerable work on named entity recognition (NER) (Manning et al., 2014), which focuses on three types (*e.g.*, `person`, `location`) and cast the problem as multi-class classification following the type mutual exclusion assumption (*i.e.*, one type per mention) (Nadeau and Sekine, 2007).

Recent work has focused on a much larger set of fine-grained types (Yosef et al., 2012; Ling and Weld, 2012). As the type mutual exclusion assumption no longer holds, they cast the problem as multi-label multi-class (hierarchical) classification problems (Gillick et al., 2014; Yosef et al., 2012; Ling and Weld, 2012). Embedding techniques are also recently applied to jointly learn feature and type representations (Yogatama et al., 2015; Dong et al., 2015). Del Corro *et al.* (2015) proposed an unsupervised method to generate context-aware candidates types, and subsequently select the most appropriate type. Gillick *et al.* (2014) discuss the label noise issue in fine-grained typing and propose three pruning heuristics. However, these heuristics aggressively delete training examples and may suffer from low recall (see Table. 4).

In the context of distant supervision, label noise issue has been studied for other information extraction tasks such as relation extraction (Takamatsu et al., 2012). In relation extraction, label noise is introduced by the false positive textual matches of entity pairs. In entity typing, however, label noise comes from the assignment of types to entity mentions without considering their contexts. The forms of distant supervision are different in these two problems. Recently, (Ren et al., 2016b) has tackled the problem of label noise in fine-grained entity typing, but focused on how to generate a clean training set instead of doing entity typing.

Partial label learning (PLL) (Zhang, 2014; Nguyen and Caruana, 2008; Cour et al., 2011) deals with the problem where each training example is associated with a set of candidate labels, where *only one is correct*. Unlike existing PLL methods, our method considers type hierarchy and correlation.

6 Conclusion and Future Work

In this paper, we study *automatic fine-grained entity typing* and propose a *hierarchical partial-label embedding* method, AFET, that models "clean" and "noisy" mentions separately and incorporates a given type hierarchy to induce loss functions. AFET builds on a joint optimization framework, learns embeddings for mentions and type-paths, and iteratively refines the model. Experiments on three public datasets show that AFET is effective, robust, and outperforms other comparing methods.

As future work, it would be interesting to study topical features as the context cues of the entity mentions, to leverage multi-sensing embedding to represent linguistic features with multiple senses, and to exploits other effective modeling methods to inject type hierarchy information. The proposed objective function is general and can be considered to incorporate various language features, to conduct integrated modeling of multiple sources, and to be extended to distantly-supervised relation extraction.

7 Acknowledgments

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA DEFT No. FA8750-13-2-0041, National Science Foundation IIS-1017362, IIS-1320617, IIS-1354329, and IIS-1523198, HDTRA1-10-1-0120, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (`www.bd2k.nih.gov`). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Timothee Cour, Ben Sapp, and Ben Taskar. 2011. Learning from partial labels. *JMLR*, 12:1501–1536.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *EMNLP*.

Xin Luna Dong, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *IJCAI*.

Jesse Dunietz and Dan Gillick. 2014. A new entity salience task with millions of training examples. *EACL*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Jyun-Yu Jiang, Chin-Yew Lin, and Pu-Jen Cheng. 2015. Entity-driven type hierarchy construction for freebase. In *WWW*.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *EMNLP*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. *ACL*.

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

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30:3–26.

Nam Nguyen and Rich Caruana. 2008. Classification with partial labels. In *KDD*.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *ACL*.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, Heng Ji, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *KDD*.

Xiang Ren, Ahmed El-Kishky, Chi Wang, and Jiawei Han. 2016a. Automatic entity recognition and typing in massive text corpora. In *WWW*.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *KDD*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia. *UM-CS-2012-015*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *ACL*.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015b. Line: Large-scale information network embedding. In *WWW*.

Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *JOTA*, 109(3):475–494.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium*, 112.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *ACL*.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *COLING*.

Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*.

Min-Ling Zhang. 2014. Disambiguation-free partial label learning. In *SDM*.

Mining Inference Formulas by Goal-Directed Random Walks

Zhuoyu Wei[1,2], **Jun Zhao**[1,2] and **Kang Liu**[1]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{zhuoyu.wei, jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Deep inference on a large-scale knowledge base (KB) needs a mass of formulas, but it is almost impossible to create all formulas manually. Data-driven methods have been proposed to mine formulas from KBs automatically, where random sampling and approximate calculation are common techniques to handle big data. Among a series of methods, Random Walk is believed to be suitable for knowledge graph data. However, a pure random walk without goals still has a poor efficiency of mining useful formulas, and even introduces lots of noise which may mislead inference. Although several heuristic rules have been proposed to direct random walks, they do not work well due to the diversity of formulas. To this end, we propose a novel goal-directed inference formula mining algorithm, which directs random walks by the specific inference target at each step. The algorithm is more inclined to visit benefic structures to infer the target, so it can increase efficiency of random walks and avoid noise simultaneously. The experiments on both WordNet and Freebase prove that our approach is has a high efficiency and performs best on the task.

1 Introduction

Recently, various knowledge bases (KBs), such as Freebase (Bollacker et al., 2008), WordNet (Miller, 1995), Yago (Hoffart et al., 2013), have been built, and researchers begin to explore how to make use of structural information to promote performances of several inference-based NLP applications, such as text entailment, knowledge base completion, question and answering. Creating useful formulas is one of the most important steps in inference, and an accurate and high coverage formula set will bring a great promotion for an inference system. For example, *Nationality(x, y)* $\wedge$ *Nationality(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)* is a high-quality formula, which means people with the same nationality probably speak the same language. However, it is a challenge to create formulas for open-domain KBs, where there are a great variety of relation types and it is impossible to construct a complete formula set by hand.

Several data-driven methods, such as Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994) and Markov Logic Network (MLN) (Richardson and Domingos, 2006), have been proposed to mine formulas automatically from KB data, which transform frequent sub-structures of KBs, e.g., paths or loops, into formulas. Figure 1.a shows a sub-graph extracted from Freebase, and the formula mentioned above about *Language* can be generated from the loop in Figure 1.d. However, the running time of these traditional probabilistic inference methods is unbearable over large-scale KBs. For example, MLN needs grounding for each candidate formula, i.e., it needs to enumerate all paths. Therefore, the computation complexity of MLN increases exponentially with the scale of a KB.

In order to handle large-scale KBs, the random walk is usually employed to replace enumerating all possible sub-structures. However, random walk is inefficient to find useful structures due to its completely randomized mechanism. For example in Fig-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1379–1388,
Austin, Texas, November 1-5, 2016.

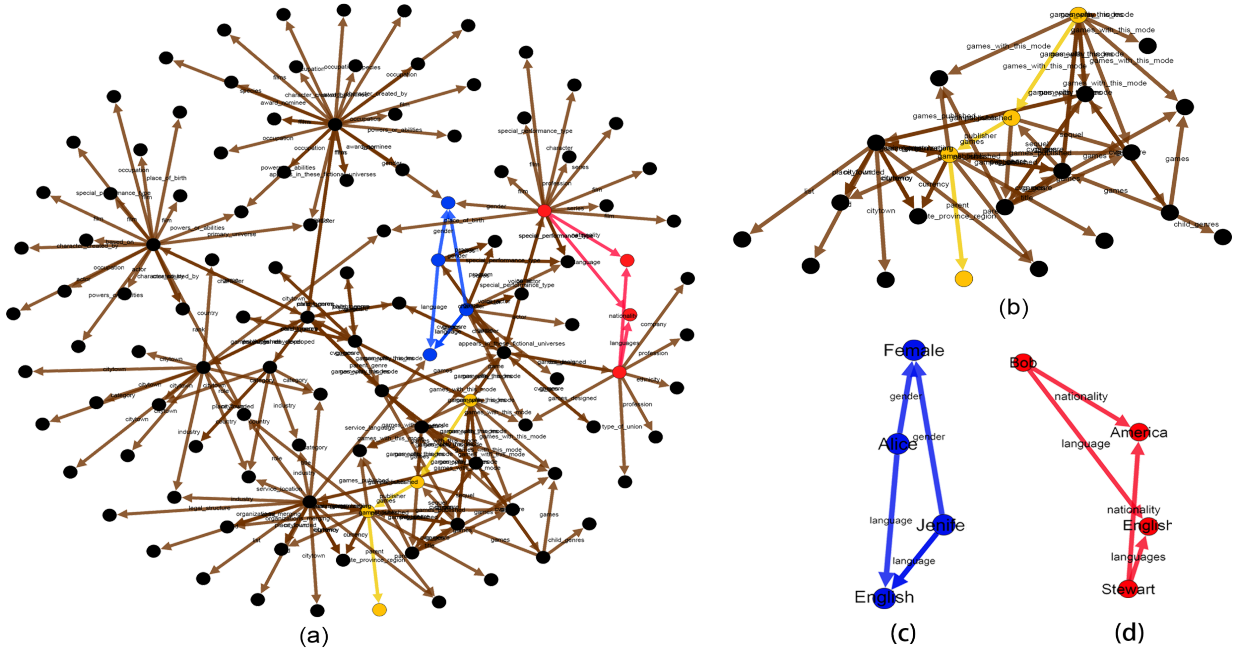


Figure 1: a) shows a subgraph extracted from Freebase. b) shows the searching space of finding the yellow path. c) shows a loop which can generate a false formula. d) shows a loop which can generate a true formula.

ure 1.b, the target path (yellow one) has a small probability to be visited, the reason is that the algorithm may select all the neighboring entity to transfer with an equal probability. This phenomenon is very common in KBs, e.g., each entity in Freebase has more than 30 neighbors in average, so there will be about 810,000 paths with length 4, and only several are useful. There have been two types of methods proposed to improve the efficiency of random walks, but they still meet serious problems, respectively.

1) **Increasing rounds of random walks.** More rounds of random walks will find more structures, but it will simultaneously introduce more noise and thus generate more false formulas. For example, the loop in Figure 1.c exists in Freebase, but it produces a false formula, *Gender(x, y)* $\wedge$ *Gender(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)*, which means people with the same gender speak the same language. This kind of structures frequently occur in KBs even the formulas are mined with a high confidence, because there are a lot of sparse structures in KBs which will lead to inaccurate confidence. According to our statistics, more than 90 percent of high-confidence formulas produced by random walk are noise.

2) **Employing heuristic rules to direct random walks.** This method directs random walks to find useful structures by rewriting the state transition probability matrix, but the artificial heuristic rules may only apply to a little part of formulas. For example, PRA (Lao and Cohen, 2010; Lao et al., 2011) assumes the more narrow distributions of elements in a formula are, the higher score the formula will obtain. However, formulas with high scores in PRA are not always true. For example, the formula in Figure 1.c has a high score in PRA, but it is not true. Oppositely, formulas with low scores in PRA are not always useless. For example, the formula, $Father(x, y) \wedge Father(y, z) \Rightarrow Grandfather(x, t)$, has a low score when x and y both have several sons, but it obviously is the most effective to infer $Grandfather$. According to our investigations, the situations are common in KBs.

In this paper, we propose a Goal-directed Random Walk algorithm to resolve the above problems. The algorithm employs the specific inference target as the direction at each step in the random walk process. In more detail, to achieve such a goal-directed mechanism, at each step of random walk, the algorithm dynamically estimates the potentials for each neighbor by using the ultimate goal, and assigns higher probabilities to the neighbors with higher potentials. Therefore, the algorithm is more inclined to visit structures which are beneficial to infer

the target and avoid transferring to noise structures. For example in Figure 1, when the inference target is *what language a person speaks*, the algorithm is more inclined to walk along *Nationality* edge than *Gender*, because *Nationality* has greater potential than *Gender* to infer *Language*. We build a real potential function based on low-rank distributional representations. The reason of replacing symbols by distributional representations is that the distributional representations have less parameters and latent semantic relationship in them can contribute to estimate potentials more precisely. In summary, the contributions of this paper are as follows.

• Compared with the basic random walk, our approach direct random walks by the inference target, which increases efficiency of mining useful formulas and has a great capability of resisting noise.

• Compared with the heuristic methods, our approach can learn the strategy of random walk automatically and dynamically adjust the strategy for different inference targets, while the heuristic methods need to write heuristic rules by hand and follow the same rule all the time.

• The experiments on link prediction task prove that our approach has a high efficiency on mining formulas and has a good performance on both WN18 and FB15K datasets.

The rest of this paper is structured as follows, Section 2 introduces the basic random walk for mining formulas. Section 3 describes our approach in detail. The experimental results and related discussions are shown in Section 4. Section 5 introduces some related works, and finally, Section 6 concludes this paper.

2 Mining Formulas by Random Walk

2.1 Frequent Pattern Mining

Mining frequent patterns from source data is a problem that has a long history, and for different specific tasks, there are different types of source data and different definitions of pattern. Mining formulas is more like frequent subgraph mining, which employs paths or loops as frequent patterns and mines them from a KB. For each relation type R, the algorithm enumerates paths from entity H to entity T for each triplet $R(H,T)$. These paths are normalized to formulas by replacing entities to variables. For example, the loop in Figure 1.d, *Nationality(Bob, America)* $\wedge$ *Nationality(Stewart, America)* $\wedge$ *Language(Bob, English)* $\Rightarrow$ *Language(Stewart, English)*, can be normalized to the formula, *Nationality(x, y)* $\wedge$ *Nationality(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)*. Support and confidence are employed to estimate a formula, where the support value of a formula $f : X \Rightarrow Y$, noted as S_f, is defined as the proportion of paths in the KB which contains the body X, and the confidence value of $X \Rightarrow Y$, noted as C_f, is defined as the proportion of the paths that contains X which also meets $X \Rightarrow Y$. C_f is calculated as follows,

$$C_f = \frac{N_f}{N_X} \tag{1}$$

where N_f is the total number of instantiated formula f and N_X is the total number of instantiated X.

2.2 Random Walk on Knowledge Graph

Enumerating paths is a time consuming process and does not apply to large-scale KBs. Therefore, random walk on the graph is proposed to collect frequent paths instead of enumerating. Random walk randomly chooses a neighbor to jump unlike enumerating which needs to search all neighbors. To estimate a formula f, the algorithm employs f's occurrence number during random walks $N_f^{'}$ to approximate the total number N_f in Equation (1), and similarly employs $N_X^{'}$ to approximate N_X. Therefore, f's confidence C_f can be approximatively estimated by $N_f^{'}$ and $N_X^{'}$, noted as $C_f^{'}$.

Random walk maintains a state transition probability matrix P, and P_{ij} means the probability of jumping from entity i to entity j. To make the confidence $C_f^{'}$ as close to the true confidence C_f as possible, the algorithm sets P as follows,

$$P_{ij} = \begin{cases} 1/d_i, & j \in Adj(i) \\ 0, & j \notin Adj(i) \end{cases} \tag{2}$$

where d_i is the out-degree of the entity i, $Adj(i)$ is the set of adjacent entities of i, and $\sum_{j=1}^{N} P_{ij} = 1$. Such a transition matrix means the algorithm may jump to all the neighboring entities with an equal probability. Such a random walk is independent from the inference target, so we call this type of random walk as a goalless random walk. The goalless mechanism causes the inefficiency of mining useful structures. When we want to mine paths for $R(H,T)$, the algorithm cannot arrive at T from H

in the majority of rounds. Even though the algorithm recalls several paths for $R(H,T)$, some of them may generate noisy formulas for inferring $R(H,T)$.

To solve the above problem, several methods direct random walks by statically modifying P. For example, PRA sets $P_{r_{ij}} = \frac{P(j|i;r)}{|R_i|}$, $P(j|i;r) = \frac{r(i,j)}{r(i,*)}$, where $P(j|i;r)$ is the probability of reaching node j from node i under the specific relation r, $r(i,*)$ is the number of edges from i under r, and R_i is the number of relation types from i. Such a transition matrix implies the more narrow distributions of elements in a formula are, the higher score the formula will obtain, which can be viewed as the heuristic rule of PRA.

3 Our Approach

3.1 Goal-Directed Random Walk

We propose to use the inference target, $\rho = R(H,T)$, to direct random walks. When predicting ρ, our approach always directs random walks to find useful structures which may generate formulas to infer ρ. For different ρ, random walks are directed by modifying the transition matrix P in different ways. Our approach dynamically calculates $P_{r_{ij}}$ when jumping from entity i to entity j under relation r as follows,

$$P_{r_{ij}} = \begin{cases} \dfrac{\Phi(r(i,j),\rho)}{\sum_{k\in Adj(i)}\Phi(r(i,k),\rho)}, & j \in Adj(i) \\ 0, & j \notin Adj(i) \end{cases} \tag{3}$$

where $\Phi(r(i,j),\rho)$ is the $r(i,j)$'s potential which measures the potential contribution to infer ρ after walking to j.

Intuitively, if $r(i,j)$ exits in a path from H to T and this path can generate a benefic formula to infer $R(H,T)$, the probability of jumping from i to j should larger and thus $\Phi(r(i,j),\rho)$ also should be larger. Reversely, if we cannot arrive at T within the maximal steps after jumping to j, or if the path produces a noisy formula leading to a wrong inference, P_{ij} and $\Phi(r(i,j),\rho)$ should both be smaller.

To explicitly build a bridge between the potential Φ and the inference goal ρ, we maximize the likelihood of paths which can infer ρ. First, we recursively define the likelihood of a path from H to t as $P_{p_{Ht}} = P_{p_{Hs}} \cdot P_{r_{st}}$, where $P_{r_{st}}$ is defined in Equation (3). We then classify a path p_{Ht} into three separate categories: a) $t = T$ and p_{Ht} can produce a benefic formula to infer $R(H,T)$; b) $t \neq T$; c) $t = T$ but p_{Ht} may generate a noisy formula which misleads inference. Finally, we define the likelihood function as follows,

$$\max P_{\mathbb{P}} = \prod_{p_{Ht}\in\mathbb{P}} P_{p_{Ht}}^{a}(1-P_{p_{Ht}})^{b+c} \tag{4}$$

where $\mathbb{P}$ is all paths found in the process of performing random walks for $R(H,T)$, and t may be equal to T or not. a, b, c are three 0-1 variables corresponding to the above categories a), b), c). Only one in a, b, c can be 1 when P_{Ht} belongs to the corresponding category. We then transform maximizing $P_{\mathbb{P}}$ to minimizing $L_{rw} = -\log P_{\mathbb{P}}$ and employ SGD to train it. In practice, there is not a clear-cut boundary between a) and c), so we divide the loss into two parts: $L_{rw} = L_{rw}^{t} + \lambda L_{rw}^{inf}$. L_{rw}^{t} is the loss of that $t \neq T$, and L_{rw}^{inf} is the loss of that p_{HT} generates a noisy formula leading to a wrong inference. λ is a super parameter to balance the two losses. L_{rw}^{t} and L_{rw}^{inf} have the same expression but are optimized in different stages. L_{rw}^{t} can be optimized during random walks, while L_{rw}^{inf} should be optimized in the inference stage. We rewrite L_{rw} for a specific path p as follows,

$$L_{rw}(p) = -y\log P_p - (1-y)\log(1-P_p) \tag{5}$$

where y is the label of the path p and $y = 1$ if p is beneficial to infer ρ. To obtain the best Φ, we compute gradients of L_{rw} as follows,

$$\begin{aligned} &\nabla L_{rw}(p) = (\nabla L_{rw}(r_{12}), \nabla L_{rw}(r_{23}), ...) \\ &\nabla L_{rw}(r_{ij}) = (\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ij}}}, \frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik_1}}}, \frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik_2}}}, ...) \\ &\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ij}}} = \frac{(P_p - y)\cdot(1-P_{r_{ij}})}{\Phi_{r_{ij}}\cdot(1-P_p)} \\ &\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik}}} = -\frac{(P_p - y)\cdot P_{r_{ij}}}{\Phi_{r_{ij}}\cdot(1-P_p)} \end{aligned} \tag{6}$$

where $\nabla L_{rw}(r_{ij})$ is the component of $\nabla L_{rw}(p)$ at r_{ij}. $\Phi(r(i,j),\rho)$ and $\Phi(r(i,k),\rho)$ are the potentials for all triplets $r(i,j) \in p$ and $r(i,k) \notin p$, and r_{ij} is short for $r(i,j)$. After iteratively updating $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ by the gradient of L_{rw}^{t}, the random walks can

be directed to find more paths from H to T, and consequently it increases efficiency of the random walk. After updating $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ by the gradient of L_{rw}^{inf}, random walk is more likely to find high-quality paths but not noise. Therefore, the goal-directed random walk increases efficiency of mining benefic formulas and has a great capability of resisting noise.

3.2 Distributional Potential Function

The potential $\Phi(r(i,j),\rho)$ measures an implicit relationship between two triplets in the KB, so the total number of parameters is the square of the KB size. It is hard to precisely estimate all Φ because of the sparsity of training data. To reduce the number of parameters, we represent each entity or relation in the KB as a low-rank numeric vector which is called embeddings (Bordes et al., 2013), and then we build a potential function Ψ on embeddings as $\Phi(r(i,j),\rho) = \Psi(E_{r(i,j)}, E_{R(H,T)})$, where $E_{r(i,j)}$ and $E_{R(H,T)}$ are the embeddings of triplets. In practice, we set $E_{r(i,j)} = [E_r, E_j]$ and $E_{R(H,T)} = [E_R, E_T]$ because E_i is the same for all triplets $r(i,*)$, where [] is a concatenation operator.

In the view of the neural network, our goal-directed mechanism is analogous to the attention mechanism. At each step, the algorithm estimates attentions for each neighboring edges by Ψ. Therefore, there are several existing expressions of Ψ, e.g., the dot product (Sukhbaatar et al., 2015) and the single-layer perceptron (Bahdanau et al., 2015). We will not compare different forms of Ψ, the detail comparison has been presented in the work (Luong et al., 2015). We directly employ the simplest dot product for Ψ as follows,

$$\Psi(E_{r(i,j)}, E_{R(H,T)}) = \sigma(E_{r(i,j)} \cdot E_{R(H,T)}) \quad (7)$$

where σ is a nonlinear function and we set it as an exponential function. Ψ has no parameters beside KB embeddings which are updated during the training period.

3.3 Integrated Inference Model

To handle the dependence between goal-directed random walk and subsequent inference, we combine them into an integrated model and optimize them together. To predict $\rho = R(H,T)$, the integrated model first collects formulas for $R(H,T)$, and then merges estimations of different formulas as features into a score function χ,

$$\chi(\rho) = \sum_{f \in F_\rho} \delta(f) \quad (8)$$

where F_ρ is the formula set obtained by random walks for ρ, and $\delta(f)$ is an estimation of formula f. The original frequent pattern mining algorithm employs formulas' confidence as $\delta(f)$ directly, but it does not produce good results (Galárraga et al., 2013). There are two ways to solve the problem: one is selecting another more suitable measure of f as $\delta(f)$ (Tan et al., 2002); the other is attaching a weight to each formula and learning weights with supervision, e.g., MLN (Richardson and Domingos, 2006) . We employ the latter method and set $\delta(f) = w_f \cdot n_f$. Finally, we employ a logistic regression classifier to predict $R(H,T)$, and the posterior probability of $R(H,T)$ is shown as follows,

$$\begin{aligned} P(\rho = y|\chi) &= \mathcal{F}(\chi)^y (1 - \mathcal{F}(\chi))^{1-y} \\ \mathcal{F}(\chi) &= \frac{1}{1+e^{-\chi}} \end{aligned} \quad (9)$$

where y is a 0-1 label of ρ. Similar to L_{rw}^t in Equation (5), we treat the negative logarithm of $P(\rho = y|\chi)$ as the loss of inference, $L_{inf} = -\log P(\rho = y|\chi)$, and turn to minimize it. Moreover, the loss L_{rw}^{inf} of the above goal-directed random walk is influenced by the result of predicting $R(H,T)$, so $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ will be also updated. Algorithm 1 shows the main process of training, where Ξ is the triplet set for training, Ψ is the potential function in Equation (7), F is the formula set, f_p is

Algorithm 1: Train Integrated Inference Model

Input: KB, Ξ
Output: Ψ, W, F
For $\rho = R(H,T) \in \Xi$
  Repeat ρ-directed Random Walk from H to t
    Update Ψ by L_{rw}^t
    If $t = T$, then $F = F \cap f_p$
  Calculate L_{inf} and L_{rw}^{inf} by ρ
  Update W by L_{inf}
  Update Ψ by L_{rw}^{inf}
Remove $f \in F$ with little w_f
Output Ψ, W, F

Dataset	Relation	Entity	Train	Valid	Test
WN18	18	40,943	141,442	5,000	5,000
FB15K	1,345	14,951	483,142	50,000	59,071

Table 1: Statistics of WN18 and FB15K

a formula generated from the path p, and H, T, t are entities in the KB. To predict $\rho = R(H, T)$, the algorithm first performs multi rounds of random walks, and each random walk can find a path p_{Ht} (at line 2). Then the algorithm decides to update Ψ by L^t_{rw} based on whether t is T (at line 3), and adds the formula p_f into the formula set when $t = T$ (at line 4). After random walks, the inference model predicts ρ, and computes L_{inf} and L^{inf}_{rw} according to the prediction result (at line 5). Finally W and Ψ are updated by L_{inf} and L^{inf}_{rw} (at line 6-7), respectively. After training by all triplets in Ξ, the algorithm removes formulas with low weights from F (at line 8) and outputs the model (at line 9). When we infer a new triplet by this model, the process is similar to Algorithm 1.

4 Experiments

We first compare our approach with several state-of-art methods on link prediction task to explore our approach's overall ability of inference. Subsequently, we evaluate formulas mined by different random walk methods to explore whether the goal-directed mechanism can increase efficiency of mining useful structures. Finally, we dive deep into the formulas generated by our approach to analyze the characters of our approach.

4.1 Datasets and Evaluation Setup

We conduct experiments on both WN18 and FB15K datasets which are subsets sampled from WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), respectively, and Table 1 shows the statistics of them. For the link prediction task, we predict the missing h or t for a triplet $r(h, t)$ in test set. The detail evaluation method is that t in $r(h, t)$ is replaced by all entities in the KB and methods need to rank the right answer at the top of the list, and so does h in $r(h, t)$. We report the mean of those true answer ranks and the Hits@10 under both 'raw' and 'filter' as TransE (Bordes et al., 2013) does, where Hits@10 is the proportion of correct entities ranked in the top 10.

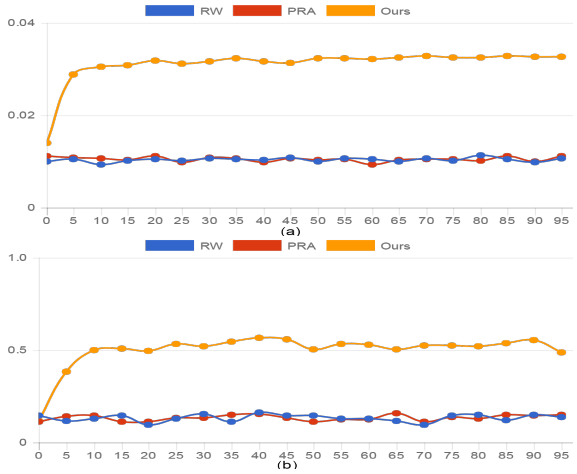


Figure 2: $Arr@10$ of three random walk algorithms and the horizontal axis represents epochs and the vertical axis represents $Arr@10$. Figure 2.a shows results on relation *_derivationally_related_form* in WN18, and Figure 2.b shows results on relation *form_of_government* in FB15K.

4.2 Baselines

We employ two types of baselines. One type is based on random walks including: a) the basic random walk algorithm whose state transition probability matrix is shown in Equation (2); b) PRA in (Lao et al., 2011) which is a typical heuristic random walk algorithm. The other type is based on KB embeddings including TransE (Bordes et al., 2013), Rescal (Nickel et al., 2011), TransH (Wang et al., 2014b), TransR (Lin et al., 2015b). These embedding-based methods have no explicit formulas, so we will not evaluate their performances on mining formulas.

4.3 Settings

We implement three random walk methods under a unified framework. To predict $r(h, *)$ quickly, we first select Top-K candidate instances, $t_{1 \to K}$, by TransE as (Wei et al., 2015), and then the algorithm infers each $r(h, t_i)$ and ranks them by inference results. We adjust parameters for our approach with the validate dataset, and the optimal configurations are set as follows. The rounds of random walk is 10, learning rate is 0.0001, training epoch is 100, the size of candidate set is 500 for WN18 and 100 for FB15K, the embeddings have 50 dimensionalities for WN18 and 100 dimensionalities for FB15K, and the embeddings are initialized by TransE. For some relations, random walk truly finds no practicable formulas, so we employ TransE to improve per-

	Dataset	WN18				FB15K			
	Metric	Mean Rank		Hits@10(%)		Mean Rank		Hits@10(%)	
		Raw	Filt	Raw	Filt	Raw	Filt	Raw	Filt
2.a	RESCAL	1,180	1,163	37.2	52.8	828	683	28.4	44.1
	TransE	263	251	75.4	89.2	243	125	34.9	47.1
	TransH	401	388	73.0	82.3	212	87	45.7	64.4
	TransR	238	225	79.8	92.0	198	77	48.2	68.7
2.b	RW	28*	17*	84.40	94.89	37*	28*	37.04	51.13
	PRA	28*	17*	**84.43**	**94.90**	37*	29*	36.72	50.73
2.d	Our approach	**28***	**17***	84.40	94.86	**34***	**25***	**53.47**	**74.75**

Table 2: Link Prediction Results on both WN18 and FB15K

formance for these relations. For embedding-based methods, we use reported results directly since the evaluation datasets are identical.

4.4 Results on Link Prediction

We show the results of link prediction for our approach and all baselines in Table 2 (* means the mean of ranks for random walk methods are evaluated in the Top-K subset), and we can obtain the following observations:

1) Our approach achieves good performances on both WN18 and FB15K. On the FB15K, our approach outperforms all baselines. It indicates that our approach is effective for inference. On the WN18, three random walk methods have similar performances. The reason is that most entities in WN18 only have a small number of neighbors, so RW and PRA can also find useful structures in a few rounds.

2) For FB15K, the performances of RW and PRA are both poor and even worse than a part of embedding-based methods, but the performance of our approach is still the best. The reason is that there are too many relation types in FB15K, so goalless random walks introduce lots of noise. Oppositely, our approach has a great capability of resisting noise for the goal-directed mechanism.

3) RW and PRA have similar performances on both datasets, which indicates the heuristic rule of PRA does not apply to all relations and formulas.

4.5 Paths Recall by Random Walks

To further explore whether the goal-directed mechanism can increase efficiency of mining paths, we compare the three random walk methods by the number of paths mined. For each triplet $R(H, T)$ in the training set, we perform 10 rounds of random walks from H and record the number of times which arrive at T, noted as Arr@10. We respectively select one relation type from WN18 and FB15K and show the comparison result in Figure 2. We can obtain the following observations:

1) With the increase of training epochs, $Arr@10$ of the goal-directed random walk first increases and then stays around a high value on both WN18 and FB15K, but the $Arr@10$ of RW and PRA always stay the same. This phenomenon indicates that the goal-directed random walk is a learnable model and can be trained to find more useful structures with epochs increasing, but RW and PRA are not.

2) RW and PRA always have similar $Arr@10$, which means PRA has not found more formulas. This indicates that the heuristic rule of PRA is not always be beneficial to mining more structures for all relations.

4.6 Example Formulas

In Table 3, we show a small number of formulas mined by our approach from FB15K, and the formulas represent different types. Some formulas contain clear logic, e.g, Formula 1 means that if the writer x contributes a story to the film y and y is adapted from the book z, x is the writer of the book z. Some formulas have a high probability of being satisfied, e.g., Formula 3 means the wedding place probably is also the burial place for some people, and Formula 7 means the parent of the person x died of the disease and thus the person x has a high risk of suffering from the disease. Some formulas depend on synonyms, e.g., *story_by* and *works_written* have the similar meaning in Formula 2. However, there are still useless formulas, e.g, Formula 8 is useless be-

Relation	Formula
works_written	
1	film_story_contributor(x,y) ∧ adapted_from(y,z) ⇒ works_written(x,z)
2	story_by(y,x) ⇒ works_written(x,y)
place_of_burial	
3	place_of_death(x,y) ⇒ place_of_burial(x,y)
4	marriage_type_of_union(x,y) ∧ marriage_location_of_ceremony(y,z) ⇒ place_of_burial(x,z)
service_language	
5	service_location(x,y) ∧ imported_from(y,z) ∧ official_language(z,w) ⇒ service_language(x,w)
6	service_location(x,y) ∧ exported_to(y,z) ∧ languages_spoken(z,w) ⇒ service_language(x,w)
disease_risk_factors	
7	parent_cause_of_death(x,y) ∧ disease_risk_factors(y,z) ⇒ disease_risk_factors(x,z)
8	disease_risk_factors(x,y)∧ -disease_risk_factors(y,x) ⇒ disease_risk_factors(x,y)

Table 3: Example Formulas Obtained by Goal-directed Random Walk

cause the body of the formula is same as the head. Such useless formula can be removed by a super-rule, which is that the head of a formula cannot occur in its body.

5 Related Work

Our work has two aspects, which are related to mining formula automatically and inference on KBs, respectively.

Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994) and Association Rule Mining (ARM) (Agrawal et al., 1993) are both early works on mining formulas. FOIT (Quinlan, 1990) and SHERLOCK (Schoenmackers et al., 2010) are typical ILP systems, but the former one usually need a lot of negative facts and the latter one focuses on mining formulas from text. AMIE (Galárraga et al., 2013) is based on ARM and proposes a new measure for formulas instead of the confidence. Several structure learning algorithms (Kok and Domingos, 2005; Kok and Domingos, 2009; Kok and Domingos, 2010) based on Markov Logic Network (MLN) (Richardson and Domingos, 2006) can also learn first order logic formulas automatically, but they are too slow to run on large KBs. ProPPR (Wang et al., 2013; Wang et al., 2014a) performs structure learning by depth first searching on the knowledge graph, which is still not efficient enough to handle web-scale KBs. PRA (Lao and Cohen, 2010; Lao et al., 2011) is a method based on random walks and employs heuristic rules to direct random walks. PRA is closely related to our approach, but unlike it, our approach dynamically calculates state transition probabilities. Another method based on random walks (Wei et al., 2015) merges embedding similarities of candidates into the random walk as a priori, while our approach employs KB embeddings to calculate potentials for neighbors.

The majority of mining formula methods can perform inference on KBs, and besides them, a dozen methods based KB embeddings can also achieve the inference goal, and the typical ones of them are TransE (Bordes et al., 2013), Rescal (Nickel et al., 2011), TransH (Wang et al., 2014b), TransR (Lin et al., 2015b). These embedding-based methods take advantage of the implicit relationship between elements of the KB and perform inference by calculating similarities. There are also methods which combine inference formulas and KB embeddings, such as PTransE (Lin et al., 2015a) and ProPPR+MF (Wang and Cohen, 2016).

6 Conclusion and Future Works

In this paper, we introduce a goal-directed random walk algorithm to increase efficiency of mining useful formulas and decrease noise simultaneously. The approach employs the inference target as the direction at each steps in the random walk process and is more inclined to visit structures helpful to inference. In empirical studies, we show our approach achieves good performances on link prediction task over large-scale KBs. In the future, we are interested in exploring mining formulas directly in the distributional spaces which may resolve the sparsity of formulas.

7 Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018), the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332). And this work was also supported by Google through focused research awards program.

References

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. International World Wide Web Conferences Steering Committee.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Stanley Kok and Pedro Domingos. 2005. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM.

Stanley Kok and Pedro Domingos. 2009. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the 26th annual international conference on machine learning*, pages 505–512. ACM.

Stanley Kok and Pedro Domingos. 2010. Learning markov logic networks using structural motifs. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 551–558.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Stephen Muggleton and Luc De Raedt. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

J. Ross Quinlan. 1990. Learning logical definitions from relations. *Machine learning*, 5(3):239–266.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. 2002. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2014a. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1199–1208. ACM.

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

Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1331–1340. ACM.

Lifted Rule Injection for Relation Embeddings

Thomas Demeester
Ghent University - iMinds
Ghent, Belgium
tdmeeste@intec.ugent.be

Tim Rocktäschel and **Sebastian Riedel**
University College London
London, UK
{t.rocktaschel,s.riedel}@cs.ucl.ac.uk

Abstract

Methods based on representation learning currently hold the state-of-the-art in many natural language processing and knowledge base inference tasks. Yet, a major challenge is how to efficiently incorporate commonsense knowledge into such models. A recent approach regularizes relation and entity representations by propositionalization of first-order logic rules. However, propositionalization does not scale beyond domains with only few entities and rules. In this paper we present a highly efficient method for incorporating implication rules into distributed representations for automated knowledge base construction. We map entity-tuple embeddings into an approximately Boolean space and encourage a partial ordering over relation embeddings based on implication rules mined from WordNet. Surprisingly, we find that the strong restriction of the entity-tuple embedding space does not hurt the expressiveness of the model and even acts as a regularizer that improves generalization. By incorporating few commonsense rules, we achieve an increase of 2 percentage points mean average precision over a matrix factorization baseline, while observing a negligible increase in runtime.

1 Introduction

Current successful methods for automated knowledge base construction tasks heavily rely on learned distributed vector representations (Nickel et al., 2012; Riedel et al., 2013; Socher et al., 2013; Chang et al., 2014; Neelakantan et al., 2015; Toutanova et al., 2015; Nickel et al., 2015; Verga et al., 2016; Verga and McCallum, 2016). Although these models are able to learn robust representations from large amounts of data, they often lack commonsense knowledge. Such knowledge is rarely explicitly stated in texts but can be found in resources like PPDB (Ganitkevitch et al., 2013) or WordNet (Miller, 1995).

Combining neural methods with symbolic commonsense knowledge, for instance in the form of implication rules, is in the focus of current research (Rocktäschel et al., 2014; Wang et al., 2014; Bowman et al., 2015; Wang et al., 2015; Vendrov et al., 2016; Hu et al., 2016; Rocktäschel and Riedel, 2016; Cohen, 2016). A recent approach (Rocktäschel et al., 2015) regularizes entity-tuple and relation embeddings via first-order logic rules. To this end, every first-order rule is propositionalized based on observed entity-tuples, and a differentiable loss term is added for every propositional rule. This approach does not scale beyond only a few entity-tuples and rules. For example, propositionalizing the rule $\forall x : \texttt{isMan}(x) \Rightarrow \texttt{isMortal}(x)$ would result in a very large number of loss terms on a large database.

In this paper, we present a method to incorporate simple rules while maintaining the computational efficiency of only modeling training facts. This is achieved by minimizing an upper bound of the loss that encourages the implication between relations to hold, entirely independent from the number of entity pairs. It only involves representations of the relations that are mentioned in rules, as well as a general rule-independent constraint on the entity-tuple embedding space. In the example given above, if we require that every component of the

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1389–1399,
Austin, Texas, November 1-5, 2016.

vector representation of `isMan` is smaller than the corresponding component of relation `isMortal`, then we can show that the rule holds for any *non-negative* representation of an entity-tuple. Hence our method avoids the need for separate loss terms for every ground atom resulting from propositionalizing rules. In statistical relational learning this type of approach is often referred to as *lifted* inference or learning (Poole, 2003; Braz, 2007) because it deals with groups of random variables at a first-order level. In this sense our approach is a lifted form of rule injection. This allows for imposing large numbers of rules while learning distributed representations of relations and entity-tuples. Besides drastically lower computation time, an important advantage of our method over Rocktäschel et al. (2015) is that when these constraints are satisfied, the injected rules always hold, even for unseen but inferred facts. While the method presented here only deals with implications and not general first-order rules, it does not rely on the assumption of independence between relations, and is hence more generally applicable.

Our contributions are fourfold: (i) we develop a very efficient way of regularizing relation representations to incorporate first-order logic implications (§3), (ii) we reveal that, against expectation, mapping entity-tuple embeddings to non-negative space does not hurt but instead improves the generalization ability of our model (§5.1) (iii) we show improvements on a knowledge base completion task by injecting mined commonsense rules from WordNet (§5.3), and finally (iv) we give a qualitative analysis of the results, demonstrating that implication constraints are indeed satisfied in an asymmetric way and result in a substantially increased structuring of the relation embedding space (§5.6).

2 Background

In this section we revisit the matrix factorization relation extraction model by Riedel et al. (2013) and introduce the notation used throughout the paper. We choose the matrix factorization model for its simplicity as the base on which we develop implication injection.

Riedel et al. (2013) represent every relation $r \in \mathcal{R}$ (selected from Freebase (Bollacker et al., 2008) or extracted as textual surface pattern) by a k-dimensional latent representation $\boldsymbol{r} \in \mathbb{R}^k$. A particular *relation instance* or *fact* is the combination of a relation r and a tuple t of entities that are engaged in that relation, and is written as $\langle r, t\rangle$. We write $\mathcal{O}$ as the set of all such input facts available for training. Furthermore, every entity-tuple $t \in \mathcal{T}$ is represented by a latent vector $\boldsymbol{t} \in \mathbb{R}^k$ (with $\mathcal{T}$ the set of all entity-tuples in $\mathcal{O}$).

Model F by Riedel et al. (2013) measures the compatibility between a relation r and an entity-tuple t using the dot product $\boldsymbol{r}^\top \boldsymbol{t}$ of their respective vector representations. During training, the representations are learned such that valid facts receive high scores, whereas negative ones receive low scores. Typically no negative evidence is available at training time, and therefore a Bayesian Personalized Ranking (BPR) objective (Rendle et al., 2009) is used. Given a pair of facts $f_p := \langle r_p, t_p\rangle \notin \mathcal{O}$ and $f_q := \langle r_q, t_q\rangle \in \mathcal{O}$, this objective requires that

$$\boldsymbol{r_p}^\top \boldsymbol{t_p} \leq \boldsymbol{r_q}^\top \boldsymbol{t_q}. \tag{1}$$

The embeddings can be trained by minimizing a convex loss function ℓ_R that penalizes violations of that requirement when iterating over the training set. In practice, each positive training fact $\langle r, t_q\rangle$ is compared with a randomly sampled unobserved fact $\langle r, t_p\rangle$ for the same relation. The overall loss can hence be written as

$$\mathcal{L}_R = \sum_{\substack{\langle r,t_q\rangle \in \mathcal{O} \\ t_p \in \mathcal{T},\ \langle r,t_p\rangle \notin \mathcal{O}}} \ell_R\big(\boldsymbol{r}^\top [\boldsymbol{t_p} - \boldsymbol{t_q}]\big). \tag{2}$$

and measures how well observed valid facts are ranked above unobserved facts, thus reconstructing the ranking of the training data. We will henceforth call $\mathcal{L}_R$ the *reconstruction loss*, to make a distinction with the *implication loss* that we will introduce later. Riedel et al. (2013) use the logistic loss $\ell_R(s) := -\log \sigma(-s)$, where $\sigma(s) := (1 + e^{-x})^{-1}$ denotes the sigmoid function. In order to avoid overfitting, an L_2 regularization term on the $\boldsymbol{r}$ and $\boldsymbol{t}$ embeddings is added to the reconstruction loss. The overall objective to minimize hence is

$$\mathcal{L}_F = \mathcal{L}_R + \alpha\big(\textstyle\sum_{\boldsymbol{r}} \|\boldsymbol{r}\|_2^2 + \sum_{\boldsymbol{t}} \|\boldsymbol{t}\|_2^2\big) \tag{3}$$

where α is the regularization strength.

3 Lifted Injection of Implications

In this section, we show how an implication

$$\forall t \in \mathcal{T} : \langle r_p, t\rangle \Rightarrow \langle r_q, t\rangle, \tag{4}$$

can be imposed independently of the entity-tuples. For simplicity, we abbreviate such implications as $r_p \Rightarrow r_q$ (*e.g.*, `professorAt` $\Rightarrow$ `employeeAt`).

3.1 Grounded Loss Formulation

The implication rule can be imposed by requiring that every tuple $t \in \mathcal{T}$ is at least as compatible with relation r_p as with r_q. Written in terms of the latent representations, eq. (4) therefore becomes

$$\forall t \in \mathcal{T} : \boldsymbol{r_p}^\top \boldsymbol{t} \leq \boldsymbol{r_q}^\top \boldsymbol{t} \tag{5}$$

If $\langle r_p, t\rangle$ is a true fact with a high score $\boldsymbol{r_p}^\top \boldsymbol{t}$, and the fact $\langle r_q, t\rangle$ has an even higher score, it must also be true, but not vice versa. We can therefore inject an implication rule by minimizing a loss term with a separate contribution from every $t \in \mathcal{T}$, adding up to the total loss if the corresponding inequality is not satisfied. In order to make the contribution of every tuple t to that loss independent of the magnitude of the tuple embedding, we divide both sides of the above inequality by $\|\boldsymbol{t}\|_1$. With $\tilde{\boldsymbol{t}} := \boldsymbol{t}/\|\boldsymbol{t}\|_1$, the implication loss for the rule $r_p \Rightarrow r_q$ can be written as

$$\mathcal{L}_I = \sum_{\forall t \in \mathcal{T}} \ell_I\big([\boldsymbol{r_p} - \boldsymbol{r_q}]^\top \tilde{\boldsymbol{t}}\big) \tag{6}$$

for an appropriate convex loss function ℓ_I, similarly to eq. (2). In practice, the summation can be reduced to those tuples that occur in combination with r_p or r_q in the training data. Still, the propositionalization in terms of training facts leads to a heavy computational cost for imposing a single implication, similar to the technique introduced in Rocktäschel et al. (2015). Moreover, with that simplification there is no guarantee that the implication between both relations would generalize towards inferred facts not seen during training.

3.2 Lifted Loss Formulation

The problems mentioned above can be avoided if instead of $\mathcal{L}_I$, a tuple-independent upper bound is minimized. Such a bound can be constructed, provided all components of $\boldsymbol{t}$ are restricted to a non-negative embedding space, *i.e.*, $\mathcal{T} \subseteq \mathbb{R}^{k,+}$. If this holds, Jensen's inequality allows us to transform eq. (6) as follows

$$\mathcal{L}_I = \sum_{\forall t \in \mathcal{T}} \ell_I\Big(\sum_{i=1}^{k} \tilde{t}_i \,[\boldsymbol{r_p} - \boldsymbol{r_q}]^\top \mathbf{1}_i\Big) \tag{7}$$

$$\leq \sum_{i=1}^{k} \ell_I\big([\boldsymbol{r_p} - \boldsymbol{r_q}]^\top \mathbf{1}_i\big) \sum_{\forall t \in \mathcal{T}} \tilde{t}_i \tag{8}$$

where $\mathbf{1}_i$ is the unit vector along dimension i in tuple-space. This is allowed because the $\{\tilde{t}_i\}_{i=1}^k$ form convex coefficients ($\tilde{t}_i > 0$, and $\sum_i \tilde{t}_i = 1$), and ℓ_I is a convex function. If we define

$$\mathcal{L}_I^U := \sum_{i=1}^{k} \ell_I\big([\boldsymbol{r_p} - \boldsymbol{r_q}]^\top \mathbf{1}_i\big) \tag{9}$$

we can write

$$\mathcal{L}_I \leq \beta \mathcal{L}_I^U \tag{10}$$

in which β is an upper bound on $\sum_t \tilde{t}_i$. One such bound is $|\mathcal{T}|$, but others are conceivable too. In practice we rescale β to a hyper-parameter $\tilde{\beta}$ that we use to control the impact of the upper bound to the overall loss. We call $\mathcal{L}_I^U$ the *lifted loss*, as it no longer depends on any of the entity-tuples; it is grounded over the unit tuples $\mathbf{1}_i$ instead.

The implication $r_p \Rightarrow r_q$ can thus be imposed by minimizing the lifted loss $\mathcal{L}_I^U$. Note that by minimizing $\mathcal{L}_I^U$, the model is encouraged to satisfy the constraint $\boldsymbol{r_p} \leq \boldsymbol{r_q}$ on the relation embeddings, where $\leq$ denotes the component-wise comparison. In fact, a sufficient condition for eq. (5) to hold, is

$$\boldsymbol{r_p} \leq \boldsymbol{r_q} \;\text{ and }\; \forall t \in \mathcal{T} : \; \boldsymbol{t} \geq \mathbf{0} \tag{11}$$

with $\mathbf{0}$ the k-dimensional null vector. This corresponds to a single relation-specific loss term, and the general restriction $\mathcal{T} \subseteq \mathbb{R}^{k,+}$ on the tuple-embedding space.

3.3 Approximately Boolean Entity Tuples

In order to impose implications by minimizing a lifted loss $\mathcal{L}_I^U$, the tuple-embedding space needs to be restricted to $\mathbb{R}^{k,+}$. We have chosen to restrict the tuple space even more than required, namely to the hypercube $\boldsymbol{t} \in [0,1]^k$, as approximately Boolean embeddings (Kruszewski et al., 2015). The tuple

embeddings are constructed from real-valued vectors $\boldsymbol{e}$, using the component-wise sigmoid function

$$\boldsymbol{t} = \sigma(\boldsymbol{e}), \quad \boldsymbol{e} \in \mathbb{R}^k. \tag{12}$$

For minimizing the loss, the gradients are hence computed with respect to $\boldsymbol{e}$, and the L_2 regularization is applied to the components of $\boldsymbol{e}$ instead of $\boldsymbol{t}$.

Other choices for ensuring the restriction $\boldsymbol{t} \geq \boldsymbol{0}$ in eq. (11) are possible, but we found that our approach works better in practice than those (*e.g.*, the exponential transformation proposed by Demeester et al. (2016)). It can also be observed that the unit tuples over which the implication loss is grounded, form a special case of approximately Boolean embeddings.

In order to investigate the impact of this restriction even when not injecting any rules, we introduce model FS: the original model F, but with sigmoidal entity-tuples:

$$\mathcal{L}_{FS} = \sum_{\substack{\langle r,t_q\rangle \in \mathcal{O} \\ t_p \in \mathcal{T},\, \langle r,t_p\rangle \notin \mathcal{O}}} \ell_R\big(\boldsymbol{r}^\top [\sigma(\boldsymbol{e_p}) - \sigma(\boldsymbol{e_q})]\big) + \alpha\big(\textstyle\sum_{\boldsymbol{r}} \|\boldsymbol{r}\|_2^2 + \sum_{\boldsymbol{e}} \|\boldsymbol{e}\|_2^2\big) \tag{13}$$

Here, $\boldsymbol{e_p}$ and $\boldsymbol{e_q}$ are the real-valued representations as in eq. (12), for tuples t_p and t_q, respectively.

With the above choice of a non-negative tuple-embedding space we can now state the full lifted rule injection model (FSL):

$$\mathcal{L}_{FSL} = \mathcal{L}_{FS} + \tilde{\beta} \sum_{I \in \mathcal{I}} \mathcal{L}_I^U \tag{14}$$

$\mathcal{L}_I^U$ denotes a lifted loss term for every rule in a set $\mathcal{I}$ of implication rules that we want to inject.

3.4 Convex Implication Loss

The logistic loss ℓ_R (see §2) is not suited for imposing implications because once the inequality in eq. (11) is satisfied, the components of $\boldsymbol{r_p}$ and $\boldsymbol{r_q}$ do not need to be separated any further. However, with ℓ_R this would continue to happen due to the small non-zero gradient. In the reconstruction loss $\mathcal{L}_R$ this is a desirable effect which further separates the scores for positive from negative examples. However, if an implication is imposed between two relations that are almost equivalent according to the training data, we still want to find almost equivalent embedding vectors. Hence, we propose to use the loss

$$\ell_I(s) = \max(0, s + \delta) \tag{15}$$

with δ a small positive margin to ensure that the gradient does not disappear before the inequality is actually satisfied. We use $\delta = 0.01$ in all experiments.

The main advantage of the presented approach over earlier methods that impose the rules in a grounded way (Rocktäschel et al., 2015; Wang et al., 2015) is the computational efficiency of imposing the lifted loss. Evaluating $\mathcal{L}_I^U$ or its gradient for one implication rule is comparable to evaluating the reconstruction loss for one pair of training facts. In typical applications there are much fewer rules than training facts and the extra computation time needed to inject these rules is therefore negligible.

4 Related Work

Recent research on combining rules with learned vector representations has been important for new developments in the field of knowledge base completion. Rocktäschel et al. (2014) and Rocktäschel et al. (2015) provided a framework to jointly maximize the probability of observed facts and propositionalized first-order logic rules. Wang et al. (2015) demonstrated how different types of rules can be incorporated using an Integer Linear Programming approach. Wang and Cohen (2016) learned embeddings for facts and first-order logic rules using matrix factorization. Yet, all of these approaches ground the rules in the training data, limiting their scalability towards large rule sets and KBs with many entities. As argued in the introduction, this forms an important motivation for the lifted rule injection model put forward in this work, which by construction does not suffer from that limitation. Wei et al. (2015) proposed an alternative strategy to tackle the scalability problem by reasoning on a filtered subset of grounded facts.

Wu et al. (2015) proposed to use a path ranking approach for capturing long-range interactions between entities, and to add these as an extra loss term, besides the loss that models pairwise relations. Our model FSL differs substantially from their approach, in that we consider tuples instead of separate entities, and we inject a given set of rules. Yet, by cre-

ating a partial ordering in the relation embeddings as a result of injecting implication rules, model FSL can also capture interactions beyond direct relations. This will be demonstrated in §5.3 by injecting rules between surface patterns only and still measuring an improvement on predictions for structured Freebase relations.

Combining logic and distributed representations is also an active field of research outside of automated knowledge base completion. Recent advances include the work by Faruqui et al. (2014), who injected ontological knowledge from WordNet into word representations. Furthermore, Vendrov et al. (2016) proposed to enforce a partial ordering in an embeddings space of images and phrases. Our method is related to such order embeddings since we define a partial ordering on relation embeddings. However, to ensure that implications hold for all entity-tuples we also need a restriction on the entity-tuple embedding space and derive bounds on the loss. Another important contribution is the recent work by Hu et al. (2016), who proposed a framework for injecting rules into general neural network architectures, by jointly training on the actual targets and on the rule-regularized predictions provided by a teacher network. Although quite different at first sight, their work could offer a way to use our model in various neural network architectures, by integrating the proposed lifted loss into the teacher network.

This paper builds upon our previous workshop paper (Demeester et al., 2016). In that work, we tested different tuple embedding transformations in an ad-hoc manner. We used approximately Boolean representations of relations instead of entity-tuples, strongly reducing the model's degrees of freedom. We now derive the FSL model from a carefully considered mathematical transformation of the grounded loss. The FSL model only restricts the tuple embedding space, whereby relation vectors remain real valued. Furthermore, previous experiments were performed on small-scale artificial datasets, whereas we now test on a real-world relation extraction benchmark.

Finally, we explicitly discuss the main differences with respect to the strongly related work from Rocktäschel et al. (2015). Their method is more general, as they cover a wide range of first-order logic rules, whereas we only discuss implications. Lifted rule injection beyond implications will be studied in future research contributions. However, albeit less general, our model has a number of clear advantages:

Scalability – Our proposed model of lifted rule injection scales according to the number of implication rules, instead of the number of rules times the number of observed facts for every relation present in a rule.

Generalizability – Injected implications will hold even for facts not seen during training, because their validity only depends on the order relation imposed on the relation representations. This is not guaranteed when training on rules grounded in training facts by Rocktäschel et al. (2015).

Training Flexibility – Our method can be trained with various loss functions, including the rank-based loss as used in Riedel et al. (2013). This was not possible for the model of Rocktäschel et al. (2015) and already leads to an improved accuracy as seen from the zero-shot learning experiment in §5.2.

Independence Assumption – In Rocktäschel et al. (2015) an implication of the form $a_p \Rightarrow a_q$ for two ground atoms a_p and a_q is modeled by the logical equivalence $\neg(a_p \wedge \neg a_q)$, and its probability is approximated in terms of the elementary probabilities $\pi(a_p)$ and $\pi(a_q)$ as $1 - \pi(a_p)\big(1 - \pi(a_q)\big)$. This assumes the independence of the two atoms a_p and a_q, which may not hold in practice. Our approach does not rely on that assumption and also works for cases of statistical dependence. For example, the independence assumption does not hold in the trivial case where the relations r_p and r_q in the two atoms are equivalent, whereas in our model, the constraints $\boldsymbol{r}_p \leq \boldsymbol{r}_q$ and $\boldsymbol{r}_p \geq \boldsymbol{r}_q$ would simply reduce to $\boldsymbol{r}_p = \boldsymbol{r}_q$.

5 Experiments and Results

We now present our experimental results. We start by describing the experimental setup and hyperparameters. Before turning to the injection of rules, we compare model F with model FS, and show that restricting the tuple embedding space has a regularization effect, rather than limiting the expressiveness of the model (§5.1). We then demonstrate that model FSL is capable of zero-shot learning (§5.2), and show that injecting high-quality WordNet rules

Test relation	#	R13-F	F	FS	FSL
person/company	106	0.75	0.73	0.74	**0.77**
location/containedby	73	0.69	0.62	0.70	**0.71**
person/nationality	28	0.19	0.20	0.20	**0.21**
author/works_written	27	0.65	**0.71**	0.69	0.65
person/place_of_birth	21	0.72	0.69	**0.72**	0.70
parent/child	19	0.76	0.77	0.81	**0.85**
person/place_of_death	19	0.83	**0.85**	0.83	**0.85**
neighborhood/neighborhood_of	11	**0.70**	0.67	0.63	0.62
person/parents	6	0.61	0.53	**0.66**	**0.66**
company/founders	4	**0.77**	0.73	0.64	0.67
sports_team/league	4	**0.59**	0.44	0.43	0.56
team_owner/teams_owned	2	0.38	**0.64**	**0.64**	0.61
team/arena_stadium	2	**0.13**	**0.13**	**0.13**	0.12
film/directed_by	2	**0.50**	0.18	0.17	0.13
broadcast/area_served	2	0.58	0.83	0.83	**1.00**
structure/architect	2	**1.00**	**1.00**	**1.00**	**1.00**
composer/compositions	2	**0.67**	0.64	0.51	0.50
person/religion	1	**1.00**	**1.00**	**1.00**	**1.00**
film/produced_by	1	0.50	**1.00**	**1.00**	0.33
Weighted MAP		0.67	0.65	0.67	0.69

Table 1: Weighted mean average precision for our reimplementation of the matrix factorization model (F) compared to restricting the entity-pair space (FS) and injecting WordNet rules (FSL). Model F results by Riedel et al. (2013) are denoted as R13-F.

leads to an improved precision (§5.3). We proceed with a visual illustration of the relation embeddings with and without injected rules (§5.4), provide details on time efficiency of the lifted rule injection method (§5.5), and show that it correctly captures the asymmetry of implication rules (§5.6).

All models were implemented in TensorFlow (Abadi et al., 2015). We use the hyperparameters of Riedel et al. (2013), with $k = 100$ hidden dimensions and a weight of $\alpha = 0.01$ for the L_2 regularization loss. We use ADAM (Kingma and Ba, 2014) for optimization with an initial learning rate of 0.005 and a mini-batch size of 8192. The embeddings are initialized by sampling uniformly from $[-0.1, 0.1]$ and we use $\tilde{\beta} = 0.1$ for the implication loss throughout our experiments.

5.1 Restricted Embedding Space

Before incorporating external commonsense knowledge into relation representations, we were curious how much we lose by restricting the entity-tuple space to approximately Boolean embeddings. We evaluate our models on the New York Times dataset introduced by Riedel et al. (2013). Surprisingly, we find that the expressiveness of the model does not suffer from this strong restriction. From Table 1 we see that restricting the tuple-embedding space seems to perform slightly better (FS) as opposed to a real-valued tuple-embedding space (F), suggesting that this restriction has a regularization effect that improves generalization. We also provide the original results for model F by Riedel et al. (2013) (denoted as R13-F) for comparison. Due to a different implementation and optimization procedure, the results for our model F and R13-F are not identical.

Inspecting the top relations for a sampled dimension in the embedding space reveals that the relation space of model FS more closely resembles clusters than that of model F (Table 2). We hypothesize that this might be caused by approximately Boolean entity-tuple representations in model FS, resulting in attribute-like entity-tuple vectors that capture which relation clusters they belong to.

5.2 Zero-shot Learning

The zero-shot learning experiment performed in Rocktäschel et al. (2015) leads to an important finding: when injecting implications with right-hand sides for Freebase relations for which no or very limited training facts are available, the model should be able to infer the validity of Freebase facts for those relations based on rules and correlations between textual surface patterns.

We inject the same hand-picked relations as used by Rocktäschel et al. (2015), after removing all Freebase training facts. The lifted rule injection (model FSL) reaches a weighted MAP of 0.35, comparable with 0.38 by the Joint model from Rocktäschel et al. (2015) (denoted R15-Joint). Note that for this experiment we initialized the Freebase relations implied by the rules with negative random vectors (sampled uniformly from $[-7.9, -8.1]$). The reason is that without any negative training facts for these relations, their components can only go up due to the implication loss, and we do not want to get values that are too high before optimization.

Figure 1 shows how the relation extraction performance improves when more Freebase relation training facts are added. It effectively measures how well the proposed models, matrix factorization (F), propositionalized rule injection (R15-Joint), and our model (FSL), can make use of the provided rules and correlations between textual surface form pat-

Table 2: Top patterns for a randomly sampled dimension in non-restricted and restricted embedding space .

Model F (non-restricted)	Model FS (restricted)
nsubj<-represent->dobj	rcmod->return->prep->to->pobj
appos->member->prep->of->pobj->team->nn	nn<-return->prep->to->pobj
nsubj<-die->dobj	nsubj<-return->prep->to->pobj
nsubj<-speak->prep->about->pobj	rcmod->leave->dobj
appos->champion->poss	nsubj<-quit->dobj

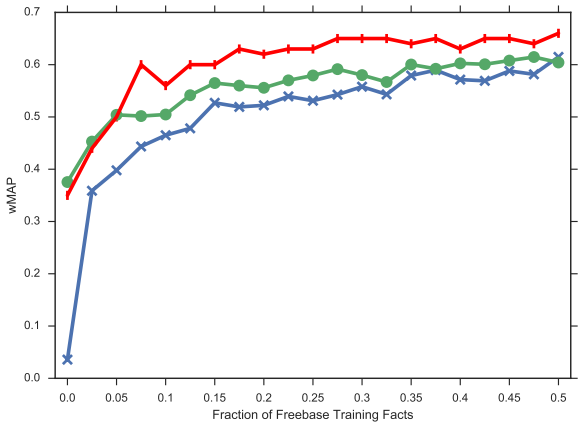


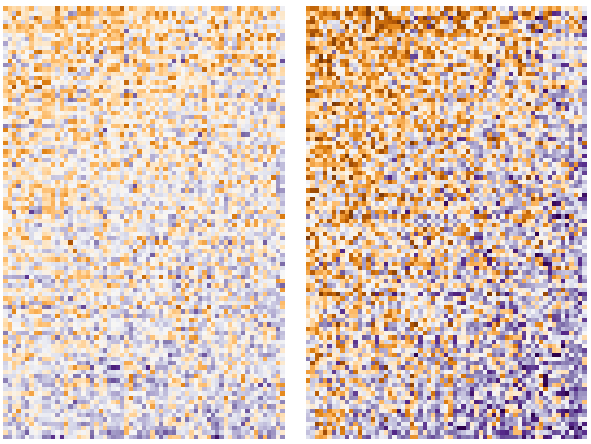
Figure 1: Weighted MAP for injecting hand-picked rules as a function of the fraction of Freebase training facts. Comparison between model F (lowest, in blue), R15-Joint (middle, in green) and model FSL (highest, in red).

terns and increased fractions of Freebase training facts. Although FSL starts at a lower performance than R15-Joint when no Freebase training facts are present, it outperforms R15-Joint and a plain matrix factorization model by a substantial margin when provided with more than 7.5% of Freebase training facts. This indicates that, in addition to being much faster than R15-Joint, it can make better use of provided rules and few training facts. We attribute this to the Bayesian personalized ranking loss instead of the logistic loss used in Rocktäschel et al. (2015). The former is compatible with our rule-injection method, but not with the approach of maximizing the expectation of propositional rules used by R15-Joint.

5.3 Injecting Knowledge from WordNet

The main purpose of this work is to be able to incorporate rules from external resources for aiding relation extraction. We use WordNet hypernyms to generate rules for the NYT dataset. To this end we iterate over all surface form patterns in the dataset and attempt to replace words in the pattern by their hypernyms. If the resulting pattern is contained in the dataset, we generate the corresponding rule. For instance, we generate a rule `appos->diplomat->amod` ⇒ `appos->official->amod` since both patterns are contained in the NYT dataset and we know from WordNet that a diplomat is an official. This leads to 427 rules from WordNet that we subsequently annotate manually to obtain 36 high-quality rules. Note that none of these rules directly imply a Freebase relation. Although the test relations all originate from Freebase, we still hope to see improvements by transitive effects, *i.e.*, better surface form representations that in turn help to predict Freebase facts.

We show results obtained by injecting these WordNet rules in Table 1 (column FSL). The weighted MAP measure increases by 2% with respect to model FS, and 4% compared to our reimplementation of the matrix factorization model F. This demonstrates that imposing a partial ordering based on implication rules can be used to incorporate logical commonsense knowledge and increase the quality of information extraction systems. Note that our evaluation setting guarantees that only indirect effects of the rules are measured, *i.e.*, we do not use any rules directly implying test relations. This shows that injecting such rules influences the relation embedding space beyond only the relations explicitly stated in the rules. For example, injecting the rule `appos<-father->appos` ⇒ `poss<-parent->appos` can contribute to improved predictions for the test relation `parent/child`.



(a) (b)

Figure 2: Visualization of embeddings (columns) for the relations that appear in the high-quality WordNet rules, (a) without and (b) with injection of these rules. Values range from -1 (orange) via 0 (white) to 1 (purple). Best viewed in color.

5.4 Visualizing Relation Embeddings

We provide a visual inspection of how the structure of the relation embedding space changes when rules are imposed. We select all relations involved in the WordNet rules, and gather them as columns in a single matrix, sorted by increasing ℓ_1 norm (values in the 100 dimensions are similarly sorted). Figures 2a and 2b show the difference between model F (without injected rules) and FSL (with rules). The values of the embeddings in model FSL are more polarized, *i.e.*, we observe stronger negative or positive components than for model F. Furthermore, FSL also reveals a clearer difference between the leftmost (mostly negative, more specific) and right-most (predominantly positive, more general) embeddings (*i.e.*, a clearer separation between positive and negative values in the plot), which results from imposing the order relation in eq. (11) when injecting implications.

5.5 Efficiency of Lifted Injection of Rules

In order to get an idea of the time efficiency of injecting rules, we measure the time per epoch when restricting the program execution to a single 2.4GHz CPU core. We measure on average 6.33s per epoch without rules (model FS), against 6.76s and 6.97s when injecting the 36 high-quality WordNet rules and the unfiltered 427 rules (model FSL), respectively. Increasing the amount of injected rules from 36 to 427 leads to an increase of only 3% in computation time, even though in our setup all rule losses are used in every training batch. This confirms the high efficiency of our lifted rule injection method.

5.6 Asymmetric Character of Implications

In order to demonstrate that injecting implications conserves their asymmetric nature, we perform the following experiment. After incorporating high-quality Wordnet rules $r_p \Rightarrow r_q$ into model FSL we select all of the tuples t_p that occur with relation r_p in a training fact $\langle r_p, t_p \rangle$. Matching these with relation r_q should result in high values for the scores $\boldsymbol{r_q}^\top \boldsymbol{t_p}$, if the implication holds. If however the tuples t_q are selected from the training facts $\langle r_q, t_q \rangle$, and matched with relation r_p, the scores $\boldsymbol{r_p}^\top \boldsymbol{t_q}$ should be much lower if the inverse implication does not hold (in other words, if r_q and r_p are not equivalent). Table 3 lists the averaged results for 5 example rules, and the average over all relations in WordNet rules, both for the case with injected rules (model FSL), and without rules (model FS). For easier comparison, the scores are mapped to the unit interval via the sigmoid function. This quantity $\sigma(\boldsymbol{r}^\top \boldsymbol{t})$ is often interpreted as the probability that the corresponding fact holds (Riedel et al., 2013), but because of the BPR-based training, only differences between scores play a role here. After injecting rules, the average scores of facts inferred by these rules (*i.e.*, column $\sigma(\boldsymbol{r_q}^\top \boldsymbol{t_p})$ for model FSL) are always higher than for facts (incorrectly) inferred by the inverse rules (column $\sigma(\boldsymbol{r_p}^\top \boldsymbol{t_q})$ for model FSL). In the fourth example, the inverse rule leads to high scores as well (on average 0.79, vs. 0.98 for the actual rule). This is due to the fact that the `daily` and `newspaper` relations are more or less equivalent, such that the components of $\boldsymbol{r_p}$ are not much below those of $\boldsymbol{r_q}$. For the last example (the `ambassador` $\Rightarrow$ `diplomat` rule), the asymmetry in the implication is maintained, although the absolute scores are rather low for these two relations.

The results for model FS reflect how strongly the implications in either direction are latently present in the training data. We can only conclude that model FS manages to capture the similarity be-

rule			model FSL		model FS	
r_p	$\Rightarrow$	r_q	$\sigma(\boldsymbol{r_q}^\top \boldsymbol{t_p})$	$\sigma(\boldsymbol{r_p}^\top \boldsymbol{t_q})$	$\sigma(\boldsymbol{r_q}^\top \boldsymbol{t_p})$	$\sigma(\boldsymbol{r_p}^\top \boldsymbol{t_q})$
appos->party->amod	$\Rightarrow$	appos->organization->amod	0.99	0.22	0.70	0.86
poss<-father->appos	$\Rightarrow$	poss<-parent->appos	0.96	0.00	0.72	0.89
appos->prosecutor->nn	$\Rightarrow$	appos->lawyer->nn	0.99	0.01	0.87	0.80
appos->daily->amod	$\Rightarrow$	appos->newspaper->amod	0.98	0.79	0.90	0.86
appos->ambassador->amod	$\Rightarrow$	appos->diplomat->amod	0.31	0.05	0.93	0.84
average over 36 high-quality Wordnet rules			0.95	0.28	0.74	0.70

Table 3: Average of $\sigma(\boldsymbol{r_q}^\top \boldsymbol{t})$ over all inferred facts $\langle r_q, t_p \rangle$ for tuples $\boldsymbol{t_p}$ from training items for relation r_p, and vice versa, for Wordnet implications $\boldsymbol{r_p} \Rightarrow \boldsymbol{r_q}$, and model FSL (injected rules) vs. model FS (no rules).

tween relations, but not the asymmetric character of implications. For example, purely based on the training data, it appears to be more likely that the `parent` relation implies the `father` relation, than vice versa. This again demonstrates the importance and added value of injecting external rules capturing commonsense knowledge.

6 Conclusions

We presented a novel, fast approach for incorporating first-order implication rules into distributed representations of relations. We termed our approach 'lifted rule injection', as it avoids the costly grounding of first-order implication rules and is thus independent of the size of the domain of entities. By construction, these rules are satisfied for any observed or unobserved fact. The presented approach requires a restriction on the entity-tuple embedding space. However, experiments on a real-world dataset show that this does not impair the expressiveness of the learned representations. On the contrary, it appears to have a beneficial regularization effect.

By incorporating rules generated from WordNet hypernyms, our model improved over a matrix factorization baseline for knowledge base completion. Especially for domains where annotation is costly and only small amounts of training facts are available, our approach provides a way to leverage external knowledge sources for inferring facts.

In future work, we want to extend the proposed ideas beyond implications towards general first-order logic rules. We believe that supporting conjunctions, disjunctions and negations would enable to debug and improve representation learning based knowledge base completion. Furthermore, we want to integrate these ideas into neural methods beyond matrix factorization approaches.

Acknowledgments

This work was supported by the Research Foundation - Flanders (FWO), Ghent University - iMinds, Microsoft Research through its PhD Scholarship Programme, an Allen Distinguished Investigator Award, and a Marie Curie Career Integration Award.

References

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*.

Rodrigo De Salvo Braz. 2007. *Lifted First-order Probabilistic Inference*. Ph.D. thesis, Champaign, IL, USA. AAI3290183.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of

knowledge bases for relation extraction. In *EMNLP*, pages 1568–1579.

William. W. Cohen. 2016. TensorLog: A Differentiable Deductive Database. *ArXiv e-prints*, May.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Regularizing relation representations by first-order implications. In *NAACL Workshop on Automated Knowledge Base Construction (AKBC)*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

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

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

German Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*.

David Poole. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 985–991, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 74–84.

Tim Rocktäschel and Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In *NAACL Workshop on Automated Knowledge Base Construction (AKBC)*.

Tim Rocktäschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *ACL Workshop on Semantic Parsing*.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. *arXiv preprint*, abs/1511.06361.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *NAACL Workshop on Automated Knowledge Base Construction (AKBC)*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 886–896. ACL.

William Yang Wang and William W. Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, New York, NY, July. AAAI.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2014. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1199–1208. ACM.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Pro-*

ceedings of the 24th International Conference on Artificial Intelligence (IJCAI), pages 1859–1865. AAAI Press.

Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1331–1340. ACM.

Fei Wu, Jun Song, Yi Yang, Xi Li, Zhongfei Zhang, and Yueting Zhuang. 2015. Structured embedding via pairwise relations and long-range interactions in knowledge base. In *AAAI Conference on Artificial Intelligence*, pages 1663–1670.

Key-Value Memory Networks for Directly Reading Documents

Alexander H. Miller[1] **Adam Fisch**[1] **Jesse Dodge**[1,2] **Amir-Hossein Karimi**[1]
Antoine Bordes[1] **Jason Weston**[1]
[1]Facebook AI Research, 770 Broadway, New York, NY, USA
[2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
{ahm,afisch,jessedodge,ahkarimi,abordes,jase}@fb.com

Abstract

Directly reading documents and being able to answer questions from them is an unsolved challenge. To avoid its inherent difficulty, question answering (QA) has been directed towards using Knowledge Bases (KBs) instead, which has proven effective. Unfortunately KBs often suffer from being too restrictive, as the schema cannot support certain types of answers, and too sparse, e.g. Wikipedia contains much more information than Freebase. In this work we introduce a new method, Key-Value Memory Networks, that makes reading documents more viable by utilizing different encodings in the addressing and output stages of the memory read operation. To compare using KBs, information extraction or Wikipedia documents directly in a single framework we construct an analysis tool, WIKIMOVIES, a QA dataset that contains raw text alongside a preprocessed KB, in the domain of movies. Our method reduces the gap between all three settings. It also achieves state-of-the-art results on the existing WIKIQA benchmark.

1 Introduction

Question answering (QA) has been a long standing research problem in natural language processing, with the first systems attempting to answer questions by directly reading documents (Voorhees and Tice, 2000). The development of large-scale Knowledge Bases (KBs) such as Freebase (Bollacker *et al.*, 2008) helped organize information into structured forms, prompting recent progress to focus on answering questions by converting them into logical forms that can be used to query such databases (Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Fader *et al.*, 2014).

Unfortunately, KBs have intrinsic limitations such as their inevitable incompleteness and fixed schemas that cannot support all varieties of answers. Since information extraction (IE) (Craven *et al.*, 2000), intended to fill in missing information in KBs, is neither accurate nor reliable enough, collections of raw textual resources and documents such as Wikipedia will always contain more information. As a result, even if KBs can be satisfactory for closed-domain problems, they are unlikely to scale up to answer general questions on any topic. Starting from this observation, in this work we study the problem of answering by directly reading documents.

Retrieving answers directly from text is harder than from KBs because information is far less structured, is indirectly and ambiguously expressed, and is usually scattered across multiple documents. This explains why using a satisfactory KB—typically only available in closed domains—is preferred over raw text. We postulate that before trying to provide answers that are not in KBs, document-based QA systems should first reach KB-based systems' performance in such closed domains, where clear comparison and evaluation is possible. To this end, this paper introduces WIKIMOVIES, a new analysis tool that allows for measuring the performance of QA systems when the knowledge source is switched from a KB to unstructured documents. WIKIMOVIES contains $\sim$100k questions in the movie domain, and was designed to be answerable by using either a perfect KB (based on OMDb[1]), Wikipedia pages or an imper-

[1]http://www.omdbapi.com

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1400–1409,
Austin, Texas, November 1-5, 2016.

fect KB obtained through running an engineered IE pipeline on those pages.

To bridge the gap between using a KB and reading documents directly, we still lack appropriate machine learning algorithms. In this work we propose the Key-Value Memory Network (KV-MemNN), a new neural network architecture that generalizes the original Memory Network (Sukhbaatar *et al.*, 2015) and can work with either knowledge source. The KV-MemNN performs QA by first storing facts in a key-value structured memory before reasoning on them in order to predict an answer. The memory is designed so that the model learns to use keys to address relevant memories with respect to the question, whose corresponding values are subsequently returned. This structure allows the model to encode prior knowledge for the considered task and to leverage possibly complex transforms between keys and values, while still being trained using standard backpropagation via stochastic gradient descent.

Our experiments on WIKIMOVIES indicate that, thanks to its key-value memory, the KV-MemNN consistently outperforms the original Memory Network, and reduces the gap between answering from a human-annotated KB, from an automatically extracted KB or from directly reading Wikipedia. We confirm our findings on WIKIQA (Yang *et al.*, 2015), another Wikipedia-based QA benchmark where no KB is available, where we demonstrate that KV-MemNN can reach state-of-the-art results—surpassing the most recent attention-based neural network models.

2 Related Work

Early QA systems were based on information retrieval and were designed to return snippets of text containing an answer (Voorhees and Tice, 2000; Banko *et al.*, 2002), with limitations in terms of question complexity and response coverage. The creation of large-scale KBs (Auer *et al.*, 2007; Bollacker *et al.*, 2008) have led to the development of a new class of QA methods based on semantic parsing (Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Fader *et al.*, 2014; Yih *et al.*, 2015) that can return precise answers to complicated compositional questions. Due to the sparsity of KB data, however, the main challenge shifts from finding answers to developing efficient information extraction methods to populate KBs automatically (Craven *et al.*, 2000; Carlson *et al.*, 2010)—not an easy problem.

For this reason, recent initiatives are returning to the original setting of directly answering from text using datasets like TRECQA (Wang *et al.*, 2007), which is based on classical TREC resources (Voorhees *et al.*, 1999), and WIKIQA (Yang *et al.*, 2015), which is extracted from Wikipedia. Both benchmarks are organized around the task of answer sentence selection, where a system must identify the sentence containing the correct answer in a collection of documents, but need not return the actual answer as a KB-based system would do. Unfortunately, these datasets are very small (hundreds of examples) and, because of their answer selection setting, do not offer the option to directly compare answering from a KB against answering from pure text. Using similar resources as the dialog dataset of Dodge *et al.* (2016), our new benchmark WIKIMOVIES addresses both deficiencies by providing a substantial corpus of question-answer pairs that can be answered by either using a KB or a corresponding set of documents.

Even though standard pipeline QA systems like AskMR (Banko *et al.*, 2002) have been recently revisited (Tsai *et al.*, 2015), the best published results on TRECQA and WIKIQA have been obtained by either convolutional neural networks (Santos *et al.*, 2016; Yin and Schütze, 2015; Wang *et al.*, 2016) or recurrent neural networks (Miao *et al.*, 2015)—both usually with attention mechanisms inspired by (Bahdanau *et al.*, 2015). In this work, we introduce KV-MemNNs, a Memory Network model that operates a symbolic memory structured as $(key, value)$ pairs. Such structured memory is not employed in any existing attention-based neural network architecture for QA. As we will show, it gives the model greater flexibility for encoding knowledge sources and helps shrink the gap between directly reading documents and answering from a KB.

3 Key-Value Memory Networks

The Key-Value Memory Network model is based on the Memory Network (MemNNs) model (Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015) which has proven useful for a variety of document reading and question answering tasks: for reading children's books and answering questions about them (Hill *et al.*, 2016), for complex reasoning over sim-

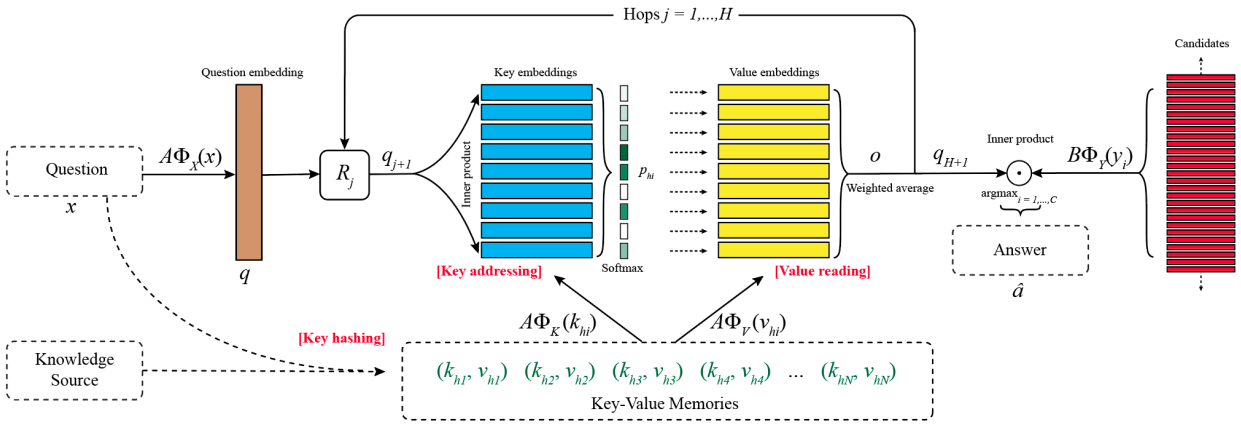


Figure 1: The Key-Value Memory Network model for question answering. See Section 3 for details.

ulated stories (Weston *et al.*, 2016) and for utilizing KBs to answer questions (Bordes *et al.*, 2015).

Key-value paired memories are a generalization of the way context (e.g. knowledge bases or documents to be read) are stored in memory. The lookup (addressing) stage is based on the key memory while the reading stage (giving the returned result) uses the value memory. This gives both (i) greater flexibility for the practitioner to encode prior knowledge about their task; and (ii) more effective power in the model via nontrivial transforms between key and value. The key should be designed with features to help match it to the question, while the value should be designed with features to help match it to the response (answer). An important property of the model is that the entire model can be trained with key-value transforms while still using standard backpropagation via stochastic gradient descent.

3.1 Model Description

Our model is based on the end-to-end Memory Network architecture of Sukhbaatar *et al.* (2015). A high-level view of both models is as follows: one defines a memory, which is a possibly very large array of slots which can encode both long-term and short-term context. At test time one is given a query (e.g. the question in QA tasks), which is used to iteratively address and read from the memory (these iterations are also referred to as "hops") looking for relevant information to answer the question. At each step, the collected information from the memory is cumulatively added to the original query to build context for the next round. At the last iteration, the final retrieved context and the most recent query are combined as features to predict a response from a list of candidates.

Figure 1 illustrates the KV-MemNN model architecture.

In KV-MemNNs we define the memory slots as pairs of vectors $(k_1, v_1) \ldots, (k_M, v_M)$ and denote the question x. The addressing and reading of the memory involves three steps:

- **Key Hashing:** the question can be used to preselect a small subset of the possibly large array. This is done using an inverted index that finds a subset $(k_{h_1}, v_{h_1}), \ldots, (k_{h_N}, v_{h_N})$ of memories of size N where the key shares at least one word with the question with frequency $< F = 1000$ (to ignore stop words), following Dodge *et al.* (2016). More sophisticated retrieval schemes could be used here, see e.g. Manning *et al.* (2008),

- **Key Addressing:** during addressing, each candidate memory is assigned a relevance probability by comparing the question to each key:

$$p_{h_i} = \text{Softmax}(A\Phi_X(x) \cdot A\Phi_K(k_{h_i}))$$

  where $\Phi_\cdot$ are feature maps of dimension D, A is a $d \times D$ matrix and $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$. We discuss choices of feature map in Sec. 3.2.

- **Value Reading:** in the final reading step, the values of the memories are read by taking their weighted sum using the addressing probabilities,

and the vector o is returned:

$$o = \sum_i p_{h_i} A\Phi_V(v_{h_i}) \ .$$

The memory access process is conducted by the "controller" neural network using $q = A\Phi_X(x)$ as the query. After receiving the result o, the query is updated with $q_2 = R_1(q + o)$ where R is a $d \times d$ matrix. The memory access is then repeated (specifically, only the addressing and reading steps, but not the hashing), using a different matrix R_j on each hop, j. The key addressing equation is transformed accordingly to use the updated query:

$$p_{h_i} = \text{Softmax}(q_{j+1}^\top A\Phi_K(k_{h_i})) \ .$$

The motivation for this is that new evidence can be combined into the query to focus on and retrieve more pertinent information in subsequent accesses. Finally, after a fixed number H hops, the resulting state of the controller is used to compute a final prediction over the possible outputs:

$$\hat{a} = \text{argmax}_{i=1,\ldots,C}\text{Softmax}(q_{H+1}^\top B\Phi_Y(y_i))$$

where y_i are the possible candidate outputs, e.g. all the entities in the KB, or all possible candidate answer sentences in the case of a dataset like WIKIQA (see Sec. 5.2). The $d \times D$ matrix B can also be constrained to be identical to A. The whole network is trained end-to-end, and the model learns to perform the iterative accesses to output the desired target a by minimizing a standard cross-entropy loss between $\hat{a}$ and the correct answer a. Backpropagation and stochastic gradient descent are thus used to learn the matrices A, B and $R_1, \ldots, R_H$.

To obtain the standard End-To-End Memory Network of Sukhbaatar *et al.* (2015) one can simply set the key and value to be the same for all memories. Hashing was not used in that paper, but is important for computational efficiency for large memory sizes, as already shown in Dodge *et al.* (2016). We will now go on to describe specific applications of key-value memories for the task of reading KBs or documents.

3.2 Key-Value Memories

There are a variety of ways to employ key-value memories that can have important effects on overall performance. The ability to encode prior knowledge in this way is an important component of KV-MemNNs, and we are free to define Φ_X, Φ_Y, Φ_K and Φ_V for the query, answer, keys and values respectively. We now describe several possible variants of Φ_K and Φ_V that we tried in our experiments, for simplicity we kept Φ_X and Φ_Y fixed as bag-of-words representations.

KB Triple Knowledge base entries have a structure of triple "subject *relation* object" (see Table 1 for examples). The representation we consider is simple: the key is composed of the left-hand side entity (subject) and the relation, and the value is the right-hand side entity (object). We double the KB and consider the reversed relation as well (e.g. we now have two triples "Blade Runner *directed_by* Ridley Scott" and "Ridley Scott *!directed_by* Blade Runner" where *!directed_by* is a different entry in the dictionary than *directed_by*). Having the entry both ways round is important for answering different kinds of questions ("Who directed Blade Runner?" vs. "What did Ridley Scott direct?"). For a standard MemNN that does not have key-value pairs the whole triple has to be encoded into the same memory slot.

Sentence Level For representing a document, one can split it up into sentences, with each memory slot encoding one sentence. Both the key and the value encode the entire sentence as a bag-of-words. As the key and value are the same in this case, this is identical to a standard MemNN and this approach has been used in several papers (Weston *et al.*, 2016; Dodge *et al.*, 2016).

Window Level Documents are split up into windows of W words; in our tasks we only include windows where the center word is an entity. Windows are represented using bag-of-words. Window representations for MemNNs have been shown to work well previously (Hill *et al.*, 2016). However, in Key-Value MemNNs we encode the key as the entire window, and the value as only the center word, which is not possible in the MemNN architecture. This makes sense because the entire window is more likely to be pertinent as a match for the question (as the key), whereas the entity at the center is more pertinent as a match for the answer (as the value). We will compare these approaches in our experiments.

Window + Center Encoding Instead of representing the window as a pure bag-of-words, thus mixing

the window center with the rest of the window, we can also encode them with different features. Here, we double the size, D, of the dictionary and encode the center of the window and the value using the second dictionary. This should help the model pick out the relevance of the window center (more related to the answer) as compared to the words either side of it (more related to the question).

Window + Title The title of a document is commonly the answer to a question that relates to the text it contains. For example "What did Harrison Ford star in?" can be (partially) answered by the Wikipedia document with the title "Blade Runner". For this reason, we also consider a representation where the key is the word window as before, but the value is the document title. We also keep all the standard (window, center) key-value pairs from the window-level representation as well, thus doubling the number of memory slots in comparison. To differentiate the two keys with different values we add an extra feature "_window_" or "_title_" to the key, depending on the value. The "_title_" version also includes the actual movie title in the key. This representation can be combined with center encoding. Note that this representation is inherently specific to datasets in which there is an apparent or meaningful title for each document.

4 The WikiMovies Benchmark

The WIKIMOVIES benchmark consists of question-answer pairs in the domain of movies. It was built with the following goals in mind: (i) machine learning techniques should have ample training examples for learning; and (ii) one can analyze easily the performance of different representations of knowledge and break down the results by question type. The dataset can be downloaded from `http://fb.ai/babi`.

4.1 Knowledge Representations

We construct three forms of knowledge representation: (i) Doc: raw Wikipedia documents consisting of the pages of the movies mentioned; (ii) KB: a classical graph-based KB consisting of entities and relations created from the Open Movie Database (OMDb) and MovieLens; and (iii) IE: information extraction performed on the Wikipedia pages to build a KB in a similar form as (ii). We take care to construct QA pairs such that they are all potentially answerable from either the KB from (ii) or the original Wikipedia documents from (i) to eliminate data sparsity issues. However, it should be noted that the advantage of working from raw documents in real applications is that data sparsity is less of a concern than for a KB, while on the other hand the KB has the information already parsed in a form amenable to manipulation by machines. This dataset can help analyze what methods we need to close the gap between all three settings, and in particular what are the best methods for reading documents when a KB is not available. A sample of the dataset is shown in Table 1.

Doc: Wikipedia Article for Blade Runner (partially shown)
Blade Runner is a 1982 American neo-noir dystopian science fiction film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, Sean Young, and Edward James Olmos. The screenplay, written by Hampton Fancher and David Peoples, is a modified film adaptation of the 1968 novel "Do Androids Dream of Electric Sheep?" by Philip K. Dick. The film depicts a dystopian Los Angeles in November 2019 in which genetically engineered replicants, which are visually indistinguishable from adult humans, are manufactured by the powerful Tyrell Corporation as well as by other "mega-corporations" around the world. Their use on Earth is banned and replicants are exclusively used for dangerous, menial, or leisure work on off-world colonies. Replicants who defy the ban and return to Earth are hunted down and "retired" by special police operatives known as "Blade Runners". . . .
KB entries for Blade Runner (subset)
Blade Runner *directed_by* Ridley Scott Blade Runner *written_by* Philip K. Dick, Hampton Fancher Blade Runner *starred_actors* Harrison Ford, Sean Young, . . . Blade Runner *release_year* 1982 Blade Runner *has_tags* dystopian, noir, police, androids, . . .
IE entries for Blade Runner (subset)
Blade Runner, Ridley Scott *directed* dystopian, science fiction, film Hampton Fancher *written* Blade Runner Blade Runner *starred* Harrison Ford, Rutger Hauer, Sean Young. . . Blade Runner *labelled* 1982 neo noir special police, Blade *retired* Blade Runner Blade Runner, special police *known* Blade
Questions for Blade Runner (subset)
Ridley Scott directed which films? What year was the movie Blade Runner released? Who is the writer of the film Blade Runner? Which films can be described by dystopian? Which movies was Philip K. Dick the writer of? Can you describe movie Blade Runner in a few words?

Table 1: WIKIMOVIES: Questions, Doc, KB and IE sources.

Doc We selected a set of Wikipedia articles about movies by identifying a set of movies from OMDb[2] that had an associated article by title match. We keep the title and the first section (before the contents box) for each article. This gives ∼17k documents (movies) which comprise the set of documents our models will read from in order to answer questions.

[2] `http://beforethecode.com/projects/omdb/download.aspx`

KB Our set of movies were also matched to the MovieLens dataset[3]. We built a KB using OMDb and MovieLens metadata with entries for each movie and nine different relation types: director, writer, actor, release year, language, genre, tags, IMDb rating and IMDb votes, with ∼10k related actors, ∼6k directors and ∼43k entities in total. The KB is stored as triples; see Table 1 for examples. IMDb ratings and votes are originally real-valued but are binned and converted to text ("unheard of", "unknown", "well known", "highly watched", "famous"). We finally only retain KB triples where the entities also appear in the Wikipedia articles[4] to try to guarantee that all QA pairs will be equally answerable by either the KB or Wikipedia document sources.

IE As an alternative to directly reading documents, we explore leveraging information extraction techniques to transform documents into a KB format. An IE-KB representation has attractive properties such as more precise and compact expressions of facts and logical key-value pairings based on subject-verb-object groupings. This can come at the cost of lower recall due to malformed or completely missing triplets. For IE we use standard open-source software followed by some task-specific engineering to improve the results. We first employ coreference resolution via the Stanford NLP Toolkit (Manning *et al.*, 2014) to reduce ambiguity by replacing pronominal ("he", "it") and nominal ("the film") references with their representative entities. Next we use the SENNA semantic role labeling tool (Collobert *et al.*, 2011) to uncover the grammatical structure of each sentence and pair verbs with their arguments. Each triplet is cleaned of words that are not recognized entities, and lemmatization is done to collapse different inflections of important task-specific verbs to one form (e.g. stars, starring, star → starred). Finally, we append the movie title to each triple similar to the "Window + Title" representation of Sec. 3.2, which improved results.

4.2 Question-Answer Pairs

Within the dataset's more than 100,000 question-answer pairs, we distinguish 13 classes of question corresponding to different kinds of edges in our KB. They range in scope from specific—such as *actor to movie*: "What movies did Harrison Ford star in?" and *movie to actors*: "Who starred in Blade Runner?"—to more general, such as *tag to movie*: "Which films can be described by *dystopian*?"; see Table 4 for the full list. For some question there can be multiple correct answers.

Using SimpleQuestions (Bordes *et al.*, 2015), an existing open-domain question answering dataset based on Freebase, we identified the subset of questions posed by human annotators that covered our question types. We created our question set by substituting the entities in those questions with entities from all of our KB triples. For example, if the original question written by an annotator was "What movies did Harrison Ford star in?", we created a pattern "What movies did [@actor] star in?", which we substitute for any other actors in our set, and repeat this for all annotations. We split the questions into disjoint training, development and test sets with ∼96k, 10k and 10k examples, respectively. The same question (even worded differently) cannot appear in both train and test sets. Note that this is much larger than most existing datasets; for example, the WIKIQA dataset (Yang *et al.*, 2015) for which we also conduct experiments in Sec. 5.2 has only ∼1000 training pairs.

Method	KB	IE	Doc
(Bordes *et al.*, 2014) QA system	93.5	56.5	N/A
Supervised Embeddings	54.4	54.4	54.4
Memory Network	78.5	63.4	69.9
Key-Value Memory Network	**93.9**	**68.3**	**76.2**

Table 2: Test results (% hits@1) on WIKIMOVIES, comparing human-annotated KB (KB), information extraction-based KB (IE), and directly reading Wikipedia documents (Doc).

Memory Representation	Doc
Sentence-level	52.4
Window-level	66.8
Window-level + Title	74.1
Window-level + Center Encoding + Title	**76.9**

Table 3: Development set performance (% hits@1) with different document memory representations for KV-MemNNs.

[3] http://grouplens.org/datasets/movielens/

[4] The dataset also includes the slightly larger version without this constraint.

5 Experiments

This section describes our experiments on WIKIMOVIES and WIKIQA.

5.1 WikiMovies

We conducted experiments on the WIKIMOVIES dataset described in Sec. 4. Our main goal is to compare the performance of KB, IE and Wikipedia (Doc) sources when trying varying learning methods. We compare four approaches: (i) the QA system of Bordes *et al.* (2014) that performs well on existing datasets WebQuestions (Berant *et al.*, 2013) and SimpleQuestions (Bordes *et al.*, 2015) that use KBs only; (ii) supervised embeddings that do not make use of a KB at all but learn question-to-answer embeddings directly and hence act as a sanity check (Dodge *et al.*, 2016); (iii) Memory Networks; and (iv) Key-Value Memory Networks. Performance is reported using the accuracy of the top hit (single answer) over all possible answers (all entities), i.e. the hits@1 metric measured in percent. In all cases hyperparameters are optimized on the development set, including the memory representations of Sec. 3.2 for MemNNs and KV-MemNNs. As MemNNs do not support key-value pairs, we concatenate key and value together when they differ instead.

The main results are given in Table 2. The QA system of Bordes *et al.* (2014) outperforms Supervised Embeddings and Memory Networks for KB and IE-based KB representations, but is designed to work with a KB, not with documents (hence the N/A in that column). However, Key-Value Memory Networks outperform all other methods on all three data source types. Reading from Wikipedia documents directly (Doc) outperforms an IE-based KB (IE), which is an encouraging result towards automated machine reading though a gap to a human-annotated KB still remains (93.9 vs. 76.2). The best memory representation for directly reading documents uses "Window-level + Center Encoding + Title" ($W = 7$ and $H = 2$); see Table 3 for a comparison of results for different representation types. Both center encoding and title features help the window-level representation, while sentence-level is inferior.

QA Breakdown A breakdown by question type comparing the different data sources for KV-MemNNs is given in Table 4. IE loses out especially to Doc (and KB) on Writer, Director and Actor to Movie, perhaps because coreference is difficult in these cases – although it has other losses elsewhere too. Note that only 56% of subject-object pairs in IE match the triples in the original KB, so losses are expected. Doc loses out to KB particularly on Tag to Movie, Movie to Tags, Movie to Writer and Movie to Actors. Tag questions are hard because they can reference more or less any word in the entire Wikipedia document; see Table 1. Movie to Writer/Actor are hard because there is likely only one or a few references to the answer across all documents, whereas for Writer/Actor to Movie there are more possible answers to find.

Question Type	KB	IE	Doc
Writer to Movie	97	72	91
Tag to Movie	85	35	49
Movie to Year	95	75	89
Movie to Writer	95	61	64
Movie to Tags	94	47	48
Movie to Language	96	62	84
Movie to IMDb Votes	92	92	92
Movie to IMDb Rating	94	75	92
Movie to Genre	97	84	86
Movie to Director	93	76	79
Movie to Actors	91	64	64
Director to Movie	90	78	91
Actor to Movie	93	66	83

Table 4: Breakdown of test results (% hits@1) on WIKIMOVIES for Key-Value Memory Networks using different knowledge representations.

Knowledge Representation	KV-MemNN
KB	93.9
One Template Sentence	82.9
All Templates Sentences	80.0
One Template + Coreference	76.0
One Template + Conjunctions	74.0
All Templates + Conj. + Coref.	72.5
Wikipedia Documents	76.2

Table 5: Analysis of test set results (% hits@1) for KB vs. Synthetic Docs on WIKIMOVIES.

KB vs. Synthetic Document Analysis To further understand the difference between using a KB versus reading documents directly, we conducted an experiment where we constructed synthetic documents using the KB. For a given movie, we use a simple grammar to construct a synthetic "Wikipedia" doc-

Method	MAP	MRR
Word Cnt	0.4891	0.4924
Wgt Word Cnt	0.5099	0.5132
2-gram CNN (Yang *et al.*, 2015)	0.6520	0.6652
AP-CNN (Santos *et al.*, 2016)	0.6886	0.6957
Attentive LSTM (Miao *et al.*, 2015)	0.6886	0.7069
Attentive CNN (Yin and Schütze, 2015)	0.6921	0.7108
L.D.C. (Wang *et al.*, 2016)	0.7058	0.7226
Memory Network	0.5170	0.5236
Key-Value Memory Network	**0.7069**	**0.7265**

Table 6: Test results on WikiQA.

ument based on the KB triples: for each relation type we have a set of template phrases (100 in total) used to generate the fact, e.g. "Blade Runner came out in 1982" for the entry BLADE RUNNER RELEASE_YEAR 1982. We can then parameterize the complexity of our synthetic documents: (i) using one template, or all of them; (ii) using conjunctions to combine facts into single sentences or not; and (iii) using coreference between sentences where we replace the movie name with "it".[5] The purpose of this experiment is to find which aspects are responsible for the gap in performance to a KB. The results are given in Table 5. They indicate that some of the loss (93.9% for KB to 82.9% for One Template Sentence) in performance is due directly to representing in sentence form, making the subject, relation and object harder to extract. Moving to a larger number of templates does not deteriorate performance much (80%). The remaining performance drop seems to be split roughly equally between conjunctions (74%) and coreference (76%). The hardest synthetic dataset combines these (All Templates + Conj. + Coref.) and is actually harder than using the real Wikipedia documents (72.5% vs. 76.2%). This is possibly because the amount of conjunctions and coreferences we make are artificially too high (50% and 80% of the time, respectively).

5.2 WikiQA

WIKIQA (Yang *et al.*, 2015) is an existing dataset for answer sentence selection using Wikipedia as the knowledge source. The task is, given a question, to select the sentence coming from a Wikipedia document that best answers the question, where performance is measured using mean average precision (MAP) and mean reciprocal rank (MRR) of the ranked set of answers. The dataset uses a pre-built information retrieval step and hence provides a fixed set of candidate sentences per question, so systems do not have to consider ranking all of Wikipedia. In contrast to WIKIMOVIES, the training set size is small ($\sim$1000 examples) while the topic is much more broad (all of Wikipedia, rather than just movies) and the questions can only be answered by reading the documents, so no comparison to the use of KBs can be performed. However, a wide range of methods have already been tried on WIKIQA, thus providing a useful benchmark to test if the same results found on WIKIMOVIES carry across to WIKIQA, in particular the performance of Key-Value Memory Networks.

Due to the size of the training set, following many other works (Yang *et al.*, 2015; Santos *et al.*, 2016; Miao *et al.*, 2015) we pre-trained the word vectors (matrices A and B which are constrained to be identical) before training KV-MemNNs. We employed Supervised Embeddings (Dodge *et al.*, 2016) for that goal, training on all of Wikipedia while treating the input as a random sentence and the target as the subsequent sentence. We then trained KV-MemNNs with dropout regularization: we sample words from the question, memory representations and the answers, choosing the dropout rate using the development set. Finally, again following other successful methods (Yin and Schütze, 2015), we combine our approach with exact matching word features between question and answers. Key hashing was not used as candidates were already pre-selected. To represent the memories, we used the Window-Level representation (the best choice on the dev set was $W = 7$) as the key and the whole sentence as the value, as the value should match the answer which in this case is a sentence. Additionally, in the representation all numbers in the text and the phrase "how many" in the question were replaced with the feature "_number_". The best choice of hops was also $H = 2$ for KV-MemNNs.

The results are given in Table 6. Key-Value Memory Networks outperform a large set of other methods, although the results of the L.D.C. method of (Wang *et al.*, 2016) are very similar. Memory Networks, which cannot easily pair windows to sentences, perform much worse, highlighting the importance of key-value memories.

[5] This data is also part of the WIKIMOVIES benchmark.

6 Conclusion

We studied the problem of directly reading documents in order to answer questions, concentrating our analysis on the gap between such direct methods and using human-annotated or automatically constructed KBs. We presented a new model, Key-Value Memory Networks, which helps bridge this gap, outperforming several other methods across two datasets, WIKIMOVIES and WIKIQA. However, some gap in performance still remains. WIKIMOVIES serves as an analysis tool to shed some light on the causes. Future work should try to close this gap further.

Key-Value Memory Networks are versatile models for reading documents or KBs and answering questions about them—allowing to encode prior knowledge about the task at hand in the key and value memories. These models could be applied to storing and reading memories for other tasks as well, and future work should try them in other domains, such as in a full dialog setting.

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *Semantic Web Conference, 2007*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR, 2015*.

Banko, M., Brill, E., Dumais, S., and Lin, J. (2002). Askmsr: Question answering using the worldwide web. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002*.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP, 2013*.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data, 2008*.

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *EMNLP, 2014*.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence, 2010*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, **12**, 2493–2537.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial intelligence*, **118**, 69–113.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. (2016). Evaluating prerequisite qualities for learning end-to-end dialog systems. In *ICLR, 2016*.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. In *KDD, 2014*.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2016). The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR, 2016*.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP, 2013*.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL: System Demonstrations, 2014*.

Miao, Y., Yu, L., and Blunsom, P. (2015). Neural variational inference for text processing. *arXiv preprint arXiv:1511.06038*.

Santos, C. d., Tan, M., Xiang, B., and Zhou, B. (2016). Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *NIPS, 2015*.

Tsai, C., Yih, W.-t., and Burges, C. (2015). Web-based question answering: Revisiting askmsr. Technical report, Technical Report MSR-TR-2015-20, Microsoft Research.

Voorhees, E. M. *et al.* (1999). The trec-8 question answering track report. In *Trec, 1999*.

Voorhees, E. M. and Tice, D. M. (2000). Building a question answering test collection. In *ACM SIGIR Conference on Research and Development in Information Retrieval, 2000*.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL, 2007*.

Wang, Z., Mi, H., and Ittycheriah, A. (2016). Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *ICLR, 2015*.

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2016). Towards ai-complete question answering: a set of prerequisite toy tasks. In *ICLR, 2016*.

Yang, Y., Yih, W.-t., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP, 2015*.

Yih, W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL, 2015*.

Yin, W. and Schütze, H. (2015). Convolutional neural network for paraphrase identification. In *NACL: Human Language Technologies, 2015*.

Analyzing Framing through the Casts of Characters in the News

Dallas Card[1] **Justin H. Gross**[2] **Amber E. Boydstun**[3] **Noah A. Smith**[4]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]Department of Political Science, University of Massachusetts, Amherst, MA 01003, USA
[3]Department of Political Science, University of California, Davis, CA 95616, USA
[4]Computer Science & Engineering, University of Washington, WA 98195, USA
dcard@cmu.edu aboydstun@ucdavis.edu jhgross@polsci.umass.edu
nasmith@cs.washington.edu

Abstract

We present an unsupervised model for the discovery and clustering of latent "personas" (characterizations of entities). Our model simultaneously clusters documents featuring similar collections of personas. We evaluate this model on a collection of news articles about immigration, showing that personas help predict the coarse-grained framing annotations in the Media Frames Corpus. We also introduce automated model selection as a fair and robust form of feature evaluation.

1 Introduction

Social science tells us that communication almost inescapably involves **framing**—choosing "a few elements of perceived reality and assembling a narrative that highlights connections among them to promote a particular interpretation" (Entman, 2007). Memorable examples include loaded phrases (*death tax*, *war on terror*), but the literature attests a much wider range of linguistic means toward this end (Pan and Kosicki, 1993; Greene and Resnik, 2009; Choi et al., 2012; Baumer et al., 2015).

Framing is associated with several phenomena to which NLP has been applied, including *ideology* (Lin et al., 2006; Hardisty et al., 2010; Iyyer et al., 2014), *sentiment* (Pang and Lee, 2008; Feldman, 2013), and *stance* (Walker et al., 2012; Hasan and Ng, 2013). Although such author attributes are interesting, framing scholarship is concerned with persistent patterns of representation of particular issues—without necessarily tying these to the states or intentions of authors—and the effects that such patterns may have on public opinion and policy. We also note that NLP has often been used in large-scale studies of news and its relation to other social phenomena (Leskovec et al., 2009; Gentzkow and Shapiro, 2010; Smith et al., 2013; Niculae et al., 2015).

Can framing be automatically recognized? If so, social-scientific studies of framing will be enabled by new *measurements*, and new applications might bring framing effects to the consciousness of everyday readers. Several recent studies have begun to explore unsupervised framing analysis of political text using autoregressive and hierarchical topic models (Nguyen et al., 2013; Nguyen et al., 2015; Tsur et al., 2015), but most of these conceptualize framing along a single dimension. Rather than trying to place individual articles on a continuum from liberal to conservative or positive to negative, we are interested in discovering broad-based patterns in the ways in which the media communicate about issues.

Here, our focus is on the narratives found in news stories, specifically the participants in those stories. Insofar as journalists make use of archetypal narratives (e.g., the struggle of an individual against a more powerful adversary), we expect to see recurring representations of characters in these narratives (Schneider and Ingram, 1993; Van Gorp, 2010). A classic example is the contrast between "worthy" and "unworthy" victims (Herman and Chomsky, 1988). More recently, Glenn Greenwald has pointed out how he was repeatedly characterized as an *activist* or *blogger*, rather than a *journalist* during his reporting on the NSA (Greenwald, 2014).

Our model builds on the "Dirichlet persona model" (DPM) introduced by Bamman et al. (2013)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1410–1420,
Austin, Texas, November 1-5, 2016.

for the unsupervised discovery of what they called "personas" in short film summaries (e.g., the "dark hero"). As in the DPM, we operationalize personas as mixture of textually-expressed characteristics: what they do, what is done to them, and their descriptive attributes. We begin by providing a description of our full model, after which we highlight the differences from the DPM.

This paper's main contributions are:

- We strengthen the DPM's assumptions about the *combinations* of personas found in documents, applying a Dirichlet process prior to infer patterns of coocurrence (§3). The result is a clustering of documents based on the collections of personas they use, discovered simultaneously with those personas.
- Going beyond named characters, we allow Bamman-style personas to account for entities like institutions, objects, and concepts (§5).
- We find that our model produces interpretable clusters that provide insight into our corpus of immigration news articles (§6).
- We propose a new kind of evaluation based on Bayesian optimization. Given a supervised learning problem, we treat the inclusion of a candidate feature set (here, personas) as a hyperparameter to be optimized alongside other hyperparameters (§7).
- In the case of U.S. news stories about immigration, we find that personas are, in many cases, helpful for automatically inferring the coarse-grained framing and tone employed in a piece of text, as defined in the Media Frames Corpus (Card et al., 2015) (§7).

2 Model Description

The plate diagram for the new model is shown in Figure 1 (right), with the original DPM (Bamman et al., 2013) shown on the left.

As evidence, the model considers tuples $\langle w, r, e, i\rangle$, where w is a word token and r is the category of syntactic relation[1] it bears to an entity with index e mentioned in document with index i. The model's generative story explains this evidence

[1]We adopt the terminology from Bamman et al. (2013) of "agent", "patient", and "attribute", even though these categories of relations are defined in terms of syntactic dependences.

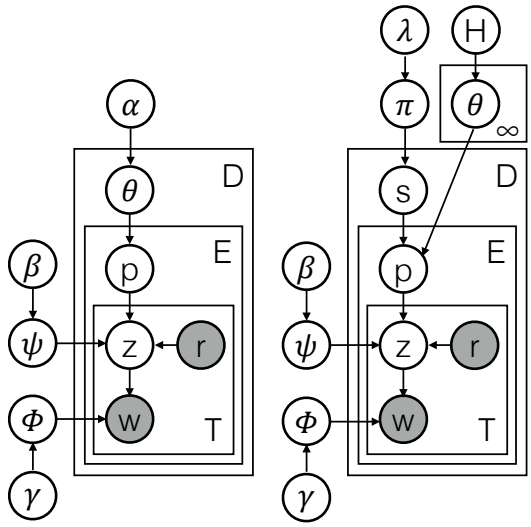


Figure 1: Plate diagrams for the DPM (left), and for the new model (right).

as follows:

1. Let there be K topics as in LDA (Blei et al., 2003). Each topic $\phi_k \sim \text{Dir}(\gamma)$ is a multinomial over the V words in the vocabulary, drawn from a Dirichlet parameterized by γ.
2. For each of P personas p, and for each syntactic relation type r, define a multinomial $\psi_{p,r}$ over the K topics, each drawn from a Dirichlet parameterized by β.
3. Assume an infinite set of distributions over personas drawn from a base distribution H. Each of these $\theta_j \sim \text{Dir}(\alpha)$ is a multinomial over the P personas, with an associated probability of being selected π_j, drawn from the stick-breaking process with hyperparameter λ.
4. For each document i:
   (a) Draw a cluster assignment $s_i \sim \pi$, with corresponding multinomial distribution over personas θ_{s_i}.
   (b) For each entity e participating in i:
      i. Draw e's persona $p_e \sim \theta_{s_i}$.
      ii. For every $\langle r, w\rangle$ tuple associated with e in i, draw $z \sim \psi_{p_e,r}$ then $w \sim \phi_z$.

The DPM (Figure 1, left) has a similar generative story, except that each document has a unique distribution over personas. As such, step 4(a) is replaced with a draw from a symmetric Dirichlet distribution $\theta_i \sim \text{Dir}(\alpha)$.

3 Clustering Stories

The DPM assumes that each document has a unique distribution (θ_i) from which its personas are drawn. However, for entities mentioned in news articles (as well as for the dramatis personae of films), we would expect certain types of personas to occur together frequently, such as articles about lawmakers and laws. Thus we would like to cluster documents based on their "casts" of personas. To do this, we have added a Dirichlet process (DP) prior on the document-specific distribution over personas (step 3), which allows the number of clusters to adapt to the size and complexity of the corpus (Antoniak, 1974; Escobar and West, 1994).

Although the model admits an unbounded number of distributions over personas, the properties of DPs are such that the number used by D documents will tend to be much less than D. As a result, inference under this model provides topics ϕ (distributions over words) interpretable as textual descriptors of entities, personas ψ (distributions over reusable topics), and clusters of articles s with associated distributions over personas θ.

Following Bamman et al. (2013), we perform inference using collapsed Gibbs sampling, collapsing out the distributions over words (ϕ), topics (ψ), and personas (θ), as well as π. On each iteration, we first sample a cluster for each document, followed by a persona for each entity, followed by a topic for each tuple. Because we assume a conjugate base measure, sampling clusters can be done efficiently using the Chinese restaurant process (Aldous, 1985) for story types, personas, and topics, with slice sampling for hyperparameters (α, β, γ, λ). Because such algorithms are well known to NLP readers, we have relegated details to the supplementary material.

During sampling, we discard samples from the first 10,000 iterations, and collect one sample from every tenth iteration for following 1,000 iterations. We sample hyperparameters every 20 iterations for the first 500 iterations, and every 100 thereafter.

4 Dataset

The Media Frames Corpus (MFC; Card et al., 2015) consists of annotations for approximately 4,200 articles about immigration taken from 13 U.S. newspapers over the years 1980–2012. The annotations for these articles are in terms of a set of 15 general-purpose "framing dimensions" (such as Politics and Legality), developed to be broadly applicable to a variety of issues, and to be recognizable in text (by trained annotators). Each article has been annotated with a "primary frame" (the overall dominant aspect of immigration being emphasized), as well as an overall "tone" (pro, neutral, or anti), which is the extent to which a pro-immigration advocate would like to see the article in print, without implying any any stance taken by the author.[2] The MFC contains at least two independent annotations for each article; agreement on the primary frame and tone was established through discussion in cases of initial disagreement. A complete list of these framing dimensions is given in the supplementary material.

In order to train our model on a larger collection of articles, we use the original corpus of articles from which the annotated articles in the MFC were drawn. This produces a corpus of approximately 37,000 articles about immigration; we train the persona model on this larger dataset, only using the smaller set for evaluation on a secondary task. Note that the MFC annotations are not used by our model; rather, we hypothesize that the personas it discovers may serve as features to help predict framing—this serves as one of our evaluations (§7).

5 Identifying Entities

The original focus of the DPM was on *named* characters in movies, which could be identified using named entity recognition and pronominal coreference (Bamman et al., 2013), or name matching for pre-defined characters (Bamman et al., 2014). Here, we are interested in applying our model to entities about which we assume no specific prior knowledge.

In order to include a broader set of entities, we preprocess the corpus and apply a series of filters. First, we obtain lemmas, part-of-speech tags, dependencies, coreference resolution, and named entities from the Stanford CoreNLP pipeline (Manning et al., 2014), as well as supersense tags from the AMALGrAM tagger (Schneider and Smith, 2015). For each document, we consider all tokens with a

[2]The MFC also contains more fine-grained annotations of spans of text which cue each of the framing dimensions, but we do not make use of those here.

NN* or PRP part of speech as possible entities, partially clustered by coreference. We then merge all clusters (including singletons) within each document that share a non-pronomial mention word.

Next, we exclude all clusters lacking at least one mention classified as a person, organization, location, group, object, artifact, process, or act (by CoreNLP or AMALGrAM). From these, we extract $\langle w, r, e, i \rangle$ tuples using extraction patterns lightly adapted from (Bamman et al., 2013). (The complete set of patterns are given in the supplementary material.) To further restrict the set of entities to those that have sufficient evidence, we construct a vocabulary for each of the three relations, and exclude words that appear less than three times in the corresponding vocabulary.[3] We then apply one last filter to exclude entities that have fewer than three qualifying tuples across all mentions. From the dataset described in §4, we extract 128,655 entities, mentioned using 11,262 different mention words, with 575,910 tuples and 11,104 distinct $\langle r, w \rangle$ pairs.

6 Exploratory Analysis

Here we discuss our model, as estimated on the corpus of 37,000 articles discussed in §4 with 50 personas and 100 topics; these values were not tuned. A cursory examination of topics shows that each tends to be a group of either verbs or attributes. Personas, on the other hand, blend topics to include all three relation types. The estimated Dirichlet hyperparameters are all $\ll 1$, giving sparse (and hence easily scanned) distributions over personas, topics, and words.

Table 1 shows all 50 personas. For each p, we show (i) the mention words most strongly associated with p, and (ii) $\langle r, w \rangle$ pairs associated with the persona. (To save space, "I" denotes *immigrant*.) Recall that, like the Dirichlet persona model, our model says nothing about the mention words; they are *not* included as evidence during inference.[4] Nonetheless, each persona is strongly associated with a sparse handful of mention words, and we find that labeling each persona by its most strongly associated mention word (excluding *immigrant*) is often sensible (these are capitalized in Table 1, though in some cases the relation words differentiate strongly (e.g., the *group* personas, IDs 17 and 18 in Table 1).

The model finds expected participants (such as *workers*, political *candidates*, and *refugees*), but also more conceptual entities, such as *laws*, *bills* (IDs 3, 37), and the U.S.-Mexican *border* (ID 5), which looms large in the immigration debate. Some interesting distinctions are discovered, such as two of the *worker* personas, one high-skilled and residing legally (ID 48), the other illegal (ID 49).

Using the original publication dates of the articles, we can estimate the frequency of appearance of each persona within immigration coverage by summing the posterior distribution over personas for each entity mention, and plotting these frequencies across time. (Note that time metadata is not given to the model as evidence.) We find immediately that personas can signal events. Figure 2 shows these temporal trajectories for a small, selected set of personas. Although *bills* and *laws* are conceptually similar, and have similar trajectories from 1980 to 2005, they are strongly divergent in 2006 and 2010. These are particularly notable years for immigration policy, corresponding to the failed Comprehensive Immigration Reform Act of 2006 (Senate bill S.2611) and Arizona's controversial anti-immigration laws from 2010.[5] Refugees, by contrast, show a marked spike around the year 2000. Inspection showed this persona to be strongly tied to the case of Elián González, which received a great deal of media attention in that year.

The main advantage of the extended model over the DPM is being able to cluster articles by "casts." During sampling, thousands of clusters are created (and mostly destroyed). Ultimately, our inference procedure settled on approximately 110 clusters, and we consider two examples. Figure 3 shows the temporal trajectories of the two clusters with the greatest representation of the *refugee* persona. Both show the characteristic spike around the year 2000. The top personas for these two clusters are given in Table

[3] We also exclude the lemma "say" as a stopword, as it is the most common verb in the corpus by an order of magnitude

[4] We did explore adding mention words as evidence, but they tended to dominate the relation tuples. Because our interest is in a richer set of framing devices than simply the words used to refer to people (and other entities), we consider here only the model based on the surrounding context.

[5] Other notable events which appear to be represented include the Illegal Immigration Reform and Immigrant Responsibility Act of 1996, and the Secure Fence Act of 2006.

ID	Mention words	Relations
1	AGENT police official authority	federal$_m$ tell$_p$ find$_a$ arrest$_a$ local$_m$ tell$_a$
2	ASYLUM crime refugee asylum_seeker	political$_m$ seek$_p$ grant$_p$ commit$_p$ serious$_m$ deny$_p$
3	BILL law immigration_reform measure	comprehensive$_m$ pass$_a$ pass$_p$ make$_a$ have$_a$ support$_p$
4	BOAT van crime document	criminal$_m$ other$_m$ have$_p$ use$_a$ use$_p$ be$_a$
5	BORDER border_patrol border_agent	mexican$_m$ cross$_p$ secure$_p$ southern$_m$ u.s.-mexico$_m$ close$_p$
6	BUSH official mcnary people I	have$_a$ tell$_a$ want$_a$ tell$_p$ former$_m$ call$_a$
7	CANDIDATE bush romney leader	republican$_m$ presidential$_m$ democratic$_m$ have$_a$ call$_a$ support$_a$
8	CARD document visa status	green$_m$ new$_m$ get$_p$ temporary$_m$ fake$_m$ permanent$_m$
9	CARD visa state document	consular$_m$ federal$_m$ have$_a$ mexican$_m$ receive$_p$ get$_p$
10	COMPANY country I state nation	have$_a$ regional$_m$ global$_m$ rural$_m$ take$_a$ require$_p$
11	COUNTRY people I citizen united_states	american$_m$ other$_m$ enter$_p$ have$_a$ leave$_p$ central$_m$
12	COUPLE marriage people I class	gay$_m$ bilingual$_m$ same-sex$_m$ have$_a$ prime$_m$ seasonal$_m$
13	COURT lawsuit suit ruling	federal$_m$ file$_p$ rule$_a$ civil$_m$ file$_a$ have$_a$
14	EMPLOYER company people business	hire$_a$ have$_a$ many$_m$ require$_p$ employ$_a$ local$_m$
15	FENCE amendment law wall	real$_m$ 14th$_m$ virtual$_m$ build$_p$ be$_a$ have$_a$
16	GOVERNMENT court judge official	federal$_m$ local$_m$ have$_a$ rule$_a$ ask$_p$ other$_m$
17	GROUP deportation attack country	terrorist$_m$ civil$_m$ face$_p$ armed$_m$ islamic$_m$ muslim$_m$
18	GROUP I voter people bush	hispanic$_m$ immigrant$_m$ local$_m$ many$_m$ want$_a$ have$_a$
19	I ALIEN immigration people worker	illegal$_m$ allow$_p$ have$_a$ legal$_m$ undocumented$_m$ live$_a$
20	I ALIEN people criminal inmate	illegal$_m$ criminal$_m$ deport$_p$ immigrant$_m$ detain$_p$ release$_p$
21	I ALIEN worker immigration employer	illegal$_m$ hire$_p$ undocumented$_m$ employ$_p$ legal$_m$ hire$_a$
22	I ALIEN worker people immigration	illegal$_m$ arrest$_p$ undocumented$_m$ arrest$_a$ charge$_p$ transport$_p$
23	I CHILD worker people student	immigrant$_m$ foreign-born$_m$ have$_a$ many$_m$ come$_a$ new$_m$
24	I GROUP people population business	new$_m$ immigrant$_m$ other$_m$ many$_m$ asian$_m$ have$_a$
25	I GROUP program center city	new$_m$ have$_a$ first$_m$ be$_a$ other$_m$ make$_a$
26	I IMMIGRATION alien worker	illegal$_m$ legal$_m$ hire$_p$ have$_a$ allow$_p$ undocumented$_m$
27	I IMMIGRATION alien worker people	illegal$_m$ legal$_m$ have$_a$ be$_a$ come$_a$ immigrant$_m$
28	I JEWS refugee israel child	soviet$_m$ jewish$_m$ russian$_m$ have$_a$ vietnamese$_m$ israeli$_m$
29	I MAN alien refugee people	illegal$_m$ chinese$_m$ cuban$_m$ arrest$_p$ haitian$_m$ find$_p$
30	I PEOPLE child student worker	many$_m$ young$_m$ have$_a$ illegal$_m$ come$_a$ be$_a$
31	I PEOPLE country woman man	black$_m$ muslim$_m$ african$_m$ have$_a$ come$_a$ korean$_m$
32	I WORKER people citizen job	american$_m$ new$_m$ have$_a$ mexican$_m$ illegal$_m$ many$_m$
33	I WORKER resident student people	legal$_m$ foreign$_m$ permanent$_m$ have$_a$ allow$_p$ skilled$_m$
34	I WORKER student people child	undocumented$_m$ illegal$_m$ immigrant$_m$ have$_a$ allow$_p$ live$_a$
35	JOB I people immigration law	have$_p$ have$_a$ be$_a$ take$_p$ good$_m$ make$_a$
36	JOB study survey I labor	find$_a$ new$_m$ find$_p$ show$_a$ fill$_p$ take$_p$
37	LAW immigration_law bill measure	new$_m$ federal$_m$ enforce$_p$ require$_a$ pass$_p$ allow$_a$
38	MAN I woman people haitians	deport$_p$ have$_a$ arrest$_p$ hold$_p$ release$_p$ face$_a$
39	MAN people agent official I	arrest$_p$ charge$_p$ other$_m$ former$_m$ have$_a$ face$_a$
40	MAN woman I people girl	tell$_a$ kill$_p$ have$_a$ other$_m$ young$_m$ take$_p$
41	PEOPLE I child man woman	have$_a$ come$_a$ live$_a$ go$_a$ tell$_p$ work$_a$
42	PROFILING violence abuse discrimination	racial$_m$ domestic$_m$ safe$_m$ physical$_m$ be$_a$ affordable$_m$
43	PROGRAM system law agency	new$_m$ national$_m$ federal$_m$ create$_p$ use$_p$ special$_m$
44	REFUGEE I boy people elian	cuban$_m$ haitian$_m$ chinese$_m$ have$_a$ allow$_p$ return$_p$
45	SCHOOL people I family english	have$_a$ high$_m$ see$_a$ come$_a$ go$_a$ be$_a$
46	SERVICE school care college	public$_m$ medical$_m$ provide$_p$ deny$_p$ receive$_p$ attend$_p$
47	TRAFFICKING rights group flight	human$_m$ international$_m$ commercial$_m$ be$_a$ have$_a$ take$_a$
48	WORKER I immigration student company	foreign$_m$ legal$_m$ skilled$_m$ hire$_p$ american$_m$ have$_a$
49	WORKER I people woman man	mexican$_m$ immigrant$_m$ undocumented$_m$ migrant$_m$ illegal$_m$
50	YEAR program month income	fiscal$_m$ last$_m$ end$_a$ next$_m$ previous$_m$ begin$_a$

Table 1: Personas with their associated mention words and relation tuples (a = agent, p patient, m = modifier/attribute); I denotes "immigrant."

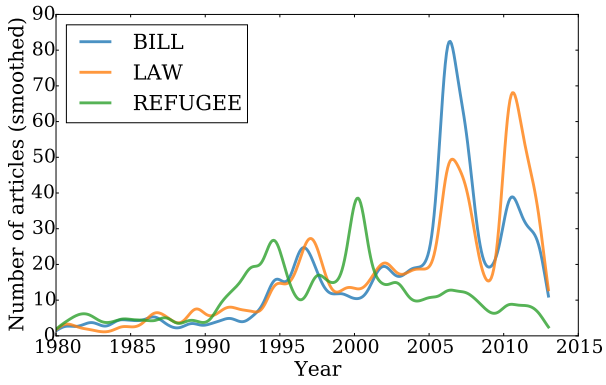


Figure 2: Temporal patterns of the mentions of selected personas.

Refugee story cluster A		
Frequency	Persona	ID
0.49	REFUGEE immigrant boy	44
0.10	BUSH official mcnary	6
0.06	IMMIGRANT man alien	29
0.05	ASYLUM crime refugee	2
Refugee story cluster B		
Frequency	Persona	ID
0.29	MAN immigrant woman	38
0.23	REFUGEE immigrant boy	44
0.12	COURT lawsuit suit	13
0.10	GOVERNMENT court judge	16

Table 2: Truncated distribution over personas for the two clusters depicted in Figure 3. IDs index into Table 1.

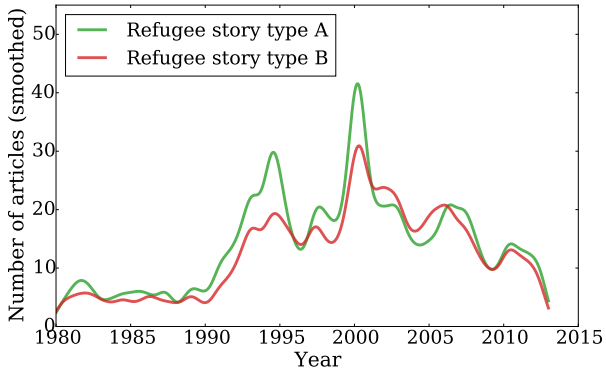


Figure 3: Temporal patterns of two clusters with the greatest overall representation of the *refugee* persona.

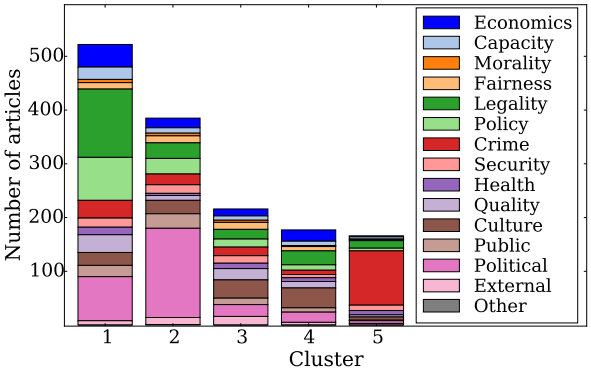


Figure 4: Number of annotated articles in each of the five most frequent clusters, with colors showing the proportion of articles annotated with each primary frame.

2. Type A, which includes a story with the headline "Protesters vow to keep Elián in U.S.," emphasizes political aspects, while type B (e.g., "Court says no to rights for refugees") emphasizes legal aspects. Note that *Political* and *Legality* are two of the framing dimensions used in the MFC.

Do these persona-cast clusters relate to frames? For the five most common story clusters, (which have no overlap with the two refugee story types), Figure 4 shows the number of annotated articles with each of the primary frames if we assign each article to its most likely cluster. The second and fifth clusters correlate particularly well with primary frames (*Political* and *Crime*, respectively). This is further reinforced by looking at the most frequent persona for each of these story clusters which are *candidate* (ID 7) for the second and *immigrant* (ID 22), characterized by illegal$_m$ and arrest$_p$, for the fifth.

7 Experiments: Personas and Framing

We evaluate personas as features for automatic analysis of framing and tone, as defined in the MFC (§4). Specifically, we build multi-class text classifiers (separately) for the primary frame and the tone of a news article, for which there are 15 and 3 classes, respectively. Because there are only a few thousand annotated articles, we applied 10-fold cross-validation to estimate performance.

Features are derived from our model by considering each persona and each story cluster as a potential feature. A document's feature values for story types are the proportion of samples in which it was assigned to each cluster. Persona feature values are similarly derived by the proportion of samples in which each entity was assigned to each persona, with the persona values for each entity in each document summed into a single set of persona values per

Primary frame					
Features:	MF	(W)	(W,P_1)	(W,P_2)	(W,P_2,S)
Accuracy:	0.174	0.529	0.537	*0.540	0.537
# Features:	0	3.9k	3.5k	3.5k	2.8k
Tone					
Accuracy:	0.497	0.628	0.631	0.628	0.630
# Features:	0	5.0k	5.0k	5.0k	4.0k

Table 3: Evaluation using a direct comparison to a simple baseline. Each model uses the union of listed features. (W = unigrams and bigrams, P_1 = personas from DPM, P_2 = personas from our model, S = story clusters; MF = always predict most frequent class.) * indicates a statistically significant difference compared to the (W) baseline ($p<0.05$).

document. We did not use the topics (z) discovered by our model as features.

7.1 Experiment 1: Direct Comparison

For the first experiment, we train independent multiclass logistic regression classifiers for predicting primary frame and tone. We consider adding persona and/or story cluster features to baseline classifiers based only on unigrams and bigrams with binarized counts, a simple but robust baseline (Wang and Manning, 2012).[6] In all cases, we use L_1 regularization and use 5-fold cross validation within each split's training set to determine the strength of regularization. We then repeat this for each of the 10 folds, thereby producing one prediction (of primary frame and tone) for every annotated article. The results of this experiment are given in Table 3; for predicting the primary frame, classifiers that used persona and/or story cluster features achieve higher accuracy than the bag-of-words baseline (W); the classifier using personas from our model but not story clusters is significantly better than the baseline.[7] The enhanced models are also more compact, on average, using fewer effective features. A benefit to predicting tone is also observed, but it did not reach statistical significance.

7.2 Experiment 2: Automatic Evaluation

Although bag-of-n-grams models are known to be a strong baseline for text classification, researchers familiar with the extensive catalogue of features offered by NLP will potentially see them as a straw man. We propose a new and more rigorous method of comparison, in which a wide range of features are offered to an automatic model selection algorithm for each of the prediction tasks, with the features to be evaluated withheld from the baseline.

Because no single combination of features and regularization strength is best for all situations, it is an empirical question which features are best for each task. We therefore make use of Bayesian optimization (Bayesopt) to make as many modeling decisions as possible (Pelikan, 2005; Snoek et al., 2012; Bergstra et al., 2015; Yogatama et al., 2015).

In particular, let F be the set of features that might be used as input to any text classification algorithm. Let f be a new feature that is being proposed. Allow the inclusion or exclusion of each feature in the feature set to be a hyperparameter to be optimized, along with any additional decisions such as input transformations (e.g., lowercasing), and feature transformations (e.g., normalization). Using an automatic model selection algorithm such as Bayesian optimization, allow the performance on the validation set to guide choices about all of these hyperparameters on each iteration, and set up two independent experiments.

For the first condition, A_1, allow the algorithm access to all features in F. For the second, A_2, allow the algorithm access to all features in $F \cup f$. After R iterations of each, choose the best model or the best set of models from each of A_1 and A_2 (M_1 and M_2, respectively), based on performance on the validation set. Finally, compare the selected models in terms of performance on the test set (using an appropriate metric such as F_1), and examine the features included in each of the best models. If f is a helpful feature, we should expect to see that, a) $F_1(M_2) > F_1(M_1)$, and b), f is included in the best model(s) found by A_2.

If $F_1(M_2) > F_1(M_1)$ but f is not included in the best models from A_2, this suggests that the performance improvement may simply be a matter of chance, and there is no evidence that f is helpful. By contrast, if f is included in the best models, but $F_1(M_2)$ is not significantly better than $F_1(M_1)$, this suggests that f is offering some value, perhaps in a more compressed form of the useful signal from other features, but does not actually offer better per-

[6] We also binarized the persona feature values.

[7] Two-tailed McNemar's test ($p<0.05$).

Features:	(B)	(B,P_1)	(B,P_2)	(B,P_2,S)
Primary frame	0.566	0.568	0.568	0.567
Tone	0.667	0.671	0.667	0.671

Table 4: Mean accuracy of the best three iterations from Bayesian optimization (chosen based on validation accuracy). (B = features from many NLP tools, P_1=personas from the DPM, P_2 = personas from our model, S=story clusters.)

formance.

For this experiment, we use the tree-structured Parzen estimator for Bayesian optimization (Bergstra et al., 2015), with L_1-regularized logistic regression as the underlying classifier, and set $R = 40$. In addition to the entities and story clusters identified by these models, we allow these classifiers access to a large set of features, including unigrams, bigrams, parts of speech, named entities, dependency tuples, ordinal sentiment values (Manning et al., 2014), multi-word expressions (Justeson and Katz, 1995), supersense tags (Schneider and Smith, 2015), Brown clusters (Brown et al., 1992), frame semantic features (Das et al., 2010), and topics produced by standard LDA (Blei et al., 2003). The inclusion or exclusion of each feature is determined automatically on each iteration, along with feature transformations (removal of rare words, lowercasing, and binary or normalized counts).

The baseline, denoted "B," offers all features except personas and story clusters to Bayesopt; we consider adding DPM personas, our model's personas, and our model's personas and story clusters. Table 4 shows test-set accuracy for each setup, averaged across the three best models returned by Bayesopt.

Using this more rigorous form of evaluation, approximately the same accuracy is obtained in all experimental conditions. However, we can still gain insight into which features are useful by examining those selected by the best models in each condition. For primary frame prediction, both personas and story clusters are included by the best models in every case where they have been offered as possible features, as are unigrams, dependency tuples, and semantic frames. Other commonly-selected features include bigrams and part of speech tags. For predicting tone, personas are only included by half of the best models, with the most common features being unigrams, bigrams, semantic frames, and Brown clusters. As expected, the best models in each condition obtain better performance than the models from experiment 1, thanks to the inclusion of additional features and transformations.

This secondary evaluation suggests that for this task, persona features are useful in predicting the primary frame, but are unable to offer improved performance over existing features, such as semantic frames. However, the fact that that both personas and story clusters are included by all the best models for predicting the primary frame suggests that they are competitive with other features, and perhaps offer useful information in a more compact form.

8 Qualitative Evaluation

Prior to exposure to any output of our model, one of the co-authors on this paper (Gross, who has expertise in both framing and the immigration issue) prepared a list of personas he expected to frequently occur in American news coverage of immigration. Given the example of the "skilled immigrant," he listed 22 additional named personas, along with a few examples of things they do, things done to them, and attributes.

The list he prepared includes several different characterizations of immigrants (low-skilled, unauthorized, legal, citizen children, undocumented children, refugees, naturalized citizens), non-immigrant personas (U.S. workers, smugglers, politicians, officials, border patrol, vigilantes), related pairs (pro / anti advocacy groups, employers / guest workers, criminals / victims), and a few more conceptual entities (the border, bills, executive actions). Of these, almost all are arguably represented in the personas we have discovered. However, there is rarely a perfect one-to-one mapping: predefined personas are sometimes merged (e.g., "the border" and "border patrols") or split (e.g., legislation, employers, and various categories of immigrants). Personas which don't emerge from our model include smugglers, guest workers, vigilantes, and victims of immigrant criminals. On the other hand, our model proposes far more non-person entities, such as ID cards, courts, companies, jobs, and programs.

These partial matchings between predefined personas and the results of our model are generally

identifiable by comparing the names given to the predefined personas to the the most commonly occurring mention words and attributes of our discovered personas. The attributes and action words given to the predefined personas are harder to evaluate, as many of them are rare (e.g. politicians "vacillate") or compound phrases (e.g. low-skilled immigrants "do jobs Americans won't do") that tend to miss the more obvious properties captured by our model. For example, the *employer* persona captured by our model engages in actions like *hire*, *employ*, and *pay*. By contrast, the terms given for the predefined "business owners" persona are "lobby" and "rely on immigrant labor." Our unsupervised discovery of this persona can clearly be matched to the predefined persona in this case, but doesn't provide such fine-grained insight into how they might be characterized.

The best match between predefined and discovered personas is the U.S.-Mexican border. Of the words given for the predefined persona, almost all are more frequently associated with *border* than with any other discovered persona ("Mexican-U.S.," "lawless," "porous," "unprotected," "guarded," and "militarized"). The most commonly associated words discovered by our model that are missing from the predefined description include *crossed*, *secured*, *southern*, and *closed*.

While this qualitative evaluation helps to demonstrate the face validity of our model, it would be better to have a more comprehensive set of predefined personas, based on input from additional experts. Moreover, it also illustrates the challenge of trying to match the output of an unsupervised model to expected results. Not only is some merging and splitting of categories inevitable, there was a mismatch in this case in the types of entities to be described (people as opposed to more abstract entities), and the ways of describing them (rare but specific words as opposed to more generic but potentially obvious terms).

9 Related Work

Much NLP has focused on identifying entities or events (Ratinov and Roth, 2009; Ritter et al., 2012), analyzing schemes or narrative events in terms of characters (Chambers and Jurafsky, 2009), inferring the relationships between entities (O'Connor et al., 2013; Iyyer et al., 2016), and predicting personality types from text (Flekova and Gurevych, 2015). Bamman also applied variants of the DPM to characters in novels (Bamman et al., 2014).

Previous work on sentiment, stance, and opinion mining has focused on recognizing stance or political sentiment in online ideological debates (Somasundaran and Wiebe, 2010; Hasan and Ng, 2014; Sridhar et al., 2015), and other forms of social media (O'Connor et al., 2010; Agarwal et al., 2011), and recently through the lens of connotation frames (Rashkin et al., 2016). Opinion mining and sentiment analysis are the subject of ongoing research in NLP and have long served as test platforms for new methodologies (Socher et al., 2013; İrsoy and Cardie, 2014; Tai et al., 2015)

Framing is arguably one of the most important concepts in the social sciences, with roots in to sociology, psychology, and mass communication (Gitlin, 1980; Benford and Snow, 2000; D'Angelo and Kuypers, 2010); the scope and relevance of framing is widely debated (Rees et al., 2001), with many authors applying the concept of framing to analyzing documents on particular issues (Baumgartner et al., 2008; Berinsky and Kinder, 2006).

10 Conclusion

We have extended models for discovering latent personas to simultaneously cluster documents by their "casts" of personas. Our exploration of the model's inferences and their incorporation into a challenging text analysis task—characterizing coarse-grained framing in news articles—demonstrate that personas are a useful abstraction when applying NLP to social-scientific inquiry. Finally, we introduced a Bayesian optimization approach to rigorously assess the usefulness of new features in machine learning tasks.

Acknowledgments

The authors thank members of the ARK group and anonymous reviewers for helpful feedback on this work. This research was made possible by a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship (to D.C.), a Bloomberg Data Science Research Grant (to J.H.G., A.E.B., and N.A.S.), and a University of Washington Innovation Award (to N.A.S.).

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J. Passonneau. 2011. Sentiment analysis of twitter data. In *Proc. of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014)*.

D. Aldous. 1985. Exchangeability and related topics. In *École d'Été St Flour 1983*, pages 1–198. Springer-Verlag.

Charles E. Antoniak. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *Annals of Statistics*, 2(6), November.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proc. of ACL*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proc. of ACL*.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *Proc. of NAACL*.

Frank R. Baumgartner, Suzanna L. De Boef, and Amber E. Boydstun. 2008. *The decline of the death penalty and the discovery of innocence*. Cambridge University Press.

Robert D. Benford and David A. Snow. 2000. Framing processes and social movements: An overview and assessment. *Annual Review of Sociology*, 26:611–639.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science and Discovery*, 8(1).

Adam J. Berinsky and Donald R. Kinder. 2006. Making sense of issues through media frames: Understanding the Kosovo crisis. *Journal of Politics*, 68(3):640–656.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram models of natural language. *Computional Linguistics*, 18(4):467–479.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proc. of ACL*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proc. of ACL*.

Eunsol Choi, Chenhao Tan, Lillian Lee, Cristian Danescu-Niculescu-Mizil, and Jennifer Spindel. 2012. Hedge detection as a lens on framing in the GMO debates: A position paper. In *Proc of. Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 70–79.

Paul D'Angelo and Jim A. Kuypers. 2010. *Doing News Framing Analysis*. Routledge.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proc. of NAACL*.

Robert M. Entman. 2007. Framing bias: Media in the distribution of power. *Journal of Communication*, 57(1):163–173.

Michael D. Escobar and Mike West. 1994. Bayesian density estimation and inference using mixtures. *J. Amer. Statist. Assoc.*, 90:577–588.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89.

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proc. of EMNLP*.

Matthew Gentzkow and Jesse M. Shapiro. 2010. What drives media slant? Evidence from U.S. daily newspapers. *Econometrica*, 78(1):35–71.

Todd Gitlin. 1980. *The Whole World is Watching*. Berkeley: University of California Press.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proc. of ACL*.

Glenn Greenwald. 2014. *No Place to Hide*. Picador.

Eric Hardisty, Jordan L. Boyd-Graber, and Philip Resnik. 2010. Modeling perspective using adaptor grammars. In *Proc. of EMNLP*.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proc. of IJCNLP*.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proc. of EMNLP*.

Edward S. Herman and Noam Chomsky. 1988. *Manufacturing Consent*. Vintage.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proc of EMNLP*.

Mohit Iyyer, Peter Enns, Jordan L. Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proc. of NAACL*.

J. Justeson and S. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proc. of KDD*.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? Identifying perspectives at the document and sentence levels. In *Proc. of CoNNL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL*.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Proc. of NIPS*.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to Republican legislators in the 112th congress. In *Proc. of ACL*.

Vlad Niculae, Caroline Suen, Justine Zhang, Cristian Danescu-Niculescu-Mizil, , and Jure Leskovec. 2015. QUOTUS: The structure of political media coverage as revealed by quoting patterns. In *Proceedings of WWW 2015*.

Brendan T. O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM*.

Brendan O'Connor, Brandon M. Stewart, and Noah A. Smith. 2013. Learning to extract international relations from political context. In *Proc. of ACL*.

Zhongdang Pan and Gerald M. Kosicki. 1993. Framing analysis: An approach to news discourse. *Political communication*, 10(1):55–75.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2).

M Pelikan. 2005. Bayesian optimization algorithm. In *Hierarchical Bayesian optimization algorithm*, pages 31–48. Springer.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation frames: A data-driven investigation. In *Proc. of ACL*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNNL*.

Stephen D. Rees, Oscar H. Gandy Jr., , and August E. Grant, editors. 2001. *Framing Public Life*. Routledge.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *KDD*.

Anne Schneider and Helen Ingram. 1993. Social construction of target populations: Implications for politics and policy. *The American Political Science Review*, 87(2):334–347.

Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *Proc. of ACL*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

David A. Smith, Ryan Cordell, and Elizabeth Maddock Dillon. 2013. Infectious texts: modeling text reuse in nineteenth-century newspapers. In *Proc. of IEEE International Conference on Big Data*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Proc. of NIPS*.

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

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proc. of ACL*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*.

Oren Tsur, Dan Calacci, and David Lazer. 2015. Frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proc. of ACL*.

Baldwin Van Gorp. 2010. Strategies to take subjectivity out of framing analysis. In Paul D'Angelo and Jim A. Kuypers, editors, *Doing News Framing Analysis*, chapter 4, pages 84–109. Routledge.

Marilyn A. Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proc. of NAACL*.

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

Dani Yogatama, Lingpeng Kong, and Noah A. Smith. 2015. Bayesian optimization of text representations. In *EMNLP*.

The Teams Corpus and Entrainment in Multi-Party Spoken Dialogues

Diane Litman
University of Pittsburgh
Pittsburgh, PA

Susannah Paletz
University of Maryland
College Park, MD

Zahra Rahimi and **Stefani Allegretti** and **Caitlin Rice**
University of Pittsburgh
Pittsburgh, PA

Abstract

When interacting individuals *entrain*, they begin to speak more like each other. To support research on entrainment in cooperative multi-party dialogues, we have created a corpus where teams of three or four speakers play two rounds of a cooperative board game. We describe the experimental design and technical infrastructure used to collect our corpus, which consists of audio, video, transcriptions, and questionnaire data for 63 teams (47 hours of audio). We illustrate the use of our corpus as a novel resource for studying team entrainment by 1) developing and evaluating team-level acoustic-prosodic entrainment measures that extend existing dyad measures, and 2) investigating relationships between team entrainment and participation dominance.

1 Introduction

Linguistic entrainment[1] refers to the convergence of (para)linguistic features across speakers during conversation (Brennan and Clark, 1996; Porzel et al., 2006). Research has found that speakers entrain to both human and computer conversational partners, with the amount of entrainment often positively related to conversational and task success. However, most prior work has focused on the study of entrainment during two-party dialogues, rather than during the multi-party conversations typical of teams.

To support the study of entrainment during multi-party cooperative dialogue, we have created a large-scale corpus (over 47 hours of recordings) of teams of three or four speakers playing a cooperative board game requiring conversation. The corpus consists of audio, video, transcriptions, and questionnaire data for 63 teams. The goal of the corpus is to provide a freely-available data resource for the development and evaluation of multi-party entrainment measures that can be 1) computed using language technologies, 2) motivated and validated by the literature on teams, and 3) associated with measures of task and dialogue success.

In this paper, we first describe the experimental design and technical infrastructure used to create our corpus. We then present two case studies illustrating the use of our corpus as a novel resource for studying team entrainment: quantifying acoustic-prosodic entrainment at the team-level rather than the dyad-level, and incorporating a construct from the teamwork literature into the study of entrainment.

2 Background and Related Work

The development of methods for automatically quantifying entrainment in text and speech data is an active research area, as entrainment has been shown to correlate with success measures or with social variables for a variety of phenomena, e.g., acoustic-prosodic, lexical, and syntactic (Nenkova et al., 2008; Reitter and Moore, 2007; Mitchell et al., 2012; Levitan et al., 2012; Lee et al., 2011; Stoyanchev and Stent, 2009; Lopes et al., 2013; Lubold and Pon-Barry, 2014; Moon et al., 2014; Sinha and Cassell, 2015). Such research, in turn, requires corpora with certain properties. A high-quality spoken language corpus for studying entrain-

[1] Other terms in the literature include accommodation, adaptation, alignment, convergence, coordination and priming.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1421–1431,
Austin, Texas, November 1-5, 2016.

ment would include transcriptions suitable for natural language processing, audio recordings suitable for signal processing, and meta-data such as task success or speaker demographics.

While most research has focused on quantifying the amount of entrainment between pairs of speakers, recent work has started to develop measures for quantifying entrainment between larger groups of speakers (Friedberg et al., 2012; Danescu-Niculescu-Mizil et al., 2012; Gonzales et al., 2010). To date, however, mainly simple methods such as unweighted averaging have been used to move from pairs to groups, and the focus of prior work has been on text rather than speech (e.g., Wikipedia, computer-mediated discussions, lexical analysis of transcriptions). In this paper we both investigate group acoustic-prosodic entrainment and examine relationships between group entrainment and a factor from the teamwork literature called participation equality / dominance (Paletz and Schunn, 2011).

Also, while freely available speech corpora have supported the study of entrainment in two-party dialogues (e.g., Switchboard, Maptask, the Columbia Games Corpus, Let's Go), few community resources exist for the study of multi-party entrainment. Some multi-party resources are only text-based (e.g., the online Slashdot forum (Allen et al., 2014), chat dialogues (Afantenos et al., 2015)). Those speech resources that do exist are often less than ideal as they were created for other purposes (e.g., Supreme Court arguments (Beňuš et al., 2014; Danescu-Niculescu-Mizil et al., 2012), the AMI meeting corpus (Carletta et al., 2006)). Although not created to study entrainment, the KTH-Idiap Group-Interviewing corpus (Oertel et al., 2014) is perhaps most relevant as it was explicitly designed to support research on group dynamics. However, the corpus contains only 5 hours of speech, and participants were PhD students so did not differ on variables such as age and social status.

The Teams corpus presented and used in this paper was designed to add several notable extensions to existing multi-party spoken dialogue resources. In particular, the Teams corpus was experimentally collected to constrain the team processes, tasks, and outcomes in ways that facilitate an investigation of team entrainment. First, the corpus consists of over 45 hours of cooperative task-oriented dialogues between three or four speakers, where audio and video files were collected and transcribed using best practices for computational processing. Second, the corpus was collected using an experimental manipulation informed by the organizational and social psychological literature on team processes in order to create high versus low-entrainment conditions. Third, since the social psychological literature suggests that team dynamics are more complex than an average of dyadic interactions, validated questionnaires were used to collect relevant variables of interest to researchers on teams, and individual participants were recruited so that teams would exhibit diversity with respect to these variables.

3 Experimental Study

The Teams corpus was collected in a laboratory experiment. The laboratory setting enabled high-quality audio and video capture, while the experimental study allowed manipulations to vary entrainment and to collect measures of team processes.[2]

3.1 Design

Our data collection was via an experiment with a 2 by 2 within-and-between subjects design. Teams of 3-4 participants spent 2-3 hours in our lab taking self-report questionnaires and being audio and video-taped playing a cooperative board game. Two manipulations were designed to increase the likelihood of task success and entrainment [3]. For the first manipulation, half the teams were given a *teamwork* training intervention in which participants were given specific advice based on a needs analysis of the team skills important to the game (Gregory et al., 2013). Such mixed teamwork/taskwork training has been shown to improve team process outcomes (Salas et al., 2008). The other half only had

[2] A lab experiment involving a two-player game requiring spoken communication was similarly used to collect the Columbia Games Corpus of 12 spontaneous task-oriented dyadic conversations, which has been used in multiple studies of two-party entrainment (Levitan and Hirschberg, 2011; Levitan et al., 2012; Levitan et al., 2011). Our corpus is approximately 5 times larger, includes speech from teams rather than from dyads, and relatedly includes new types of team-related meta-data. Our corpus also contains both video and audio as our dialogues were face-to-face rather than restricted to voice.

[3] As discussed in Section 2, prior research has often found positive relationships between success and entrainment.

M: And then [I'm here.]
E: [Oh.]
P: [Yeah] probably wanna save [Whispering Garden.]
E: [Whispering- Yeah.]
M: [Uh yeah, that's one,] [two,]
P: [Yeah.]
M: [three.]
P: [Perfect.]

Figure 1: Dialogue excerpt from a Forbidden Island^{TM} game. E=Engineer, M=Messenger, and P=Pilot roles in the game. Square brackets indicate overlapping speech.

training on the rules of the game, which all teams received.

For the second manipulation, each team played two isomorphic versions of the game. The game was originally designed to be played multiple times, with each session unique depending on the random placement of specific board tiles and the order of deck cards. To maintain experimental control, two specific deck card orders and board tile patterns that had the same underlying opportunities and obstacles were created. 33 teams played one game first, and 30 teams played the other game first. In either case, by the second time, the team should have a better grasp of the game and appropriate strategies.

3.2 Task

For the team task, we chose the cooperative board game Forbidden Island^{TM}, where players take on the roles of adventurers seeking treasures on an island before it is flooded. We chose this game because it both demands collaboration and is logistically feasible for our experiment. The cooperative task-oriented nature of the game requires players to communicate to achieve their goals (e.g., discussing cards and strategies in real time, see Figure 1), lending itself directly to eliciting entrainment. Further, the game gives each player a different role to achieve the team goals, as well as game-specific terminology, generalizing to real-world situations with teamwork (e.g. aviation, health care). Logistically, Forbidden Island^{TM} can be played equally well with three or four players. This feature allowed us to schedule teams of four participants, but still play the game even if only three showed up. A typical game is also short enough to be played twice within an experimental session. Game rules were adapted to ensure the game difficulty was suitable for novice players (e.g., requiring three rather than four treasures be found before completing the game). As noted in Section 3.1, two isomorphic versions of the game were constructed so that the first and second games would appear visually different but the difficulty level would be identical between and within teams. This isomorphism was accomplished by maintaining the position of tiles and cards that determined order-of-play and game difficulty, while systematically shifting the position of non-critical tiles and cards.

3.3 Recruitment

Participants aged 18 years and older who are native speakers of American English were recruited via electronic and hardcopy flyers and paid for their time. They were males and females of any ethnicity from a university and its surrounding community. To increase ethnicity, race, and age diversity (rare in corpora typically drawn only from student samples), we advertised in non-student locations in predominantly ethnic minority neighborhoods.

3.4 Procedure

As a team's participants arrived in the lab, each completed a questionnaire to collect personality, demographic, and other information such as experience with the game Forbidden Island^{TM}. Participants were then taught how to play the game by watching a video and playing a tutorial game, then given a few minutes to ask specific questions. Teams in the intervention condition (the between-subjects manipulation of our experimental design) were given an extra 10 minutes before the first game to receive training about teamwork strategies such as team roles, communication needs, and how to coordinate their actions (Gregory et al., 2013), as well as additional information adapted for the Forbidden Island^{TM} task itself. Then each team played the game twice for no more than 35 minutes per game. Teams were told that not completing a game in 35 minutes counted as a loss, and that winning scores for the rest of the games would be inversely related to game length (a timer was displayed on a computer monitor during each game). The intervention condition teams were also given an additional 5 minutes before the second game to discuss what went well and poorly with their team processes. Finally, both between and after the two games, all participants filled out question-

naires regarding their team processes.

3.5 Data Capture

Game participants were located around a round table 48 inches in diameter in our game-playing lab, enabling comfortable participant access to the game board. Each participant sat in a particular location depending on their role in the game. The survey data were collected in a separate workstation lab using Qualtrics, a web-based, survey software tool.

To collect high-quality speech data with minimal cross-talk, audio was recorded using Sennheiser ME 3-ew close-talk microphones. Each microphone was connected to a Presonus AudioBox 1818VSL multichannel audio interface sampling at 96k, 24 bits. Audio recordings were monitored using Reaper Digital Audio Workstation v 4.76. Each game yielded one stereo recording with the synchronized speech from all speakers, along with 3 or 4 individual files (one per participant) representing the audio recording from each microphone. Reaper was used to render .WAV files with a 48000 Hz sampling rate and a 16 bit PCM Wav bit depth.

To complement the speech, four wall-mounted Zoom Q4 cameras captured WVGA/30 .MOV video recordings. The audio streams recorded from the cameras are at the central room, not the individual, level. A master audio signal was used to synchronize the videos with each other and with the audio from the microphones. Note that the videos also provide backup audio streams (recording at 256kbps AAC) for the microphones. In addition, the videos provide information about the games that are not always obvious from the audio[4], as well as non-verbal data for future analysis (e.g., of gesture or posture).

4 The Teams Corpus

Our experiment ran from February through August 2015, yielding over 47 hours of recordings from 63 teams[5] (216 individuals).

4.1 Descriptive Statistics

The 216 participants in our experiment were on average 25.3 years old (min=18, max=67, *SD*=11.3).

	Control (n=31)		Intervention (n=32)	
	3-per.	4-per.	3-per.	4-per.
# of teams	20	11	16	16
avg g1 time	26.6	28.0	26.4	27.3
avg g2 time	18.0	17.7	18.2	19.7

Table 1: Team descriptives ($n = 63$).

There were 135 females (62.5%) and 81 males (37.5%). The highest level of education (whether completed or not) ranged from high school (28 participants, 13.0%) to undergraduate (153 participants, 70.8%) to postgraduate/professional (35 participants, 16.2%). 145 participants (67.1%) were currently students. 35 participants (16.2%) knew at least one of their team members. The most frequent self-reported ethnicity/races were Caucasian (166), Asian (31), Black (24), and Hispanic (10) (multiple ethnicities were allowed). Thus, our recruitment yielded demographically diverse participants in ways that are useful for team research.

Table 1 shows the distribution of the teams in our corpus by experimental condition (control versus intervention) and team size (3 versus 4 person). For each of these groups of teams, the table also shows the average time they took in minutes to play games 1 and 2, respectively. A 3-way ANOVA shows a significant within-team effect for game, with first games taking significantly longer than second games (27.1 vs. 18.4 minutes, $p < .001$). The average game length did not significantly differ by experimental condition ($p > .7$) or by team size ($p > .3$), and there were also no interaction effects.

Our team-level data provides preliminary evidence for the success of one of our experimental manipulations, as second games were significantly shorter than first games. [6]

4.2 Audio Segmentation and Transcription

After the experiment was completed, our multiple audio track speech was manually segmented and transcribed using the Higgins Annotation Tool[7].

[4] We are currently using the videos to annotate game-specific measures of task success.

[5] A power analysis for our experiment yielded a minimum target sample size of 52 teams.

[6] The time to complete a game is an easy to compute but a shallow (inverse) success measure. We are currently annotating our data for game-specific and dialogue-based success measures, and will also examine success in terms of team process measures computable from the questionnaires (Section 4.3).

[7] http://www.speech.kth.se/hat/

Each audio track, which corresponds to each individual player, appears on a separate line in Higgins. A time stamp line applies to all of the (synchronized) audio tracks. To do transcription, each participant's speech is first segmented into inter-pausal units, pause-free chunks of speech from a single speaker (Levitan and Hirschberg, 2011). The threshold used for pause length (i.e., silence) for our corpus is 200 milliseconds. Once speech is segmented in a specific audio track, a corresponding text line appears where the transcriber manually types in the text for the corresponding audio segment. Within each transcription, text segments may also be defined and assigned values. We are using segments to annotate non-lexical aspects such as laughs.

4.3 Questionnaire Data

The pre-game questionnaire was used to collect individual demographic information such as discussed in Section 4.1, and self-reported data related to personality (John et al., 1991), cognitive styles (Miron et al., 2004), and collective orientation ("the propensity to work in a collective manner in team settings" (Driskell et al., 2010)). The between and post-game questionnaires elicited perceptions of team processes such as cohesion, satisfaction, and potency/efficacy (Wendt et al., 2009; Wageman et al., 2005; Guzzo et al., 1993). Such information was collected as a novel resource for studying multi-party entrainment, since team processes have been shown to be positively related to performance (Beal et al., 2003; Mullen and Copper, 1994).

4.4 Public Release

The Teams corpus will be freely available for research purposes[8], with the first release coordinated with the publication of this paper. The team level contents of the first release will consist of 63 game 1 and 62 [9] game 2 WAV files. The individual level contents of this release will consist of the demographic responses for the 216 participants in XLSX format. Later corpus releases will include associated audio segmentations and transcriptions in XML format, game-level video files, and personality and team process measures.

[8] https://sites.google.com/site/teamentrainmentstudy/corpus

[9] One audio file was not properly saved during the experiment. The corresponding single-channel audio extracted from the game's video will be provided instead.

5 Case Studies Using the Teams Corpus

This section presents results from two case studies illustrating the use of the Teams corpus for novel research in multi-party dialogue entrainment. The first study proposes new team level measures that build on existing dyad-level measures of proximity and convergence, then uses these team measures to investigate whether prior dyad-level acoustic-prosodic entrainment findings generalize to teams. The second study investigates relationships between team convergence and participation equality / dominance.

5.1 Acoustic-Prosodic Team Entrainment

Speakers do not entrain on all linguistic features of conversations, and when they do entrain, they may entrain in different ways on different features. In this section we examine whether teams entrain on different acoustic-prosodic features during each of their two game conversations. Our current approach to measuring team-level entrainment is based on averaging dyad-level measures. We build on two dyad measures, namely, proximity and convergence (Levitan and Hirschberg, 2011). In a conversation, proximity measures feature similarity over the entire conversation, while convergence measures an increase in feature proximity over time.

5.1.1 Feature Extraction from Speaker Audio

We focus on the acoustic-prosodic dimensions of pitch, intensity, and voice quality, following previous work on dyad entrainment (Levitan and Hirschberg, 2011; Lubold and Pon-Barry, 2014; Borrie et al., 2015). Pitch is related to the frequency of the sound wave. Intensity describes the rate of energy flow. Jitter and shimmer are measures of variations of frequency and energy, respectively, which are descriptive of voice quality. We use the Praat software (Boersma and Heuven, 2002) to extract the following 9 acoustic-prosodic features: minimum (min), maximum (max), mean and standard deviation (SD) of pitch; min, max, mean of intensity;

local jitter[10]; and local shimmer[11]. Features are extracted separately for each speaker and for each game. Before feature extraction, each game-level audio file for each speaker is pre-processed to remove silences (using a threshold of 1 second).

5.1.2 Measuring Team Proximity

Proximity quantifies the similarity of a feature value between conversational partners over their entire conversation. Intuitively, if a team has entrained on a feature in terms of proximity during a particular game, speakers within the same team should be more similar (or equivalently, less different) to each other than to all the other speakers in the corpus who are not on their team and are playing the same game (i.e., game 1 or game 2). For each game we computed a team-level partner difference ($TDiff_p$) and a team-level other difference ($TDiff_o$). In Section 5.1.4 we report paired t-test analyses to infer entrainment within a game when $TDiff_p$ is significantly smaller than $TDiff_o$.

The partner difference for a speaker in a dyad (Levitan and Hirschberg, 2011) is the absolute difference between the feature value for a speaker and her partner. For each team, we averaged these absolute values for all members of the team:

$$TDiff_p = \frac{\sum_{\forall i \neq j \in team}(|speaker_i - speaker_j|)}{|team| * (|team| - 1)} \quad (1)$$

The other difference for a speaker in a dyad (Levitan and Hirschberg, 2011) is the mean of the absolute differences between the speaker's value for a feature and the values of each of the speakers in the corpus (for the same game number) with whom the speaker was not partnered (set X in Formula 2). For each team, we averaged these means for all the members of the team:

$$TDiff_o = \frac{\sum_{\forall i \in team}(\frac{\sum_j |speaker_i - X_j|}{|X|})}{|team|} \quad (2)$$

For proximity, all of the feature values were normalized within a game based on gender[12] using z-scores ($z = \frac{v_{ij} - \mu_j}{\sigma_j}$; v_{ij} = value of speaker i in game j where $j \in \{1, 2\}$, μ_j = gender mean in game j, and σ_j = gender standard deviation in game j.)

Feature	Game1	Game2
Pitch-min	0.844	0.193
Pitch-max	−1.092	0.022
Pitch-mean	−1.297	−1.294
Pitch-sd	−0.407	−1.652
Intensity-mean	−4.469*	−4.911*
Intensity-min	−2.653*	−2.069*
Intensity-max	−3.625*	−2.853*
Shimmer-local	−2.390*	−2.782*
Jitter-local	−1.242	−2.702*

Table 2: Proximity t-values of a paired t-test comparing team-level partner ($TDiff_p$) vs. other ($TDiff_o$). Negative t-values indicate that partner differences are smaller than other differences. * $p < .05$. $n = 62$.

5.1.3 Measuring Team Convergence

Intuitively, there is evidence of convergence when speakers within a conversation become more similar to each other later in the conversation. While feature value differences are compared across teams to infer proximity entrainment, feature value differences within a single team are compared across time for convergence entrainment. Since differing time intervals have been examined in the dyad literature, we compared features extracted from the first versus last three, five, and seven minutes of each game, as well as from the two game halves.[13] Convergence was inferred via paired t-tests when the partner differences (Equation 1) in the second time interval were significantly smaller than in the earlier time interval (e.g., the $TDiff_p$ in the last 3 minutes of game 1 is smaller than $TDiff_p$ in the first 3 minutes of game 1). To break the games into different time intervals for feature extraction, we used the raw audio files to extract the breaking points of the conversation and then mapped these points to each of the processed audio files where silence was removed.

5.1.4 Team-Level Entrainment Results

The proximity results are shown in Table 2. Negative t-values indicate that differences between speak-

[10]The average absolute difference between the amplitudes of consecutive periods, divided by the average amplitude.

[11]The average absolute difference between consecutive periods, divided by the average amplitude.

[12]Normalization is done only for proximity, since comparisons for convergence are within (rather than between) teams.

[13](Levitan and Hirschberg, 2011) also looked for convergence between the two halves of the first game in their corpus.

	First vs. last 3 minutes		First vs. last 5 minutes		First vs. last 7 minutes		First vs. second half	
Feature	Game1	Game2	Game1	Game2	Game1	Game2	Game1	Game2
Pitch-min	**2.474***	−0.709	1.487	−1.299	1.359	−1.622	0.329	−0.884
Pitch-max	**4.947***	1.260	1.892	−0.468	1.348	−0.424	0.457	0.627
Pitch-mean	−2.687*	0.109	−2.900*	0.417	−2.965*	−0.361	−1.905	−0.266
Pitch-sd	1.364	0.409	1.919	0.591	1.807	0.576	1.271	0.089
Intensity-mean	−0.275	−2.946*	−0.454	−2.245*	−0.229	−1.825	−0.360	−1.540
Intensity-min	0.595	−3.188*	−0.136	−4.335*	0.009	−3.317*	−0.972	−3.324*
Intensity-max	0.328	0.327	−0.731	1.081	−0.140	0.511	−0.222	0.469
Shimmer-local	**2.896***	−0.476	**3.396***	−1.941	**3.006***	−1.704	**2.794***	−0.914
Jitter-local	**3.205***	0.725	**2.796***	0.242	**2.867***	0.469	**2.973***	0.260

Table 3: Convergence t-values of paired t-tests comparing team-level partner differences ($TDiff_p$) of first 3, 5, 7 minutes vs. last 3, 5, 7 minutes, respectively, and of first vs. second game half, for each game. Positive t-values indicate convergence (i.e., that partner differences in the second interval are smaller than in the first). Negative t-values indicate divergence. Significant convergence results are in bold. * $p < .05$. $n = 62$.

ers who are all within the same team are smaller than differences between team members and other speakers in the corpus. Thus, negative values are indicative of team entrainment. The results show that the team members were significantly more similar to each other than to other speakers on intensity mean, min, and max and on shimmer for both games. Team-level entrainment on jitter was significant for only the second game.

The convergence results are shown in Table 3 for four different temporal comparison intervals. Comparison of the significant game 1 results shows that teams entrained on pitch min, pitch max, shimmer, and jitter in at least one of the intervals. Both shimmer and jitter converged for all choices of temporal units. For pitch, convergence was instead only seen using the first and last 3 minutes, which are the intervals farthest in the game from each other. The only feature that diverged during game 1 is pitch-mean. The rest of the features did not show significant team-level partner differences during game 1 for any temporal interval and thus exhibited maintenance, meaning that the team members neither converged nor diverged. During game 2, we observed maintenance for all features except for intensity-mean and intensity-min, which diverged. Together our results suggest that when teams in our corpus converged on a feature, they did so earlier in the experiment (namely, just during the first game, and sometimes just in the earliest part of the first game).

As a divergent validity check for convergence, for each of the 62 teams, we constructed artificial versions of the real conversations between team members: For each member of the team, we randomly permuted the silence and speech intervals extracted by Praat. Ideally, we should not see evidence of convergence within these constructed conversations. Our results confirm that there was no significant entrainment on either of the two constructed games, for all temporal comparison intervals and all features.

In summary, team acoustic-prosodic entrainment did not occur for all features. For the features that did show entrainment, results varied depending on whether proximity or convergence was examined, and by the time intervals compared. With respect to type of entrainment, when looking at the entire game 1, there was significant evidence of entrainment (proximity) on mean, min, max intensity, and shimmer. Although there was no significant proximity for min, max pitch and jitter, they did become more similar (converged) over time. With respect to time, team convergence was found for shimmer and jitter independently of temporal interval examined, but for pitch only when comparing the most distant temporal intervals in game 1.

5.2 Participation Equality / Dominance

Within psychology, equality of participation has been associated with successful team performance and decision-making (e.g., (Mesmer-Magnus and DeChurch, 2009; Stasser and Titus, 1987)). Within computational linguistics, balance of participation with respect to proposal of ideas was associated with more productive small group (online) conversations (Niculae and Danescu-Niculescu-Mizil, 2016).

Extending this literature, we perform a novel in-

	Model 1			Model 2		
	B	SEB	β	B	SEB	β
Session Length	0.187	0.067	0.328*	0.197	0.064	0.344*
Team Size	108.706	47.398	0.269*	69.721	47.980	0.173
Participation Dominance				−1077.747	429.130	−0.299*
Model R^2		0.186			0.266	
Model F		6.761*			7.015*	

Table 4: Summary of hierarchical regression analysis for variables predicting entrainment on pitch-max. * $p < .05$. $n = 62$.

vestigation of the association between participation equality/dominance and team entrainment, focusing on the time interval showing the most significant convergence results in Section 5.1.4 (entrainment on pitch-max, pitch-min, jitter, and shimmer from the first to last 3 minutes of game 1).

Equation 3 defines the participation of player i in a team, where $speech\_length_i$ is the sum of the lengths of the speech intervals of player i:

$$participation_i = \frac{speech\_length_i}{\sum_{m \in team} speech\_length_m} \quad (3)$$

Participation dominance in turn is the standard deviation of the participation for all team members:

$$Dominance = \sigma(Participation),$$
$$Participation = \{participation_i | i \in team\} \quad (4)$$

Higher standard deviations indicate a greater range of participation from team members, and lower standard deviations indicate more participation equality.

We performed a hierarchical regression analysis for each of the four acoustic-prosodic features noted above as the target entrainment variable. As in the convergence section, we measured entrainment as the average differences ($TDiff_p$) of the team in the first interval minus the second interval. Larger positive numbers are indicative of more entrainment. The independent variables we included in our analysis are: team size, session length, average age of the team members, percentage of the female players in each team, and participation dominance. The first four are covariates that have been found to be or are likely related to team communication and/or dynamics. We hypothesized that participation dominance would be related to entrainment above and beyond these other potential variables.

Table 4 presents the results with pitch-max for entrainment. (The other 3 entrainment variables did not show significant relationships with participation.) The standardized βs indicate the effect size and direction of the individual variables on pitch-max, whereas the R^2 indicates the effect size of the model of all the variables together. Average age and percent female were not significantly related to entrainment on pitch-max, so were excluded from the final analyses.

First, both team size and session length were entered as potential independent variables into the regression analysis with pitch-max as the dependent variable. This model (Model 1) was significant. Specifically, team size and session length were both significantly positively associated with entrainment on pitch-max. That is, as team size or session length increased, entrainment also increased.

Participation dominance was then entered to create Model 2, which included team size, session length, and participation dominance. The amount of variance explained for participation dominance was significant above and beyond the variables entered in Model 1, $\Delta R^2 = 0.08$, $\Delta F(1, 58) = 6.307$, $p = 0.015$. Specifically, there was a significant negative association between participation dominance and entrainment on pitch-max, such that greater participation equality was related to greater entrainment. This suggests that the more each team member is given a chance to equally contribute, the more likely they are to entrain on their maximum pitch.

6 Summary and Broader Implications

The long-term goal of our research is to use speech and language processing, informed by the teamwork literature, to develop computational measures of conversational team entrainment that will be useful for predicting team success. We first described the design and contents of the Teams corpus, which is being made freely available for research purposes. Experimental manipulations, high-quality

audio and video with time-aligned transcriptions, and self-reported team process data make the corpus a unique resource for studying multi-party dialogue entrainment. We provided two examples illustrating the use of the Teams corpus to facilitate new directions in the study of entrainment: quantifying acoustic-prosodic entrainment at the team rather than the dyad-level, and incorporating the teamwork construct of participation dominance into the study of entrainment. Our current plans include continued corpus development (recall Section 4.4), and using more sophisticated methods than dyad averaging (e.g., using weighting based on team process measures) to move from dyads to teams.

With respect to broader impact, our entrainment measures could be used to mine existing corpora for naturalistic successful and unsuccessful conversations, or to trigger online interventions by dialogue systems participating in multi-party conversations. After additional research understanding the important thresholds for entrainment, organizations could unobtrusively measure team effectiveness during entrainment, and intervene with training to aid teams with low entrainment. Similar interventions would be useful for conversational agents that monitor and facilitate group interactions (e.g., in education via computer-supported collaborative learning). Our work could also support the development of data mining applications for corpora such as team meetings or discussions, from classrooms to boardrooms. Finally, our corpus could support natural language processing research regarding any other aspect of teamwork (e.g., affect, conflict, topic modeling). In sum, the Teams Corpus should provide usable, multi-channel data for examining team processes for a range of purposes and research disciplines.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. 1420784 and 1420377. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors wish to thank Catharine Oertel and Mattias Heldner for advice regarding both lab and equipment needs, and the Learning Research and Development Center at the University of Pittsburgh for lab renovation. Finally, we would like to thank Anish Kumar, the Pitt NLP group, and the anonymous reviewers for their help in improving the paper.

References

Stergos Afantenos, Eric Kow, Nicholas Asher, and Jérémy Perret. 2015. Discourse parsing for multi-party chat dialogues. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 928–937.

Kelsey Allen, Giuseppe Carenini, and Raymond Ng. 2014. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1169–1180.

Daniel J. Beal, Robin. R. Cohen, Michael J. Burke, and Christy L. McLendon. 2003. Cohesion and performance in groups: A meta-analytic clarification of construct relations. *Journal of Applied Psychology*, 88:989–1004.

Štefan Beňuš, Agustín Gravano, Rivka Levitan, Sarah Ita Levitan, Laura Willson, and Julia Hirschberg. 2014. Entrainment, dominance and alliance in supreme court hearings. *Knowledge-Based Systems*, 71:3–14.

Paul Boersma and Vincent van Heuven. 2002. Praat, a system for doing phonetics by computer. *Glot international*, 5(9/10):341–345.

Stephanie A Borrie, Nichola Lubold, and Heather Pon-Barry. 2015. Disordered speech disrupts conversational entrainment: a study of acoustic-prosodic entrainment and communicative success in populations with communication challenges. *Frontiers in psychology*, 6.

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(6):1482–1493.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Iain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. 2006. The ami meeting corpus: A pre-announcement. In Steve Renals and Samy Bengio, editors, *Machine Learning for Multimodal Interaction*, volume 3869 of *Lecture Notes in Computer Science*, pages 28–39.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*, pages 699–708.

James E. Driskell, Eduardo Salas, and Sandra Hughes. 2010. Collective orientation and team performance: Development of an individual differences measure. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 52:316–328.

Heather Friedberg, Diane Litman, and Susannah B. F. Paletz. 2012. Lexical entrainment and success in student engineering groups. In *Proceedings Fourth IEEE Workshop on Spoken Language Technology (SLT)*, Miami, Florida, December.

Amy L. Gonzales, Jeffrey T. Hancock, and James W. Pennebaker. 2010. Language style matching as a predictor of social dynamics in small groups. *Communication Research*, 37:3–19.

M. E. Gregory, J. Feitosa, T. Driskell, E. Salas, and W. B. Vessey, 2013. *Developing and enhancing teamwork in organizations: Evidence-based best practices and guidelines*, chapter Designing, delivering, and evaluating team training in organizations. Jossey-Bass, San Francisco.

Richard A. Guzzo, Paul R. Yost, Richard J. Campbell, and Gregory P. Shea. 1993. Potency in groups: Articulating a construct. *British Journal of Social Psychology*, 32:87–106.

O. P. John, E. M. Donahue, and R. L. Kentle. 1991. The big five inventory-versions 4a and 54. University of California, Berkeley, Institute of Personality and Social Research. http://www.ocf.berkeley.edu/ johnlab/bfi.htm.

Chi-Chun Lee, Athanasios Katsamanis, Matthew P. Black, Brian R. Baucom, Panayiotis G. Georgiou, and Shrikanth Narayanan. 2011. An analysis of pca-based vocal entrainment measures in married couples' affective spoken interactions. In *INTERSPEECH*, pages 3101–3104.

Rivka Levitan and Julia Hirschberg. 2011. Measuring acoustic-prosodic entrainment with respect to multiple levels and dimensions. In *Interspeech*.

Rivka Levitan, Agustín Gravano, and Julia Hirschberg. 2011. Entrainment in speech preceding backchannels. In *Proceedings of ACL/HLT*, June.

Rivka Levitan, Agustín Gravano, Laura Willson, Stefan Benus, Julia Hirschberg, and Ani Nenkova. 2012. Acoustic-prosodic entrainment and social behavior. In *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 11–19.

José Lopes, Maxine Eskenazi, and Isabel Trancoso. 2013. Automated two-way entrainment to improve spoken dialog system performance. In *ICASSP*, pages 8372–8376.

Nichola Lubold and Heather Pon-Barry. 2014. Acoustic-prosodic entrainment and rapport in collaborative learning dialogues. In *Proceedings of the 2014 ACM workshop on Multimodal Learning Analytics Workshop and Grand Challenge*, pages 5–12. ACM.

Jessica R. Mesmer-Magnus and Leslie A. DeChurch. 2009. Information sharing and team performance: A meta-analysis. *Journal of Applied Psychology*, 94:535–546.

Ella Miron, Miriam Erez, and Eitan Naveh. 2004. Do personal characteristics and cultural values that promote innovation, quality, and efficiency compete or complement each other? *Journal of Organizational Behavior*, 25:175–199.

Christopher Michael Mitchell, Kristy Elizabeth Boyer, and James C. Lester. 2012. From strangers to partners: Examining convergence within a longitudinal study of task-oriented dialogue. In *SIGDIAL Conference*, pages 94–98.

Seungwhan Moon, Saloni Potdar, and Lara Martin. 2014. Identifying student leaders from mooc discussion forums through language influence. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*, pages 15–20.

Brian Mullen and Carolyn Copper. 1994. The relation between group cohesiveness and performance: An integration. *Psychological Bulletin*, 115(2):210–227.

Ani Nenkova, Agustín Gravano, and Julia Hirschberg. 2008. High frequency word entrainment in spoken dialogue. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 169–172.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2016. Conversational markers of constructive discussions. *arXiv preprint arXiv:1604.07407*.

Catharine Oertel, Kenneth A. Funes Mora, Samira Sheikhi, Jean-Marc Odobez, and Joakim Gustafson. 2014. Who will get the grant?: A multimodal corpus for the analysis of conversational behaviours in group interviews. In *Proceedings of the 2014 Workshop on Understanding and Modeling Multiparty, Multimodal Interactions*, pages 27–32.

Susanah B. F. Paletz and Christian D. Schunn. 2011. Assessing group-level participation in fluid teams: Testing a new metric. *Behavioral Research Methods*, 43:522–536.

Robert Porzel, Annika Scheffler, and Rainer Malaka. 2006. How entrainment increases dialogical effectiveness. In *Proceedings of the IUI'06 Workshop on Effective Multimodal Dialogue Interaction*, pages 35–42.

David Reitter and Johanna D. Moore. 2007. Predicting success in dialogue. In *Proceedings of the 45th Meeting of the Association of Computational Linguistics*, pages 808–815.

Eduardo Salas, Deborah DiazGranados, Cameron Klein, C. Shawn Burke, Kevin C. Stagl, Gerald F. Goodwin, and Stanley M. Halpin. 2008. Does team training improve team performance? a meta-analysis. *Human Factors*, 50:903–933.

Tanmay Sinha and Justine Cassell. 2015. Fine-grained analyses of interpersonal processes and their effect on learning. In *Artificial Intelligence in Education: 17th International Conference*, pages 781–785.

Garold Stasser and William Titus. 1987. Effects of information load and percentage of shared information on the dissemination of unshared information during group discussion. *Journal of Personality and Social Psychology*, 53:81–93.

Svetlana Stoyanchev and Amanda Stent. 2009. Lexical and syntactic priming and their impact in deployed spoken dialog systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 189–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ruth Wageman, J. Richard Hackman, and Erin Lehman. 2005. Team diagnostic survey: Development of an instrument. *Journal of Applied Behavioral Science*, 41:373–398.

Hein Wendt, Martin C. Euwema, and I. J. Hetty van Emmerik. 2009. Leadership and team cohesiveness across cultures. *The Leadership Quarterly*, 20:358–370.

Personalized Emphasis Framing for Persuasive Message Generation

Tao Ding and **Shimei Pan**
Department of Information Systems
University of Maryland, Baltimore County
{taoding01,shimei}@umbc.edu

Abstract

In this paper, we present a study on personalized emphasis framing which can be used to tailor the content of a message to enhance its appeal to different individuals. With this framework, we directly model content selection decisions based on a set of psychologically-motivated domain-independent personal traits including personality (e.g., extraversion) and basic human values (e.g., self-transcendence). We also demonstrate how the analysis results can be used in automated personalized content selection for persuasive message generation.

1 Introduction

Persuasion is an integral part of our personal and professional lives. The topic of generating persuasive messages has been investigated in different fields with varied focuses. Psychologists focus on the cognitive, social and emotional processes of a persuader and a persuadee to understand what makes a communication persuasive (Hovland et al., 1953; Petty and Cacioppo, 1986; Smith and Petty, 1996). Marketing researchers are interested in applying theories of persuasion in promoting consumer products (Szybillo and Heslin, 1973; Han and Shavitt, 1994; Campbell and Kirmani, 2000; Kirmani and Campbell, 2004; Ford, 2005; Hirsh et al., 2012). Natural Language Generation (NLG) researchers are interested in studying the relation between language usage and persuasion in order to build automated systems that produce persuasive messages (Guerini et al., 2011; Reiter et al., 2003).

It is also generally believed that persuasion is more effective when it is custom-tailored to reflect the interests and concerns of the intended audience (Noar et al., 2007; Dijkstra, 2008; Hirsh et al., 2012). A proven tailoring tactic commonly used by politicians, marketing executives, as well as public health advocators is content framing (Meyerowitz and Chaiken, 1987; Maheswaran and Meyers-Levy, 1990; Grewal et al., 1994; Rothman and Salovey, 1997). Previous framing research has mainly focused on two types of framing strategies: *emphasis framing* and *equivalence framing*. To *emphasis frame* a message is to simplify reality by focusing on a subset of the aspects of a situation or an issue and make them more prominent in a communication to promote certain definition, causal interpretation and moral evaluation (Entman and Rojecki, 1993). For example, in political debating, the topic of *nuclear energy* can be framed as an *economic development* issue, a *safety* issue or an *environmental* issue. In marketing, the same car can be framed as a *low cost* car, a *performance* car, or a *green* car. With different framing strategies, the authors try to appeal to individuals with different beliefs and concerns. In contrast, *equivalence framing* focuses on presenting content as either loss-framed or gain-framed messages. For example, a smoking cessation message can employ a gain-frame like "You will live longer if you quit smoking", or a loss-frame such as "You will die sooner if you do not quit smoking". Even though the messages are equivalent factually, the frames can influence a receiver's behavior either to encourage a desirable behavior or to avoid an unwanted outcome (Tversky and Kahneman, 1981). In this study, we focus on personalized *emphasis framing* which selectively emphasize the aspects of an entity (e.g., a car) to enhance its appeal to a given receiver.

Using emphasis framing as the framework for personalized content selection, we can take advantage of rich findings in prior framing research that link content selection decisions to a set of psychologically-motivated domain-independent personal characteristics. This has made our work more generalizable than those relying on application-specific user characteristics (e.g., use an individual's smoking habit to tailor a smoke cessation message). Since content framing is a part of the content determination process, the model we propose is a part of the *content planner* in a Natural Language Generation (NLG) system (Reiter and

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1432–1441,
Austin, Texas, November 1-5, 2016.

Dale, 1997).

There are three main contributions of this work.

1. To the best of our knowledge, this is the first effort in building an automated model of emphasis framing for personalized persuasive message generation.

2. We made content selection decisions based on a set of psychologically-motivated application-independent user traits, such as personality and basic human values, which makes our work more generalizable than those relying on domain-specific user characteristics and preferences.

3. We propose a cascade content selection model that integrates personalized content selection patterns in automated persuasive message generation.

2 Related Work

In the following, we summarize the research that is most relevant to our work including prior psychology and communication studies that link emphasis framing with personal traits. Since building computational models of emphasis framing was not the primary goal in these studies, we also include literature on personalized Natural Language Generation.

2.1 Emphasis framing and Personal Traits

There is a large body of social, marketing and communication theories on framing effects. (Zaller, 1992; Zaller and Feldman, 1992) point out that framing essentially reorganizes information to increase accessibility of an issue dimension by highlighting one cognitive path that had previously been in the dark. Others argue that the framing effect is due to a change in the rank order of the values associated with different aspects through the interaction with the content found within a message (Nelson et al., 1997; Chong and Druckman, 2007; Jacoby, 2000). The human decisions are controlled partly by the formulation of the problem and partly by the norms, habits, and personal characteristics of the decision-maker (Tversky and Kahneman, 1981).

Although most research agrees that the characteristics of a receiver play an important role in framing effectiveness, there is a significant disagreement in what characteristics of a receiver result in framing effects. For example, (Anderson, 2010) states that people with prior attitudes toward an issue can be influenced by frames, while Slothuus (Slothuus, 2008) and Tabor et al. (Taber et al., 2009) did not find a framing effect for those with strong values associated with an issue prior to exposure to the frame. The mixed results may be due to the fact that many of these studies did not take into account that people with different traits (e.g., different personality) may react to framing strategies differently.

Recently, personalized framing, especially personality-based framing research has become a hot topic. Among them, Hirsh (2012) investigates whether a persuasive appeal's effectiveness can be increased by aligning message framing with a recipient's personality profile. In this study, for a given mobile phone, they constructed five advertisements, each designed to target one of the five major personality traits. Their results indicate that advertisements were evaluated more positively when they cohered with participants' personality. In a separate study, (Conroy et al., 2012) found that certain personality traits, particularly openness, agreeableness, and conscientiousness mediate framing effects when participants were presented with different frames of political and health issues such as civil liberties, medical treatments, energy, affirmative action, and gun control.

Inspired by the above research, we also employ psychologically-motivated trait models to capture individual characteristics. In addition to personality, we also incorporate basic human values since framing effects were also shown to be related to personal beliefs and motivations. As a result, we have significantly increased the scope of our study over prior research. Moreover, unlike prior research where only messages hand-crafted by experts were used, we are interested in building computational models to automatically select a subset of the aspects to highlight based on personal traits.

2.2 Personalized NLG

There is also a large body of work on personalized Natural Language Generation (NLG). For example, STOP is a Natural Language Generation (NLG) system that generates tailored smoking cessation letters based on a user's responses to a four-page

smoking questionnaire (Reiter et al., 2003); PERSIVAL customizes the content of search summaries based on its relevance to a given patient's health record (McKeown et al., 2003); MATCH (Johnston et al., 2002) is a multimodal dialogue system that tailors the content of its responses based on a user's restaurant preferences; M-PIRO (Burenhult, 2002) tailors the words and the complexity of museum object descriptions for different audiences (e.g. adults, children, and experts); PERSONAGE (Mairesse and Walker, 2011) and CRAG 2 (Gill et al., 2012) vary linguistic styles to project intended personality in spoken utterances. In addition, Carenini and Moore (Carenini and Moore, 2006) employed a multiattribute utility theory-based user preference model for personalized evaluative argument generation. Among them, STOP, PERSIVAL and MATCH use domain-specific user models while M-PIRO, PERSONAGE and GRAG2 employ domain independent user properties, such as expertise and personality. For PERSONAGE and GRAG2, personality traits are mainly used to adapt linguistic styles. So far, there has not been much work focusing on using domain-independent user traits to automatically adapt message content to improve its persuasive appeal.

3 Acquiring Personal Traits

Since prior study often links framing effects to individual characteristics such as personality and individual motivations and beliefs, here we focus on two widely-accepted trait models in psychology: the Big5 personality model (Goldberg, 1993) and Schwartz's basic human value model (Schwartz, 2003). Figure 1 shows the description of each of the Big5 personality traits along with each of the five basic human value traits.

To acquire the personality and value traits of a person, traditionally, psychometric tests, such as the IPIP test for Big 5 personality (Yarkoni, 2010a) and the PVQ survey for values (Schwartz, 2003), were used. Recent research in the field of psycholinguistics has shown that it is possible to automatically infer personal traits from one's linguistic footprint, such as tweets, Facebook posts and blogs (Yarkoni, 2010b; Celli and Polonio, 2013; Chen et al., 2014). Unlike psychometric tests, automated trait analysis

Model	Factors	Description
Big5 Personality Model	Openness to Experience	The degree of intellectual curiosity, creativity and a preference for novelty and variety a person has.
	Conscientiousness	A tendency to be organized and dependable, show self-discipline, act dutifully, and prefer planned rather than spontaneous behavior
	Extraversion	Energy, positive emotions, assertiveness, sociability and the tendency to seek stimulation in the company of others, and talkativeness.
	Agreeableness	A tendency to be compassionate and cooperative. Also a measure of one's trusting, helpful, well-tempered nature.
	Neuroticism	The tendency to experience unpleasant emotions easily, of low emotional stability and impulse control.
Schwartz's basic human values	Openness to change	the desire for independence and new, novel, exciting and challenging experiences:
	Self-enhancement	The pursuit of self-interests including social status, prestige, resources and personal success through demonstrating competence
	Conservation	Self-restriction and conform to tradition, social expectations and norms; Pursue safety, harmony and stability; resist to change
	Hedonism	Pursuit of pleasure and sensuous gratification for oneself.
	Self-transcendence	Concern and pursue the enhancement and protection of the welfare and interests of others as well as the protection for nature.

Figure 1: Description of Two Trait Models

allows us to infer personal traits for a large number of people, which makes it possible to scale up automated personal persuasion for a very large population (e,g., millions of social media users).

4 Acquiring Author Framing Strategy

Framing effects are often subtle and may be influenced by many factors, such as the credibility of the authors, the personality of the receivers and the context of the communication. In the first study, we investigate whether it is feasible to build a personalized content selection model based on a writer's (a.k.a. an author's) content framing strategies.

To investigate this, we first randomly generated ten cars, each include eight aspects: *safety, fuel economy, quality, style, price, luxury, performance and durability*. The value of each aspect was randomly generated on a 5-point Likert scale: "1 (very bad)", "2 (bad)", "3 (average)", "4 (good)", and "5 (excellent)". We also conducted a large-scale personality and basic human value survey on Amazon Mechanical Turk (AMT). We used the 50-item IPIP survey (Goldberg, 1993) to obtain a Amazon Mechanical Turk worker (a.k.a. Turker)'s personality scores and the 21-item PVQ survey (Schwartz, 2003) to obtain his/her basic value scores. To en-

sure the quality of the data from AMT, we added two qualification criteria. A qualified Turker must (1) have submitted over 5000 tasks (2) with an acceptance rate over 95%. The survey also included several validation questions, which are pairs of questions that are paraphrases of each other. If the answers to a pair of validation questions are significantly different, the user data were excluded from our analysis. After removing invalid data, we collected the traits of 836 Turkers. The raw personality scores, ranging from 10 to 50, and raw value scores, ranging from 1 to 6, were computed directly from the survey answers. The normalized trait scores, ranging from 0 to 1, were computed using their rank percentiles in this population.

In addition, we designed two Human Intelligence Tasks (HITs) on AMT: a content customization task and a validation task. In the content customization task (a.k.a. Task 1), a Turker was asked to select one car aspect to emphasize in his message for a receiver. The validation task (a.k.a. Task 2) was used to validate whether a receiver prefers the message customized for her or not.

Specifically, in Task 1, the Turkers were asked to imagine that they work for a marketing firm on a campaign to promote a new car. Each Turker was given the specification of a car (randomly selected from the 10 randomly generated cars) and a receiver (randomly selected from the 836 Turkers whose trait scores were known to us). The Turker was asked to write a campaign message to persuade the receiver to buy the car. But the Turker can only select one of the eight car aspects to include in his message. Since customizing a message based on an interaction of all ten traits can be very challenging for a Turker, we used a simplified trait profile in our study. The simplified trait profile contains only two traits: the most prominent personality trait and the most prominent value trait. The prominence of a trait was defined based on the normalized trait score. The more different a trait score is from the median (.50), the more prominent the trait is. For comparison, for the same car, we also asked the same writer to select a car aspect for someone who has an opposite trait profile. The opposite trait profile is defined as the one that is most different from the given trait profile (with the lowest cosine similarity). After the writer selected a car aspect, he also wrote a campaign message using the selected aspect. Overall, after removing invalid data, we collected 490 pairs of messages for 131 pairs of receivers.

To validate the framing effect, in Task 2, we asked a new set of Turkers (receivers) to first complete an IPIP personality survey and a PVQ human value survey. Based on the survey results, we computed the trait profile for each of them. In addition, for each receiver in Task 2, we matched his/her trait profile with the 131 pairs of trait profiles collected in Task 1. The profile with the highest matching score (computed based on cosine similarity) was selected and its associated message pair was retrieved.

Then we presented the receiver with a pair of messages, one created for someone with matching trait profile, the other for someone with the opposite trait profile. We also randomized the order these messages were presented. Finally, we asked the receivers to rate which message they prefer more. If the framing strategies used by the Turkers (authors) in Task 1 were effective, then the Turkers (receivers) in Task 2 will prefer the messages tailored for them more than the ones tailored for someone with the opposite trait profile. Overall, after filtering out invalid data, we have collected the results from 145 receivers. Among them, 77 prefer the messages written for them, while 68 prefer the messages written for someone with the opposite trait profiles. We performed a *sign test* to determine whether the difference is statistically significant and the result was negative ($p < 0.2$).

Although moderate personalization effects were found in previous framing research, only expert-crafted messages were used (Hirsh et al., 2012). Here, when Turkers (mostly non-experts) were asked to customize the messages based on a receiver's traits, no significant effects were found. Since authors' emphasis framing strategies were not effective, we can not directly use authors' data to learn their emphasis framing strategies. Next, we present several experiments designed to automatically derive emphasis framing strategies based on a receiver's traits and his/her aspect selection decisions.

5 Learning Emphasis Framing Strategies

To derive emphasis framing patterns based on a receiver's traits and his/her aspect selection decisions, we designed another HIT (Task 3) on AMT to collect data. In Task 3, each Turker was asked to take the IPIP and PVQ surveys so that we can obtain his/her Big5 personality and value scores. In addition, we also asked him/her to rank all eight car aspects based on their importance to him/her. To control the influence of the value of a car aspect on a user's aspect selection decision (e.g., if the value of "safety" is "poor" and the value of "fuel economy" is "good", to promote the car, people almost always describe it as "a car with good fuel economy", not "an unsafe car", regardless of a receiver's personality). In this study, we kept the values of all car aspects unspecified. After removing invalid data, our dataset has 594 responses, each contains a Turker's personality and value scores as well as his/her rank of the eight car aspects. In the following, we describe how we analyze the relationship between aspect rank and personal traits.

5.1 Pattern Discovery with Regression

In our first study, we employed regression analysis to identify significant correlations between personal traits and aspect ranks. Specifically, we trained eight linear regression models, one for each of the eight car aspects. The dependent variable in each model is the rank of an aspect (from 1 to 8) and the independent variables are the ten user traits. In the regression analysis, we only focused on the main effects since a full interaction model with ten traits will require much more data to train. Since the raw scores of the personality and value traits use different scales, we normalized these scores so that they are all from 0 to 1. Table 1 shows the regression results.

Several interesting patterns were discovered in this analysis: (a) a positive correlation between the rank of "luxury" and "self-enhancement", a trait often associated with people who pursue self-interests and value social status, prestige and personal success ($p < 0.0001$). This pattern suggests that to promote a car to someone who scores high on "self-enhancement", we need to highlight the "luxury" aspect of a car. (b) the rank of "safety" is positively correlated with "conservation", a trait associated with people who conform to tradition and pursue safety, harmony, and stability ($p < 0.005$). This result suggests that for someone values "conservation", it is better to emphasize "car safety" in a personalized sales message. (c) "self-transcendence", a trait often associated with people who pursue the protection of the welfare of others and the nature, is positively correlated with the rank of "fuel economy" ($p < 0.005$) but negatively correlated with the rank of "style" ($p < 0.005$). This suggests that for someone who values "self-transcendence", it is better to emphasize "fuel economy", but not so much on "style". Other significant correlations uncovered in this analysis include a negative correlation between car "price" and "conservation" ($p < 0.005$), a negative correlation between car "safety" and "conscientiousness" ($p < 0.05$), and a positive correlation between "openness to change" and car "performance" ($p < 0.05$).

5.2 Pattern Discovery with Constrained Clustering

In the regression analysis, we only considered the main framing effects. In order to discover high-order interaction patterns with limited data, we want to use clustering to group people with similar traits together. In addition, we also want that the people in a cluster share similar aspect preferences. Otherwise, we won't be able to link the trait patterns discovered in a cluster with specific aspect preferences. Thus, we employed constrained clustering in this analysis. With constrained clustering, we can ensure the homogeneity of the aspect preferences within each resulting cluster.

To facilitate this analysis, first we mapped the aspect ranks obtained in Task 3 into discrete categories. For a complete rank of eight car aspects, we mapped the top three ranked aspects to an "Important" class, bottom three to a "Not-Important" class, and the middle two to a "Neutral" class. In addition, we encoded the aspect homogeneity requirement as constraints. Typically, constrained clustering incorporates either a set of must-link constraints, a set of cannot-link constraints, or both. A must-link constraint is used to specify that the two data instances in the must-link relation should be placed in the same cluster. A cannot-link constraint is used

Table 1: Results of the Regression Analysis

	Safety	Fuel	Quality	Style	Price	Luxury	Perf	Durab
Agreeableness	0.39	-0.52	-0.53	0.54	0.81	0.004	-0.62	-0.27
Conscientiousness	-1.75 *	-0.31	0.80	0.29	-0.01	0.27	0.83	-0.12
Extroversion	0.69	-0.71	0.008	-0.25	-0.37	0.48	-0.07	0.224
Neurotisim	1.08	-0.01	-0.46	-0.11	-0.32	-0.07	0.18	-0.28
Openness	1.59	-0.05	0.01	-0.99	0.36	-0.53	-0.46	0.07
Conservation	1.99 **	-0.99	-0.66	0.84	-1.72 **	0.21	0.38	-0.03
Hedonism	1.47	-0.15	-0.69	0.16	0.51	-0.06	-0.82	-0.43
Openness to change	-2.15	0.08	0.58	0.48	-1.99 *	-0.38	2.29*	1.07
Self-enhancement	-1.39	-1.12	0.58	0.47	-0.31	2.41 ***	0.77	-1.41
Self-transcendence	1.33	2.37 **	1.36	-2.47 **	-0.91	-1.01	-0.33	-0.32

Note: $p < 0.05$, ** $p < 0.005$, *** $p < 0.0001$

Table 2: Patterns Discovered in Clustering Analysis

Feature	Cluster	Accuracy	Label	Significant traits
Safety	1	0.7	Important	Extrave(+),Neuroti(+)
	2	0.64	Neutral	Conscie(+),Hedonis(+),Open(+),Self-en(+)
	3	0.71	Important	Conscie(+),Open(-)
Fuel	1	0.54	Neutral	Open(-),Self-en(-)
	2	0.54	Not-Important	Hedonis(+),Open(+),Self-en(+)
Quality	1	0.43	Important	Extrave(+),Neuroti(+)
	2	0.45	Non-Important	Hedonis(+),Open(+),Self-en(+)
	3	0.45	Not-Important	Conscie(+),Open(-)
Style	1	0.5	Not-Important	Hedonis(-),Open(-)
	2	0.55	Neutral	Conscie(+),Extrave(+),Neuroti(+)
	3	0.62	Neutral	Conscie(+),Hedonis(+),Open(+),Self-en(+)
	4	0.74	Not-Important	Conscie(+),Open(-)
Performance	1	0.73	Neutral	Extrave(+),Neuroti(+)
	2	0.5	Neutral	Conscie(+),Open(-)
	3	0.4	Not-Important	Hedonis(-),Open(-)
Durability	1	0.56	Not-Important	Extrave(+),Hedonis(+),Self-en(+)
	2	0.36	Important	Conscie(+),Hedonis(+),Open(+),Self-en(+)

Note: $CV < 0.12$ $P < 0.001$ $Diff > 0.2$

to specify that the two instances in the cannot-link relation should not be put in the same cluster. These constraints act as a guide for which a constrained clustering algorithm will use to find clusters that satisfy these requirements.

To encode the homogeneity constraint, for each car aspect (e.g. safety), we can simply add must-links between every pair of Turkers if they share the same aspect preference (e.g., both consider "safety" important) and add cannot-links for every pair of Turkers who do not share the same aspect preferences (e.g., one Turker considers "safety" "Important", the other considers it "Not-Important"). But with both must-links and cannot-links, it is very likely we will get three big clusters, each is related to one of the three categories: Important, Neutral and Not-Important. Although the resulting clusters satisfy the aspect preference homogeneity requirement, they fail to group people with similar traits together. As a result, in this analysis, we only used cannot-links, which not only guarantees the homogeneity of aspect preferences, but also creates smaller clusters that group people with similar traits together.

We employed the Metric Pairwise Constrained KMeans algorithm (MPCK-MEANS) (Bilenko et al., 2004) to incorporate the aspect preference homogeneity requirement. The optimal cluster number K was determined empirically by running MPCK-MEANS with different Ks, $K \in [3, 20]$ (3 is the minimum number of clusters since we have 3 different aspect preference categories).

To determine whether the resulting clusters capture any interesting patterns, we used two pattern selection criteria (a) a homogeneity criterion which requires that there is at least one trait whose values in the cluster is relatively homogeneous; (b) a distinctiveness criterion which requires that for the traits identified in (a), their cluster means need to be significantly different from the population means. For (a), we used the coefficient of variation (CV) as the homogeneity measure. CV, also known as relative standard deviation (RSD), is a standardized measure of dispersion of a probability or count distribution. It is often expressed as a percentage and is defined as the ratio of the standard deviation σ to the mean $|\mu|$. In the study, we required that all the CVs of

homogeneous traits to be lower than 0.12. For (b) we required that the differences of the means need to be significant based on an independent sample t-test with $p < 0.001$ and the difference of means is greater than 0.2.

Table 2 highlights some of the patterns discovered using this approach. In this table, we list the cluster id, cluster label (Important, Not-Important, Neutral), clustering accuracy, and significant traits in the cluster ("+" indicates that the cluster mean is higher than population mean, "-" means the opposite). For example, based on the Safety-1 pattern, people who are more extraverted (extrave (+)) and more neurotic (neurotic (+)) tend to consider "car safety" important. Similarly, based on pattern Safety-3, people who are more conscientious (conscie(+)) but less open (open(-)) tend to consider "safety" important. Other interesting patterns include: people who are less open (open(-)) and do not value hedonism (hedomis (-)) don't consider performance very important (performance-3), and people who are more extraverted (extrave(+)), value hedonism and self-enhancement (hedonis(+), self-en(+)) do not think durability important (durability-1).

6 Apply Emphasis Framing in NLG

The patterns derived in the previous section can be used in personalized content selection for Natural Language Generation. In general, to promote a car, people tend to highlight the good aspects and avoid the bad aspects, regardless of a receiver's personality. For example, people will likely to highlight the fuel economy aspect if a car is very fuel efficient while de-highlight the same aspect if a car is not fuel efficient. Thus, during content selection, to take the value of an aspect into consideration, we employ a cascade NLG model that integrates value-based content selection with trait-based personalization.

The input to the cascade content selection model includes: (1) the values of all the car aspects; (2) the trait scores of a receiver; (3) the eight linear-regression models learned in Section 5.2, one for each aspect; (4) the interaction rules learned in Section 5.3; (5) *n*, the number of aspects needed in the output; (6) the value difference threshold δ_1 that determines whether the values of two or more aspects are significantly different; (7) the rank difference threshold δ_2 that determines whether the ranks of two or more aspects predicted by the linear regression models are significantly different.

To select *n* aspects to emphasize, our system first ranks all the aspects based on their values. If the value of the n-th aspect v_n is significantly better than that of the (n+1)-th aspect v_{n+1} (i.e., their difference is greater than δ_1), we output the top n aspects directly. Otherwise, for all the aspects whose values are either the same or not significantly worse than v_n, their ranks will be determined by the trait-based linear regression models. Moreover, after re-ranking relevant aspects based on the predicted ranks from the regression models, if the predicted rank of the n-th aspect r_n is significantly better than that of the (n+1)-th aspect r_{n+1} (i.e., the rank difference is greater than δ_2), we just output the top n aspects in this list. Otherwise, for those aspects whose ranks predicted by the linear regression models are not significantly lower than r_n, we use the interaction rules discovered in Section 5.3 to further adjust their ranking scores (i.e., increase the rank by δ_2 if the cluster label is "Important", or decrease by δ_2 if "Not-Important"). For each aspect, if more than one interaction rule applies, more accurate rules take precedence over less accurate rules. Finally, the system will output the top n aspects in the final list.

We use an example shown in Figure 2 to illustrate the cascade aspect selection process. In this example, we assume n=3, δ_1=1 and δ_2=0.5. We first sorted all the aspects based on their values. Since the values of "Fuel Economy" and "Luxury" are significantly better than the 3rd-ranked aspect "Price", their ranks are not affected by personalized aspect selection. Similarly, since the values of "Performance" and "Style" are significantly lower than that of the 3rd-ranked aspect, their ranks are also not affected by personalization. Since the value differences among the rest 4 aspects, "Price", "Durability", "Quality" and "Safety" are all equal or not significant worse than v_3, we used trait-based personalized ranks predicted by the regression models to re-rank them (the output ranks from the regression models are shown in the parentheses in the column "Regression-based Re-Ranking"). After re-ranking these aspects based on the predicted ranks, since the rank of the 3rd-ranked aspect "Price" (2.2) and that of "Safety" (2.5) is within δ_2, we use the learned

Aspect	Value	Regression-based Re-Ranking	Interaction Rule-based Re-Ranking	Final Rank
Fuel Economy	5(Excellent)	**1**	**1**	1*
Luxury	5(Excellent)	**2**	**2**	2*
Price	3(Average)	3(2.2)	**4(2.2)**	4
Durability	3(Average)	6(6.7)	6(6.7)	6
Quality	3(Average)	5(4.5)	5(4.5)	5
Safety	3(Average)	4(2.5)	3(2.0)	3*
Performance	1(Very Bad)	7	7	7
Style	1(Very Bad)	8	8	8

Figure 2: A Cascade Content Selection Example

interaction rules to adjust their ranks. Since the predicted ranks of "Durability" and "Quality" are much worse than that of "Price", their ranks are not affected by the interaction rules. To apply the interaction rules, assume for a given receiver, both his extraversion and neuroticism scores are much higher than the population average, the Safety-1, Quality-1 and Performance-1 rules are applicable. Since the Safety-1 rule predicts that "Safety" is "Important" to the receiver while none of the rules affects "Price", the predicted rank for "Safety" is increased by δ_2 . After this adjustment, the ranks of all the aspects are shown in the "Final Rank" column. The top 3 aspects based on the final ranks are selected as the output (those marked with a *).

To evaluate the performance of the cascade content selection model, we conducted an additional AMT study. Given the specifications of the ten cars in Task1, we asked each AMT participant to select the top-n aspects to emphasize. Here n=1 and 3. In this task, aspect selection not only depends on the importance of an aspect to a receiver, but also the values of the aspects of a given car. We also acquired the personality and value scores of each Turker based on the IPIP personality and PVQ value survey. Finally, we compared the output of our model with the aspects selected by the Turkers. We used top-n overlapping percentage as the evaluation metrics. Overall, we collected the aspect selection results from 38 Turkers, each on ten different cars. In total, we collected 380 data instances in our ground truth dataset. We have tested different $\delta 1$ and $\delta 2$, the best results were obtained when $\delta 1 = 0$ and $\delta 2 = 0.5$. We compared our model with a baseline system which relies solely on the values of aspects to determine their ranks in the baseline system. If two or more aspects have the same value (e.g., the values of both "Price" and "Durability" are "3(Average)", their ranks were determined randomly. Table 3 shows the evaluation results. If only 1 aspect is needed in the output, the Top-1 agreement is 62% for the cascade model versus the baseline's 54%. Similarly, if 3 aspects are needed in the output, the Top-3 agreement is 87% for the cascade model versus the baseline's 46%. All the differences are statistically significant based on paired-t test ($p \leq 0.05$).

Table 3: Cascade Content Selection Evaluation

	Cascade	Baseline
Top-1 agreement	0.62	0.54
Top-3 agreement	0.87	0.46

7 Discussion

In general, there are two main challenges in adapting a personalized content selection model trained in one domain to another domain: (1) adapting the *data model* from one domain (e.g., restaurant data) to another (e.g., movie data); (2) adapting a domain-specific *user model* (e.g., a user' preferences of restaurant features such as "cuisine type") to a different domain (e.g., a user's preferences of movie features such as "movie genre"). Although our data model is in the automobile domain, we adopted a domain-independent user model motivated by psychological theories(e.g., personality and basic human values), instead of a domain-dependent user preference models (e.g, a user's preferences of "fuel economy"). This allows us to more easily apply typical domain adaptation methods such as instance-based (Zadrozny, 2004) or feature-based transfer learning (Blitzer et al., 2006) to further adapt the system and generalize the current results.

8 Conclusions

In this study, we analyzed the relationship between an individual's traits and his/her aspect framing decisions. Our analysis has uncovered interesting patterns that can be used to automatically customize a message's content to enhance its appeal to its receivers. We also proposed a cascade content selection model to automatically incorporate the analysis results in automated persuasive message generation. Our evaluation results have demonstrated the effectiveness of this approach.

References

Kristen D Anderson. 2010. Framing traits: The role of personality in framing effects.

Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

G Burenhult. 2002. Generating multilingual personalized descriptions of museum exhibits-the m-piro project.

Margaret C Campbell and Amna Kirmani. 2000. Consumers' use of persuasion knowledge: The effects of accessibility and cognitive capacity on perceptions of an influence agent. *journal of Consumer Research*, 27(1):69–83.

Sandra Carberry, Jennifer Chu-Carroll, and Stephanie Elzer. 1999. Constructing and utilizing a model of user preferences in collaborative consultation dialogues. *Computational Intelligence*, 15(3):185–217.

Giuseppe Carenini and Johanna D Moore. 2006. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952.

Fabio Celli and Luca Polonio. 2013. Relationships between personality and interactions in facebook. *Social Networking: Recent Trends, Emerging Issues and Future Outlook*, pages 41–54.

Jilin Chen, Gary Hsieh, Jalal U Mahmud, and Jeffrey Nichols. 2014. Understanding individuals' personal values from social media word use. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 405–414. ACM.

Dennis Chong and James N Druckman. 2007. Framing theory. *Annu. Rev. Polit. Sci.*, 10:103–126.

Susan Conroy, Carmine M Pariante, Maureen N Marks, Helen A Davies, Simone Farrelly, Robin Schacht, and Paul Moran. 2012. Maternal psychopathology and infant development at 18 months: the impact of maternal personality disorder and depression. *Journal of the American Academy of Child & Adolescent Psychiatry*, 51(1):51–61.

Arie Dijkstra. 2008. The psychology of tailoring-ingredients in computer-tailored persuasion. *Social and personality psychology compass*, 2(2):765–784.

Robert M Entman and Andrew Rojecki. 1993. Freezing out the public: Elite and media framing of the us anti-nuclear movement.

Christopher M Ford. 2005. Speak no evil: targeting a population's neutrality to defeat an insurgency. *Parameters: The US Army War College Quarterly, Summer*.

Alastair J Gill, Carsten Brockmann, and Jon Oberlander. 2012. Perceptions of alignment and personality in generated dialogue. In *Proceedings of the Seventh International Natural Language Generation Conference*, pages 40–48. Association for Computational Linguistics.

Lewis R. Goldberg. 1993. The structure of phenotypic personality traits. *American Psychologist*, 48(1):26.

Dhruv Grewal, Jerry Gotlieb, and Howard Marmorstein. 1994. The moderating effects of message framing and source credibility on the price-perceived risk relationship. *Journal of consumer research*, pages 145–153.

Marco Guerini, Oliviero Stock, Massimo Zancanaro, Daniel J OKeefe, Irene Mazzotta, Fiorella de Rosis, Isabella Poggi, Meiyii Y Lim, and Ruth Aylett. 2011. Approaches to verbal persuasion in intelligent user interfaces. In *Emotion-Oriented Systems*, pages 559–584. Springer.

Sang-Pil Han and Sharon Shavitt. 1994. Persuasion and culture: Advertising appeals in individualistic and collectivistic societies. *Journal of experimental social psychology*, 30(4):326–350.

Jacob B Hirsh, Sonia K Kang, and Galen V Bodenhausen. 2012. Personalized persuasion tailoring persuasive appeals to recipients personality traits. *Psychological science*, 23(6):578–581.

Carl I Hovland, Irving L Janis, and Harold H Kelley. 1953. Communication and persuasion; psychological studies of opinion change.

William G Jacoby. 2000. Issue framing and public opinion on government spending. *American Journal of Political Science*, pages 750–767.

Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. 2002. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics.

Amna Kirmani and Margaret C Campbell. 2004. Goal seeker and persuasion sentry: How consumer targets respond to interpersonal marketing persuasion. *Journal of Consumer Research*, 31(3):573–582.

Durairaj Maheswaran and Joan Meyers-Levy. 1990. The influence of message framing and issue involvement. *Journal of Marketing research*, pages 361–367.

François Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Comput. Linguist.*, 37(3):455–488, September.

Kathleen R McKeown, Noemie Elhadad, and Vasileios Hatzivassiloglou. 2003. Leveraging a common representation for personalized search and summarization in a medical digital library. In *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, pages 159–170. IEEE Computer Society.

Beth E Meyerowitz and Shelly Chaiken. 1987. The effect of message framing on breast self-examination attitudes, intentions, and behavior. *Journal of personality and social psychology*, 52(3):500.

Thomas E Nelson, Zoe M Oxley, and Rosalee A Clawson. 1997. Toward a psychology of framing effects. *Political behavior*, 19(3):221–246.

Seth M Noar, Christina N Benac, and Melissa S Harris. 2007. Does tailoring matter? meta-analytic review of tailored print health behavior change interventions. *Psychological bulletin*, 133(4):673.

Richard E Petty and John T Cacioppo. 1986. *The elaboration likelihood model of persuasion*. Springer.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(01):57–87.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1):41–58.

Alexander J Rothman and Peter Salovey. 1997. Shaping perceptions to motivate healthy behavior: the role of message framing. *Psychological bulletin*, 121(1):3.

Shalom H Schwartz. 2003. A proposal for measuring value orientations across nations. *Questionnaire Package of the European Social Survey*, pages 259–290.

Rune Slothuus. 2008. More than weighting cognitive importance: A dual-process model of issue framing effects. *Political Psychology*, 29(1):1–28.

Stephen M Smith and Richard E Petty. 1996. Message framing and persuasion: A message processing analysis. *Personality and Social Psychology Bulletin*, 22:257–268.

George J Szybillo and Richard Heslin. 1973. Resistance to persuasion: Inoculation theory in a marketing context. *Journal of Marketing Research*, pages 396–403.

Charles S Taber, Damon Cann, and Simona Kucsova. 2009. The motivated processing of political arguments. *Political Behavior*, 31(2):137–155.

Amos Tversky and Daniel Kahneman. 1981. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458.

Tal Yarkoni. 2010a. The abbreviation of personality, or how to measure 200 personality scales with 200 items. *Journal of Research in Personality*, 44(2):180–198.

Tal Yarkoni. 2010b. Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers. *Journal of research in personality*, 44(3):363–373.

Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *ICML*.

John Zaller and Stanley Feldman. 1992. A simple theory of the survey response: Answering questions versus revealing preferences. *American journal of political science*, pages 579–616.

John Zaller. 1992. *The nature and origins of mass opinion*. Cambridge university press.

Cross-Sentence Inference for Process Knowledge

Samuel Louvan[+], Chetan Naik[+], Sadhana Kumaravel[+], Heeyoung Kwon[+], Niranjan Balasubramanian[+], Peter Clark[*]

[+]Stony Brook University, [*]Allen Institute for AI,
{slouvan, cnaik, skumaravel, heekwon,niranjan}@cs.stonybrook.edu,
peterc@allenai.org

Abstract

For AI systems to reason about real world situations, they need to recognize which processes are at play and which entities play key roles in them. Our goal is to extract this kind of role-based knowledge about processes, from multiple sentence-level descriptions. This knowledge is hard to acquire; while semantic role labeling (SRL) systems can extract sentence level role information about individual mentions of a process, their results are often noisy and they do not attempt create a globally consistent characterization of a process.

To overcome this, we extend standard *within sentence* joint inference to inference across multiple sentences. This *cross sentence* inference promotes role assignments that are compatible across different descriptions of the same process. When formulated as an Integer Linear Program, this leads to improvements over within-sentence inference by nearly 3% in F1. The resulting role-based knowledge is of high quality (with a F1 of nearly 82).

1 Introduction

Knowledge about processes is essential for AI systems in order to understand and reason about the world. At the simplest level, even knowing which class of entities play key roles can be useful for tasks involving recognition and reasoning about processes. For instance, given a description "a puddle drying in the sun", one can recognize this as an instance of the process *evaporation* using a macro-level role knowledge: Among others, the typical *undergoer* of evaporation is a kind of liquid (the puddle), and the *enabler* is usually a heat source (the sun).

1) Evaporation is the process by which liquids are converted to their gaseous forms.
2) Evaporation is the process by which water is converted into water vapor.
3) Water vapor rises from water due to evaporation.
4) Clouds arise as water evaporates in the sun.

Table 1: Example sentences for the process *evaporation*. Underlined spans correspond to fillers for the *undergoer* role.

Our goal is to acquire this kind of role-based knowledge about processes from sentence-level descriptions in grade level texts. Semantic role labeling (SRL) systems can be trained to identify these process specific roles. However, these were developed for sentence-level interpretation and only ensure *within sentence* consistency of labels (Punyakanok et al., 2004; Toutanova et al., 2005; Lewis et al., 2015), limiting their ability to generate coherent characterizations of the process overall. In particular, the same process participant may appear in text at different syntactic positions, with different wording, and with different verbs, which makes it hard to extract globally consistent descriptions. In this work, we propose a cross sentence inference method to address this problem.

To illustrate the challenge consider some example sentences on *evaporation* shown in Table 1.The underlined spans correspond to fillers for an *undergoer* role i.e., the main entity that is undergoing evaporation. However, the filler *water* occurs as different syntactic arguments with different main actions. Without large amounts of process-specific training data, a supervised classifier will not able to

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1442–1451,
Austin, Texas, November 1-5, 2016.

learn these variations reliably. Nevertheless, since all these sentences are describing *evaporation*, it is highly likely that *water* plays a single role. This expectation can be encoded as a factor during inference to promote consistency and improve accuracy, and is the basis of our approach.

We formalize this *cross sentence* joint inference idea as an Integer Linear Program (ILP). Our central idea is to collect all sentences for a single process, generate candidate arguments, and assign roles that are globally consistent for all arguments within the process. This requires a notion of consistency, which we model as pairwise alignment of arguments that should receive the same label. Argument-level entailment alone turns out to be ineffective for this purpose.

Therefore, we develop an alignment classifier that uses the compatibility of contexts in which the candidate arguments are embedded. We transform the original role-label training data to create alignment pairs from arguments that get assigned the same label, thus avoiding the need for additional labeling. Finally, the ILP combines the output of the SRL classifier and the alignment classifier in an objective function in order to find globally consistent assignments.

An empirical evaluation on a process dataset shows that proposed *cross sentence* formulation outperforms a strong *within sentence* joint inference baseline, which uses scores from a custom built role classifier that is better suited for the target domain.

In summary, this work makes the following contributions:

1. A cross-sentence, collective role-labeling and alignment method for harvesting process knowledge.

2. A high quality semantic resource that provides knowledge about scientific processes discussed in grade-level texts including physical, biological, and natural processes.

3. An evaluation which shows that the proposed cross sentence inference yields high quality process knowledge.

2 Related Work

Role-based representations have been shown to be useful for Open-domain factoid question answering (Shen and Lapata, 2007; Pizzato and Mollá, 2008), grade-level science exams (Jauhar et al., 2016) , and comprehension questions on process descriptions (Berant et al., 2014). Similar to process comprehension work, we target semantic representations about processes but we focus only on a high-level summary of the process, rather than detailed sequential representation of sub-events involved. Moreover, we seek to aggregate knowledge from multiple descriptions rather than understand a single discourse about each process.

There has been substantial prior work on semantic role labeling itself, that we leverage in this work. First, there are several systems trained on the PropBank dataset, e.g., EasySRL (Lewis et al., 2015), Mate (Björkelund et al., 2009), Generalized-Inference (Punyakanok et al., 2004). Although useful, the PropBank roles are verb (predicate) specific, and thus do not produce consistent labels for a *process* (that may be expressed using several different verbs). In constrast, frame-semantic parsers, e.g., SEMAFOR (Das et al., 2010), trained on FrameNet-annotated data (Baker et al., 1998) do produce concept (frame)-specific labels, but the FrameNet training data has poor ($< 50\%$) coverage of the grade science process terms. Building a resource like FrameNet for a list of scientific processes is expensive.

Several unsupervised, and semi-supervised approaches have been proposed to address these issues for PropBank style predicate-specific roles (Swier and Stevenson, 2004; Lang and Lapata, 2011; Fürstenau and Lapata, 2009; Fürstenau and Lapata, 2012; Lang and Lapata, 2010; Klementiev, 2012). A key idea here is to cluster syntactic signatures of the arguments and use the discovered clusters as roles. Another line of research has sought to perform joint training for syntactic parsing and semantic role labeling (Lewis et al., 2015), and in using PropBank role labels to improve FrameNet processing using pivot features (Kshirsagar et al., 2015).

Some SRL methods account for context information from multiple sentences (Ruppenhofer et al., 2010; Roth and Lapata, 2015). They focus on an-

Process	Undergoer	Enabler	Action	Result
evaporation	liquid water	heat heat energy	changes convert	gas water vapor
weathering	rock solid material	weather heating	disintegration breaking down	smaller rocks smaller particles
photosynthesis	carbon dioxide CO2	solar energy light energy	convert transforms	energy food

Table 2: Examples of Target Knowledge Roles

notating individual event mentions in a document using discourse-level evidence such as co-reference chains. Our task is to aggregate knowledge about processes from multiple sentences in different documents. Although both tasks require raw SRL-style input, the different nature of the process task means that a different solution framework is needed.

Our goal is to acquire high quality semantic role based knowledge about processes. This allows us an unique opportunity to jointly interpret sentences that are discussing the same process. We build on ideas from previous within sentence joint inference (Punyakanok et al., 2004), argument similarity notions in semi and unsupervised approaches (Fürstenau and Lapata, 2012), and combining PropBank roles to propose a cross-sentence inference technique (Kshirsagar et al., 2015). The inference can be integrated with existing trained supervised learning pipelines, which can provide a score for role assignments for a given span.

3 Approach

Processes are complex events with many participating entities and inter-related sub-events. In this work, we aim for a relatively simple macro-level role-based knowledge about processes. Our task is to find classes of entities that are likely to fill key roles within a process namely, the *undergoer*, *enabler*, *result*, and *action*[1] (different verbs denoting the main action when the process is occurring, e.g., "dry"). We select these roles based on an initial analysis of grade science questions that involve recognizing instances of processes from their descriptions. Table 2 shows some examples of the target knowledge roles.

[1]For simplicity, we abuse the notion of a role to also include the main action as a role.

We develop a scalable pipeline for gathering such role-based process knowledge. The input to our system is the name of a process, e.g., "evaporate". Then we use a set of query patterns to find sentences that describe the process. A semantic role classifier then identifies the target roles in these sentences. The output is a list of typical fillers for the four process roles.

This setting presents a unique opportunity, where the goal is to perform semantic role labeling on a set of closely related sentences, sentences that describe the same process. This allows us to design a joint inference method that can promote expectations of consistency amongst the extracted role fillers.

There is no large scale training data that can be readily used for this task. Because we target process-specific and not verb-specific semantic roles, existing ProbBank (Kingsbury and Palmer, 2003) trained SRL systems cannot be used directly. Frame-semantic parsers trained on FrameNet data (Baker et al., 1998) are also not directly usable because FrameNet lacks coverage for many of the processes discussed in the science domain. Therefore, we create a process dataset that covers a relatively small number of processes, but demonstrate that the role extraction generalizes to previously unseen processes as well.

3.1 Cross-Sentence Inference

Given a set of sentences about a process, we want to extract role fillers that are globally consistent i.e., we want role assignments that are compatible. Our approach is based on two observations: First, any given role is likely to have similar fillers for a particular process. For instance, the undergoers of the evaporation process are likely to be similar – they are usually liquids. Second, similar arguments are

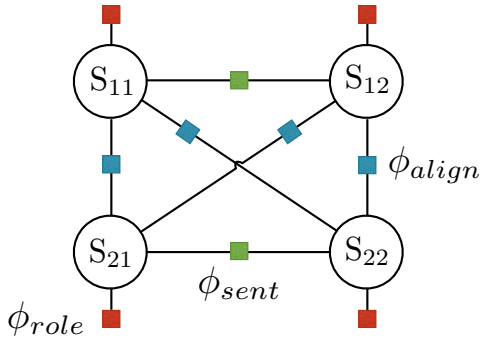


Figure 1: A factor graph representation of cross sentence inference. S_{11} and S_{12} denote role assignments for arguments a_{11} and a_{12} in one sentence, and S_{21} and S_{22} denote for arguments a_{21} and a_{22} in another. The ϕ_{role} factors score each role assignment to the arguments, and the ϕ_{align} factors score the compatibility of the connected arguments. ϕ_{sent} factors encode sentence level constraints.

unlikely to fill different roles for the same process. In evaporation, for example, it is highly unlikely that *water* is an undergoer in one sentence but is a result in another. These role-specific selectional preferences vary for each process and can be learned if there are enough example role fillers for each process during training (Zapirain et al., 2009; Zapirain et al., 2013). Since, we wish to handle processes for which we have no training data, we approximate this by modeling whether two arguments should receive the same role given their similarity and their context similarity.

Figure 1 illustrates the cross sentence inference problem using a factor graph. The S_{ij} random variables denote the role label assignment for the j^{th} argument in sentence i. Each assignment to an argument S_{ij} is scored by a combination of the role classifier's score (factor ϕ_{role}), and its pairwise compatibility with the assignments to other arguments (factor ϕ_{align}). The factors ϕ_{sent} capture two basic within sentence constraints.

3.2 Inference using ILP

We formulate the cross sentence inference task using an Integer Linear Program shown in Figure 2. The ILP seeks an assignment that maximizes a combination of individual role assignment scores and their global compatibility, which is measured as the similarity of fillers for the same role minus similarity of

$$\arg\max_{\mathbf{z}} \sum_k \sum_{i,j} z_{ijk} \Bigg(\lambda \underbrace{\phi_{role}(a_{ij}, k)}_{\text{Role classifier score}} + (1-\lambda) \underbrace{\Big[\Delta(a_{ij}, k) - \nabla(a_{ij}, k) \Big]}_{\text{Global compatibility}} \Bigg)$$

where compatibility with same roles is:

$$\Delta(a_{ij}, k) = \frac{1}{\tilde{N}_k} \sum_{l,m} z_{lmk} \phi_{align}(a_{ij}, a_{lm})$$

and compatibility with other roles is:

$$\nabla(a_{ij}) = \frac{2}{\tilde{N}_{k'}} \sum_{l,m} \sum_{n \neq k} z_{lmn} \underbrace{\phi_{align}(a_{ij}, a_{lm})}_{\text{Penalty when role n} \neq \text{k}}$$

subject to:

$$\sum_k z_{ijk} \leq 1 \qquad \forall\ a_{ij} \in \text{sentence}_i$$

$$\sum_j z_{ijk} \leq 1 \qquad \forall\ a_{ij} \in \text{sentence}_i, k \in \text{R}$$

$\tilde{N}_k$: Approximate number of arguments with role k

$\tilde{N}_{k'}$: Approximate number of arguments with role $n \neq k$

Figure 2: An Integer Linear Program formulation of the Cross-sentence Inference.

fillers of different roles.

The decision variables z_{ijk} denote role assignments to arguments. When z_{ijk} is set it denotes that argument j in sentence i (a_{ij}) has been assigned role k. The objective function uses three components to assign scores to an assignment.

1. Classifier Score $\phi_{role}(a_{ij}, k)$ – This is the score of a sentence-level role classifier for assigning role k to argument a_{ij}.
2. Within Role Compatibility $\Delta(a_{ij}, k)$ – This is a measure of argument a_{ij}'s compatibility with other arguments which have also been assigned the same role k. We measure compatibility using a notion of alignment. An argument is said to align with another if they are similar to each other in some respect (either lexically or semantically). As we show later, we develop an alignment cclassifier which predicts an alignment score ϕ_{align} for each pair of arguments. The compatibility is defined as a normalized sum of the alignment scores for argument a_{ij} paired with

other arguments that have also been assigned the role k. Without some normalization roles with many arguments will receive higher compatibility scores.To avoid this, we normalize by $(1/\tilde{N}_k)$, where $\tilde{N}_k$ refers to the number of arguments that the role classifier originally labeled with role k, an approximation to the number of arguments that are currently assigned role k by the ILP.

3. Across Role Incompatibility $\nabla(a_{ij}, k)$ – This is a measure of how well a_{ij} aligns with the other arguments that are assigned a different role ($n \neq k$). For good assignments this quantity should be low. Therefore we add this as a penalty to the objective. As with Δ, we use an approximation for normalization $(1/\tilde{N}_{k'})$, which is the product of other roles and the number of arguments in other sentences that can receive these roles. Because $\tilde{N}_{k'}$ is typically higher, we boost this score by 2 to balance against Δ.

Last, we use two sets of hard constraints to enforce the standard within-sentence expectations for roles: 1) A single argument can receive only one role label, and 2) A sentence cannot have more than one argument with the same label, except for the NONE role.

We use an off-the-shelf solver in Gurobi (www.gurobi.com) to find an approximate solution to the resulting optimization problem.

3.3 Role Classifier (Φ_{role})

The role classifier provides a score for each role label for a given argument. Although existing SRL and frame semantic parsers do not directly produce the role information we need (Section 2), we build on them by using their outputs for building a process role classifier.

Before we can assign role labels, we first need to generate candidate arguments. Using EasySRL (Lewis et al., 2015), a state-of-the-art SRL system, we generate the candidate argument spans for each predicate (verbs) in the sentence. Then, using a linear SVM classifier (Fan et al., 2008), we score the candidate arguments and the predicates for our four roles and a special NONE role to indicate the argument is not one of the four. The classifier is trained with a set of annotated examples (see Section 4) with the following sets of features.

i) Lexical and Syntactic – We use a small set of standard SRL features such as lexical and syntactic contexts of arguments (e.g., head word, its POS tag) and predicate-argument path features (e.g, dependency paths). We also add features that are specific to the nature of the process sentences. In particular, we encode syntactic relationships of arguments with respect to the process name mention in the sentence. We use Stanford CoreNLP toolkit (Manning et al., 2014) to obtain POS tags, and dependency parses to build these features.

ii) PropBank roles – While they do not have a 1-to-1 correspondence with process roles, we use the EasySRL roles coupled with the specific predicate as a feature to provide useful evidence towards the process role.

iii) Framenet Frames – We use the frames evoked by the words in the sentence to allow better feature sharing among related processes. For instance, the contexts of undergoers in evaporation and condensation are likely to be similar as they are both state changes which evoke the same `Undergo_Change` frame in FrameNet.

iv) Query patterns – We use query patterns to find sentences that are likely to contain the target roles of interest. The query pattern that retrieved a sentence can help bias the classifier towards roles that are likely to be expressed in it.

3.4 Alignment Classifier (Φ_{align})

Our goal with alignment is to identify arguments that should receive similar role labels. One way to do this argument alignment is to use techniques developed for phrase level entailment or similarity which often use resources such as WordNet and distributional embeddings such as word2vec (Mikolov et al., 2013) vectors. It turns out that this simple entailment or argument similarity, by itself, is not enough in many cases for our task[2]. Moreover, the enabler, and the result roles are often long phrases whose text-based similarity is not reliable. A more robust approach is necessary. Lexical and syntactic similarity of arguments and the context in which they are embedded can provide valuable additional information. Table 3 shows a complete list of features we use to train the alignment classifier.

[2]We used an approach that combined WordNet-based phrase similarity method, and Word2Vec vector similarity, where the vectors were learned from a general news domain.

Lexical
Entailment score of arguments. Word2vec similarity of argument vectors. Word2Vec similarity of head nodes of arguments. Word2Vec similarity of parent of the head nodes. Word2Vec similarity of verbs of argument sentences. Jaccard similarity of children of the head node.
Syntactic
Similarities of frames to right and left of arguments. Jaccard similarity of POS tags of argument. POS tag of head nodes match (boolean). POS tag of head node parents match (boolean). Similarity of dep. path from arg to process name. Similarity of POS tags on arg to process name path. Similarity of POS tags of arg's children. Similarity of the dependencies of the arg's head.
Sentence
Query patterns match argument sentences (boolean).

Table 3: Alignment Classifier Features. Similarities of sets were calculated using Jaccard co-efficient.

Fortunately, learning this classifier does not require any additional training data. The original data with annotated semantic role labels can be easily transformed to generate supervision for this classifier. For any given process, we consider all pairs of arguments in different sentences (i.e., (a_{ij}, a_{lm}) : $i \neq l$) and label them as aligned if they are labeled with the same role, or unaligned otherwise.

4 Evaluation

Our goal is to generate knowledge about processes discussed in grade-level science exams. Since existing semantic resources such as FrameNet do not provide adequate coverage for these, we created a dataset of process sentences annotated with the four process roles: undergoer, enabler, action, and result.

4.1 Dataset

This dataset consists of 1205 role fillers extracted from 537 sentences retrieved from the web. We first compiled the target processes from a list of process-oriented questions found in two collections: (i) New York Regents science exams (Clark, 2015), and (ii) helpteaching.com, a Web-based collection of practice questions. Then, we identified 127 process questions from which we obtained a set of 180 unique target processes. For each target process, we queried the web using Google to find definition-style sentences, which describe the target process. For each process we discarded some noisy sentences through a combination of automatic and manual filtering.

Query Patterns
⟨name⟩ is the process of ⟨x⟩
⟨name⟩ is the process by which ⟨x⟩
⟨name⟩ {occurs when} ⟨x⟩
⟨name⟩ { helps to | causes } ⟨x⟩

Table 4: Example query patterns used to find process description sentences.

Table 4 shows some examples of the 14 query patterns that we used to find process descriptions. Because these patterns are not process-specific, they work for unseen processes as well.

To find role fillers from these sentences, we first processed each sentence using EasySRL (Lewis et al., 2015) to generate candidate arguments. Some of the query patterns can be used to generate additional arguments. For example, in the pattern "⟨name⟩ is the process of ⟨x⟩" if ⟨x⟩ is a noun then it is likely to be an undergoer, and thus can be a good candidate. [3]. Then two annotators annotated the candidate arguments with the target roles if one were applicable and marked them as NONE otherwise. Disagreements were resolved by a third annotator. The annotations spanned a random subset of 54 target processes. The role label distribution is shown below:

Role	No. of instance
Undergoer	77
Enabler	154
Action	315
Result	194
NONE	465

Table 5: Role distribution

We conducted five fold cross validation experiments to test role extraction. To ensure that we are testing the generalization of the approach to unseen

[3]These patterns are ambiguous and are not adequate by themselves for accurately extracting the roles. We use them as features.

processes, we generated the folds such that the processes in the test fold were unseen during training. We compared the basic role classifier described in Section 3.3, the *within sentence* and the *cross sentence* inference models. We tune the ILP parameter λ for cross sentence inference based on a coarse-grained sweep on the training folds.

We also compared with a simple baseline that learned a mapping from PropBank roles produced by EasySRL system to the process roles by using the roles and the verb as features. We also add the FrameNet frames invoked by the lexical unit in the sentence. Note this is essentially a subset of the features we use in our role classifier. As a second baseline, we compare with a (nearly) out-of-the-box application of SEMAFOR (Das et al., 2010), a FrameNet based frame-semantic parser. We modified SEMAFOR to override the frame identification step since the process frame information is already associated with the test sentences.

4.2 Cross-Sentence Inference Results

Table 6 compares performance of the different methods. The learned role mapping of shallow semantic roles performs better than SEMAFOR but worse than the simple role classifier. SEMAFOR uses a large set of features which help it scale for a diverse set of frames in FrameNet. However, many of these many not be well suited for the process sentences in our relatively smaller dataset. Therefore, we use our custom role classifier as a strong baseline to demonstrate within and cross sentence gains. Enforcing sentence-level consistency through joint inference, shown as (+within sent.), improves over the baseline which does not use any consistency. It gains precision (by nearly 8 points), while loosing recall in the trade-off (by about 4.7 points) to yield an overall gain in F1 by 1.05 points. Enforcing cross sentence consistency, shown as (+cross sent.) provides additional gains beyond within sentence inference by another 2.38 points in F1 [4]. Table 7 shows how the gains are distributed across different roles. Cross sentence inference provides improvements for all roles, with the biggest for undergoers (nearly 4 points).

Method	Prec.	Rec.	F1
Role mapping	56.62	59.60	58.07
SEMAFOR	40.72	50.54	45.10
Role class. (ϕ_{role})	78.48	**78.62**	78.55
+ within sent.	86.25	73.91	79.60
+ cross sent.	**89.84**	75.36	**81.97**††

Table 6: Process role inference performance. †† indicates significant improvement over Role + within sentence system.

Method	Und.	Ena.	Act.	Res.
Role Class.	65.38	73.84	83.58	77.30
+ within	66.01	73.11	86.70	76.11
+ cross	**70.00**	**74.31**	**89.30**	**78.00**

Table 7: Performance (F1) across all roles.

Figure 3 shows the precision/recall plots for the basic role classifier and within and cross sentence inference. Both inference models trade recall for gains in precision. Cross sentence yields higher precision at most recall levels, for a smaller overall loss in recall compared to within sentence (1.6 versus 4.9).

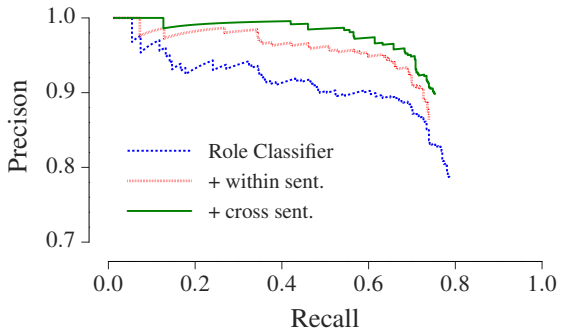


Figure 3: Precision/Recall trade-offs for process role inference. y-axis is truncated at 0.7 to better visualize the differences.

4.3 Ablations

Table 8 shows the utility of various components of cross sentence inference. Using argument entailment alone turns out to be ineffective and only produces a minor improvement (0.16 in F1). However, the alignment classifier scores are much more effective and yield about 2.37 points gain in F1. Both within and across role compatibilities, Δ and ∇, yield statistically significant improvements[5] over

[4] The single parameter in ILP turned out to be stable across the folds and obtained this best value at $\lambda = 0.8$.

[5] Significance measured using approximate randomization test

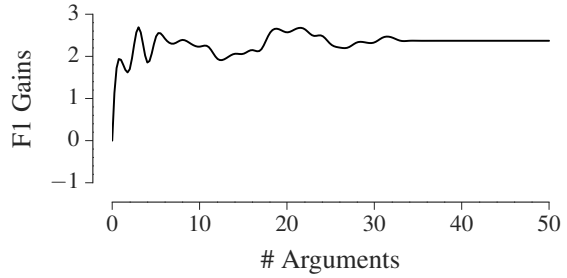


Figure 4: Cross sentence gains in F1 when varying the number of most similar arguments used to assess compatibilities.

within sentence inference. Combining these complementary compatibilities provides the best gains.

Method	Prec.	Rec.	F1
within sent.	86.25	73.91	79.60
+ Entailment			
cross sent. w/ Δ	85.13	72.64	78.39
cross sent. w/ ∇	85.98	73.36	79.17
cross sent. w/ $\Delta + \nabla$	86.62	73.91	79.76
+ Alignment Classifier			
cross sent. w/ Δ	89.07	75.36	81.64††
cross sent. w/ ∇	88.72	75.54	81.60††
cross sent. w/ $\Delta + \nabla$	**89.84**	75.36	**81.97**††

Table 8: Performance impact of inference components. †† indicates significant improvement over within sentence.

We also studied the effect of varying the number of arguments that ILP uses to measure the compatibility of role assignments. Specifically, we allow inference to use just the top k alignments from the alignment classifier. Figure 4 shows the main trend. Using just the top similar argument already yields a 1 point gain in F1. Using more arguments tends to increase gains in general but with some fluctuations. Finding an assignment that respects all compatibilities across many argument pairs can be difficult. As seen in the figure, at some of the shorter span lengths we see a slightly larger gain ($+0.3$) compared to using all spans. This hints at benefits of a more flexible formulation that makes joint decisions on alignment and role label assignments.

Table 9 shows an ablation of the alignment classifier features. Entailment of arguments is the most informative feature for argument alignment. Adding lexical and syntactic context compatibilities adds significant boosts in precision and recall. Knowing that the arguments are retrieved by the same query pattern (sentence feature) only provides minor improvements. Even though the overall classification performance is far from perfect, cross sentence can benefit from alignment as long as it provides a higher score for argument pairs that should align compared to those that should not.

Feature	P	R	F1
Entailment score only	39.55	14.59	21.32
+Lexical	50.75	26.02	34.40
+Syntactic	62.31	31.47	41.82
+Sentence	62.33	31.41	41.53

Table 9: Performance of different feature groups for alignment.

4.3.1 Error Analysis

We conduct an error analysis over a random set of 50 errors observed for cross sentence inference. In addition to issues from noisy web sentences and nested arguments from bad candidate extraction, we find the following main types of errors:

- Dissimilar role fillers (27.5 %) – In some processes, the fillers for the *result* role have high levels of variability that makes alignment error prone. For the process camouflage, for instance, the *result* roles include 'disorientation', 'protect from predator', 'remain undetected' etc.
- Bad role classifier scores (37.5%) – For some instances the role classifier assign high scores to incorrect labels, effectively preventing the ILP from flipping to the correct role. For example, the argument that follows "causes" tends to be a result in many cases but not always, leading to high scoring errors. For example, in the sentence with "...when heat from the sun causes water on earth's ...", the role classifier incorrectly assigns 'water' to a result role with high confidence.
- Improper Weighting (7.5%)– Sometimes the ILP does not improve upon a bad top choice from the role classifier. In some of these cases, rather than the fixed lambda, a different weighted combination of role and alignment classifier scores

would have helped the ILP to flip. For example, the argument 'under autumn conditions' from the sentence 'hibernation occurs when the insects are maintained under autumn conditions.' has a good role score and is similar to other correctly labeled enablers such as 'cold , winter conditions' but yet is unable to improve.

The rest (27.5 %) are due to noisy web sentences, incorrect argument extraction and errors outside the scope of cross sentence inference.

5 Conclusions

Simple role-based knowledge is essential for recognizing and reasoning about situations involving processes. In this work we developed a cross sentence inference method for automatically acquiring such role-based knowledge for new processes. The main idea is to enforce compatibility among roles extracted from sentences belonging to a single process. We find that the compatibility can be effectively assessed using an alignment classifier built without any additional supervision. Empirical evaluation on a process dataset shows that cross sentence inference using an Integer Linear Program helps improve the accuracy of process knowledge extraction.

6 Acknowledgement

The authors would like to thank the anonymous reviewers for helpful comments, Meghana Kshirsagar, Sam Thomson, Mike Lewis for answering implementation details of their systems, and the Stony Brook NLP Lab members for their valuable feedback and suggestions. This work is supported in part by Foreign Fulbright PhD Fellowship and by the grant from Allen Institute for Artificial Intelligence.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: Take the aristo challenge. *to appear*.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Hagen Fürstenau and Mirella Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore.

Hagen Fürstenau and Mirella Lapata. 2012. Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.

Sujay Kumar Jauhar, Peter D. Turney, and Eduard H. Hovy. 2016. Tables as semi-structured knowledge for question answering. In *ACL*.

Paul Kingsbury and Martha Palmer. 2003. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, volume 3. Citeseer.

Ivan Titov Alexandre Klementiev. 2012. Semi-supervised semantic role labeling: Approaching from an unsupervised perspective. In *Proceedings of the COLING Conference.*

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime G. Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *ACL*.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California, June. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction with graph partitioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1320–1331, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* ccg parsing and semantic role labelling. In *Empirical Methods in Natural Language Processing*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Luiz Augusto Pizzato and Diego Mollá. 2008. Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 74–81. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics (TACL)*, 3:449–460.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 45–50, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21.

Robert S Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *EMNLP*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL*.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez i Villodre. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *ACL*.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez i Villodre, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39:631–663.

Toward Socially-Infused Information Extraction: Embedding Authors, Mentions, and Entities

Yi Yang
Georgia Institute of Technology
Atlanta, GA 30308, USA
yiyang@gatech.edu

Ming-Wei Chang
Microsoft Research
Redmond, WA 98052, USA
minchang@microsoft.com

Jacob Eisenstein
Georgia Institute of Technology
Atlanta, GA 30308, USA
jacobe@gatech.edu

Abstract

Entity linking is the task of identifying mentions of entities in text, and linking them to entries in a knowledge base. This task is especially difficult in microblogs, as there is little additional text to provide disambiguating context; rather, authors rely on an implicit common ground of shared knowledge with their readers. In this paper, we attempt to capture some of this implicit context by exploiting the social network structure in microblogs. We build on the theory of *homophily*, which implies that socially linked individuals share interests, and are therefore likely to mention the same sorts of entities. We implement this idea by encoding authors, mentions, and entities in a continuous vector space, which is constructed so that socially-connected authors have similar vector representations. These vectors are incorporated into a neural structured prediction model, which captures structural constraints that are inherent in the entity linking task. Together, these design decisions yield F1 improvements of 1%-5% on benchmark datasets, as compared to the previous state-of-the-art.

1 Introduction

Entity linking on short texts (e.g., Twitter messages) is of increasing interest, as it is an essential step for many downstream applications, such as market research (Asur and Huberman, 2010), topic detection and tracking (Mathioudakis and Koudas, 2010), and question answering (Yih et al., 2015). Tweet entity linking is a particularly difficult problem, because the short context around an entity mention is often insufficient for entity disambiguation. For example, as shown in Figure 1, the entity mention 'Giants' in tweet t_1 can refer to the NFL football team *New York Giants* or the MLB baseball team *San Francisco Giants*. In this example, it is impossible to disambiguate between these entities solely based on the individual text message.

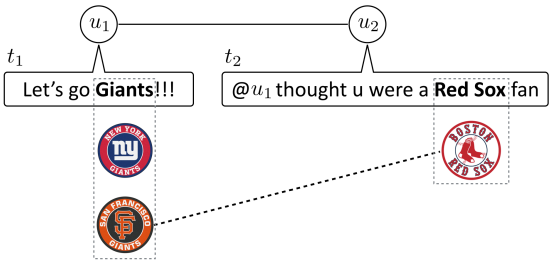


Figure 1: Illustration on leveraging social relations for entity disambiguation. Socially connected users u_1 and u_2 tend to talk about similar entities (baseball in the example).

We propose to overcome the difficulty and improve the entity disambiguation capability of the entity linking system by employing social network structures. The sociological theory of *homophily* asserts that socially connected individuals are more likely to have similar behaviors or share similar interests (McPherson et al., 2001). This property has been used to improve many natural language processing tasks such as sentiment analysis (Tan et al., 2011; Yang and Eisenstein, 2015), topic classification (Hovy, 2015) and user attribute inference (Li et al., 2015). We assume Twitter users will have similar interests in real world entities to their near neighbors — an assumption of *entity homophily* — which

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1452–1461,
Austin, Texas, November 1-5, 2016.

is demonstrated in Figure 1. The social relation between users u_1 and u_2 may lead to more coherent topics in tweets t_1 and t_2. Therefore, by successfully linking the less ambiguous mention 'Red Sox' in tweet t_2 to the *Boston Red Sox* baseball team, the tweet entity linking system will be more confident on linking 'Giants' to the *San Francisco Giants* football team in tweet t_1.

To exploit social information, we adopt the recent advance on embedding information networks (Tang et al., 2015), which induces low-dimensional representations for author nodes based on the network structure. By learning the semantic interactions between the author embeddings and the pre-trained Freebase entity embeddings, the entity linking system can incorporate more disambiguating context from the social network. We also consider low-dimensional representations of mentions, another source of related information for entity linking, with the intuition that semantically related mentions can refer to similar entities. Previously proposed approaches (Guo et al., 2013a; Yang and Chang, 2015) are based on hand-crafted features and off-the-shelf machine learning algorithms. Our preliminary study suggests that simply augmenting the traditional surface features with the distributed representations barely improves the performance of these entity linking systems. Therefore, we propose NTEL, a **N**eural model for **T**weet **E**ntity **L**inking, to leverage the distributed representations of authors, mentions, and entities. NTEL can not only make efficient use of statistical surface features built from a knowledge base, but also learn the interactions between these distributed representations.

Our contributions are summarized as follows:

- We present a novel model for entity linking that exploits distributed representations of users, mentions, and entities.
- We combine this distributed model with a feed-forward neural network that learns non-linear combinations of surface features.
- We perform message-level inference using a dynamic program to avoid overlapping mentions. The architecture is trained with loss-augmented decoding, a large margin learning technique for structured prediction.

Data	# Tweet	# Entity	Date
NEEL-train	2,340	2,202	Jul. - Aug. 2011
NEEL-test	1,164	687	Jul. - Aug. 2011
TACL	500	300	Dec. 2012

Table 1: Statistics of data sets.

- The complete system, NTEL, outperforms the previous state-of-the-art (Yang and Chang, 2015) by 3% average F1 on two benchmark datasets.

2 Data

Two publicly available datasets for tweet entity linking are adopted in the work. NEEL is originally collected and annotated for the Named Entity Extraction & Linking Challenge (Cano et al., 2014), and TACL is first used and released by Fang and Chang (2014). The datasets are then cleaned and unified by Yang and Chang (2015). The statistics of the datasets are presented in Table 1.

3 Testing Entity Homophily

The hypothesis of *entity homophily*, as presented in the introduction, is that socially connected individuals are more likely to mention similar entities than disconnected individuals. We now test the hypothesis on real data before we start building our entity linking systems.

Twitter social networks We test the assumption on the users in the NEEL-train dataset. We construct three author social networks based on the follower, mention and retweet relations between the 1,317 authors in the NEEL-train dataset, which we refer as FOLLOWER, MENTION and RETWEET. Specifically, we use the Twitter API to crawl the friends of the NEEL users (individuals that they follow) and the mention/retweet links are induced from their most recent 3,200 tweets.[1] We exploit bi-directed links to create the undirected networks, as bi-directed links result in stronger social network ties than directed links (Kwak et al., 2010; Wu et al., 2011). The numbers of social relations for the networks are 1,604, 379 and 342 respectively.

[1] We are able to obtain at most 3,200 tweets for each Twitter user, due to the Twitter API limits.

Network	$sim(i \leftrightarrow j)$	$sim(i \not\leftrightarrow j)$
FOLLOWER	0.128	0.025
MENTION	0.121	0.025
RETWEET	0.173	0.025

Table 2: The average entity-driven similarity results for the networks.

Metrics We propose to use the *entity-driven similarity* between authors to test the hypothesis of entity homophily. For a user u_i, we employ a Twitter NER system (Ritter et al., 2011) to detect entity mentions in the timeline, which we use to construct a user entity vector $\mathbf{u}_i^{(ent)}$, so that $u_{i,j}^{(ent)} = 1$ iff user i has mentioned entity j.[2] The entity-driven similarity between two users u_i and u_j is defined as the cosine similarity score between the vectors $\mathbf{u}_i^{(ent)}$ and $\mathbf{u}_j^{(ent)}$. We evaluate the three networks by calculating the average entity-driven similarity of the connected user pairs and that of the disconnected user pairs, which we name as $sim(i \leftrightarrow j)$ and $sim(i \not\leftrightarrow j)$.

Results The entity-driven similarity results of these networks are presented in Table 2. As shown, $sim(i \leftrightarrow j)$ is substantially higher than $sim(i \not\leftrightarrow j)$ on all three social networks, indicating that socially connected individuals clearly tend to mention more similar entities than disconnected individuals. Note that $sim(i \not\leftrightarrow j)$ is approximately equal to the same base rate defined by the average entity-driven similarity of all pairs of users, because the vast majority of user pairs are disconnected, no matter how to define the network. Among the three networks, RETWEET offers slightly higher $sim(i \leftrightarrow j)$ than FOLLOWER and MENTION. The results verify our hypothesis of entity homophily, which forms the basis for this research. Note that all social relation data was acquired in March 2016; by this time, the authorship information of 22.1% of the tweets in the NEEL-train dataset was no longer available, because the tweets or user accounts had been deleted.

4 Method

In this section, we present, NTEL, a novel neural based tweet entity linking framework that is able to leverage social information. We first formally define the task of tweet entity linking. Assume we are given an entity database (e.g., Wikipedia or Freebase), and a lexicon that maps a surface form into a set of entity candidates. For each input tweet, we consider any n-grams of the tweet that match the lexicon as mention candidates.[3] The entity linking system maps every mention candidate (e.g., 'Red Sox') in the message to an entity (e.g., *Boston Red Sox*) or to **Nil** (i.e., not an entity). There are two main challenges in the problem. First, a mention candidate can often potentially link to multiple entities according to the lexicon. Second, as shown in Figure 2, many mention candidates overlap with each other. Therefore, the entity linking system is required to disambiguate entities and produce non-overlapping entity assignments with respect to the mention candidates in the tweet.

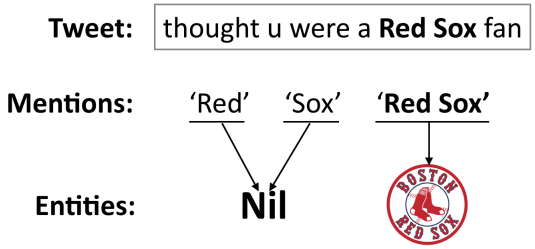


Figure 2: Illustration of the non-overlapping structure for the task of tweet entity linking. In order to link 'Red Sox' to a real entity, 'Red' and 'Sox' should be linked to **Nil**.

We formalize this task as a structured learning problem. Let $\mathbf{x}$ be the tweet, u be the author, and $\mathbf{y} = \{y_t\}_{t=1}^{T}$ be the entity assignments of the T mention candidates in the tweet. The overall scoring function $s(\mathbf{x}, \mathbf{y}, u)$ can be decomposed as follows,

$$s(\mathbf{x}, \mathbf{y}, u) = \sum_{t=1}^{T} g(\mathbf{x}, y_t, u, t), \qquad (1)$$

where $g(\mathbf{x}, y_t, u, t)$ is the scoring function for the t-th mention candidate choosing entity y_t. Note that the system needs to produce non-overlapping entity assignments, which will be resolved in the inference algorithm.

The overview of NTEL is illustrated in Figure 3. We further break down $g(\mathbf{x}, y_t, u, t)$ into two scoring

[2] We assume each name corresponds to a single entity for this metric, so this metric only approximates entity homophily.

[3] We adopted the same entity database and lexicon as those used by Yang and Chang (2015).

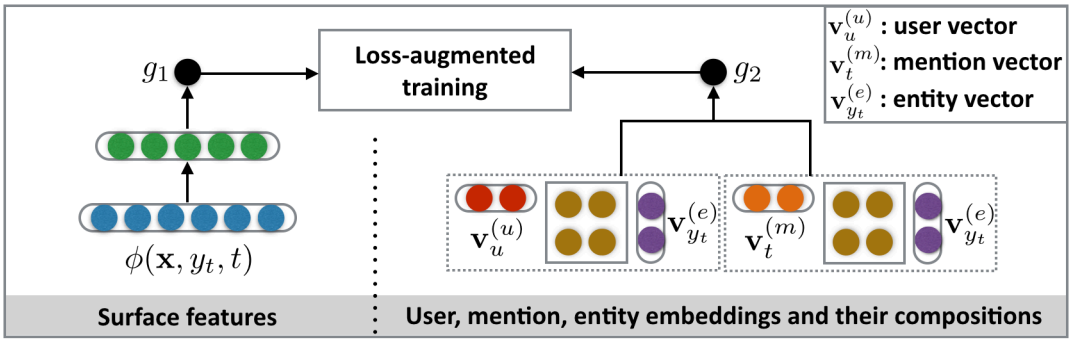


Figure 3: The proposed neural network approach for tweet entity linking. A composition model based on bilinear functions is used to learn the semantic interactions of user, mention, and entity.

functions:

$$g(\mathbf{x}, y_t, u, t; \Theta_1, \Theta_2) = g_1(\mathbf{x}, y_t, t; \Theta_1) + g_2(\mathbf{x}, y_t, u, t; \Theta_2), \quad (2)$$

where g_1 is the scoring function for our basic surface features, and g_2 is the scoring function for modeling user, mention, entity representations and their compositions. Θ_1 and Θ_2 are model parameters that will be detailed below. We choose to use a multilayer perceptron (MLP) to model $g_1(\mathbf{x}, y_t, t; \Theta_1)$, and we employ simple yet efficient bilinear functions to learn the compositions of user, mention, and entity representations $g_2(\mathbf{x}, y_t, u, t; \Theta_2)$. Finally, we present a training algorithm based on loss-augmented decoding and a non-overlapping inference algorithm.

4.1 Modeling Surface Features

We include the 37 features used by Yang and Chang (2015) as our surface feature set. These features are extracted from various sources, including a named entity recognizer, an entity type recognizer, and some statistics of the Wikipedia pages.

We exploit a multilayer perceptron (MLP) to transform the surface features to a real-valued score. The output of the MLP is formalized as follows,

$$g_1(\mathbf{x}, y_t, t; \Theta_1) = \boldsymbol{\beta}^\top \mathbf{h} + b$$
$$\mathbf{h} = \tanh(\mathbf{W}\phi(\mathbf{x}, y_t, t) + \mathbf{b}), \quad (3)$$

where $\phi(\mathbf{x}, y_t, t)$ is the feature function, $\mathbf{W}$ is an $M \times D$ matrix, the weights $\mathbf{b}$ are bias terms, and $\mathbf{h}$ is the output of the hidden layer of the MLP. $\boldsymbol{\beta}$ is an M dimensional vector of weights for the output score, and b is the bias term. The parameters of the MLP are $\Theta_1 = \{\mathbf{W}, \mathbf{b}, \boldsymbol{\beta}, b\}$. Yang and Chang (2015) argue that non-linearity is the key for obtaining good results on the task, as linear models are not expressive enough to capture the high-order relationships between the dense features. They propose a tree-based non-linear model for the task. The MLP forms simple non-linear mappings between the input features and the output score, whose parameters will be jointly learnt with other components in NTEL.

4.2 Modeling User, Mention, and Entity

To leverage the social network structure, we first train low-dimensional embeddings for the authors using the social relations. The mention and entity representations are given by word embeddings learnt with a large Twitter corpus and pre-trained Freebase entity embeddings respectively. We will denote the user, word, entity embedding matrices as:

$$\mathbf{E}^{(u)} = \{\mathbf{v}_u^{(u)}\} \quad \mathbf{E}^{(w)} = \{\mathbf{v}_w^{(w)}\} \quad \mathbf{E}^{(e)} = \{\mathbf{v}_e^{(e)}\},$$

where $\mathbf{E}^{(u)}, \mathbf{E}^{(w)}, \mathbf{E}^{(e)}$ are $V^{(u)} \times D^{(u)}, V^{(w)} \times D^{(w)}, V^{(e)} \times D^{(e)}$ matrices, and $\mathbf{v}_u^{(u)}$, $\mathbf{v}_w^{(w)}$, $\mathbf{v}_e^{(e)}$ are $D^{(u)}$, $D^{(w)}$, $D^{(e)}$ dimensional embedding vectors respectively. $V^{(u)}, V^{(w)}, V^{(e)}$ are the vocabulary sizes for users, words, and entities. Finally, we present a composition model for learning semantic interactions between user, mention, and entity.

User embeddings We obtain low-dimensional Twitter author embeddings $\mathbf{E}^{(u)}$ using *LINE* — the recently proposed model for embedding information networks (Tang et al., 2015). Specifically, we train *LINE* with the second-order proximity, which assumes that Twitter users sharing many neighbors are

close to each other in the embedding space. According to the original paper, the second-order proximity yields slightly better performances than the first-order proximity, which assumes connecting users are close to each other, on a variety of downstream tasks.

Mention embeddings The representation of a mention is the average of embeddings of words it contains. As each mention is typically one to three words, the simple representations often perform surprisingly well (Socher et al., 2013). We adopt the structured skip-gram model (Ling et al., 2015) to learn the word embeddings $\mathbf{E}^{(w)}$ on a Twitter corpus with 52 million tweets (Owoputi et al., 2013). The mention vector of the t-th mention candidate can be written as:

$$\mathbf{v}_t^{(m)} = \frac{1}{|\mathbf{x}_t^{(w)}|} \sum_{w \in \mathbf{x}_t^{(w)}} \mathbf{v}_w^{(w)}, \quad (4)$$

where $\mathbf{x}_t^{(w)}$ is the set of words in the mention.

Entity embeddings We use the pre-trained Freebase entity embeddings released by Google to represent entity candidates, which we refer as $\mathbf{E}^{(e)}$.[4] The embeddings are trained with the skip-gram model (Mikolov et al., 2013) on 100 billion words from various news articles. The entity embeddings can also be learnt from Wikipedia hyperlinks or Freebase entity relations, which we leave as future work.

Compositions of user, mention, and entity The distributed representations of users, mentions, and entities offer additional information that is useful for improving entity disambiguation capability. In particular, we explore the information by making two assumptions: socially connected users are interested in similar entities (entity homophily), and semantically related mentions are likely to be linked to similar entities.

We utilize a simple composition model that takes the form of the summation of two bilinear scoring functions, each of which explicitly leverages one of the assumptions. Given the author representation $\mathbf{v}_u^{(u)}$, the mention representation $\mathbf{v}_t^{(m)}$, and the entity representation $\mathbf{v}_{y_t}^{(e)}$, the output of the model can be written as:

$$\begin{aligned} g_2(\mathbf{x}, y_t, u, t; \Theta_2) =& {\mathbf{v}_u^{(u)}}^\top \mathbf{W}^{(u,e)} \mathbf{v}_{y_t}^{(e)} \\ &+ {\mathbf{v}_t^{(m)}}^\top \mathbf{W}^{(m,e)} \mathbf{v}_{y_t}^{(e)}, \end{aligned} \quad (5)$$

where $\mathbf{W}^{(u,e)}$ and $\mathbf{W}^{(m,e)}$ are $D^{(u)} \times D^{(e)}$ and $D^{(w)} \times D^{(e)}$ bilinear transformation matrices. Similar bilinear formulation has been used in the literature of knowledge base completion and inference (Socher et al., 2013; Yang et al., 2014). The parameters of the composition model are $\Theta_2 = \{\mathbf{W}^{(u,e)}, \mathbf{W}^{(m,e)}, \mathbf{E}^{(u)}, \mathbf{E}^{(w)}, \mathbf{E}^{(e)}\}$.

4.3 Non-overlapping Inference

The non-overlapping constraint for entity assignments requires inference method that is different from the standard Viterbi algorithm for a linear chain. We now present a variant of the Viterbi algorithm for the non-overlapping structure. Given the overall scoring function $g(\mathbf{x}, y_t, u, t)$ for the t-th mention candidate choosing an entity y_t, we sort the mention candidates by their end indices and define the Viterbi recursion by

$$\hat{y}_t = \underset{y_t \in \mathcal{Y}_{\mathbf{x}_t}, y_t \neq \textbf{Nil}}{\arg\max}\ g(\mathbf{x}, y_t, u, t) \quad (6)$$

$$a(1) = \max(g(\mathbf{x}, \textbf{Nil}, u, 1), g(\mathbf{x}, \hat{y_1}, u, 1)) \quad (7)$$

$$a(t) = \max\left(\psi_t(\textbf{Nil}), \psi_t(\hat{y}_t)\right) \quad (8)$$

$$\psi_t(\textbf{Nil}) = g(\mathbf{x}, \textbf{Nil}, u, t) + a(t-1) \quad (9)$$

$$\begin{aligned} \psi_t(\hat{y}_t) =& g(\mathbf{x}, \hat{y}_t, u, t) + \sum_{prev(t) < t' < t} g(\mathbf{x}, \textbf{Nil}, u, t') \\ &+ a(prev(t)) \end{aligned} \quad (10)$$

where $\mathcal{Y}_{\mathbf{x}_t}$ is set of entity candidates for the t-th mention candidate, and $prev(t)$ is a function that points out the previous non-overlapping mention candidate for the t-th mention candidate. We exclude any second-order features between entities. Therefore, for each mention candidate, we only need to decide whether it can take the highest scored entity candidate $\hat{y}_t$ or the special **Nil** entity based on whether it is overlapped with other mention candidates.

[4] Available at https://code.google.com/archive/p/word2vec/

4.4 Loss-augmented Training

The parameters need to be learnt during training are $\Theta = [\Theta_1, \{\mathbf{W}^{(u,e)}, \mathbf{W}^{(m,e)}\}]$.[5] We train NTEL by minimizing the following loss function for each training tweet:

$$L(\Theta) = \max_{\mathbf{y} \in \mathcal{Y}_\mathbf{x}} (\Delta(\mathbf{y}, \mathbf{y}^*) + s(\mathbf{x}, \mathbf{y}, u)) - s(\mathbf{x}, \mathbf{y}^*, u), \quad (11)$$

where $\mathbf{y}^*$ is the gold structure, $\mathcal{Y}_\mathbf{x}$ represents the set of valid output structures for $\mathbf{x}$, and $\Delta(\mathbf{y}, \mathbf{y}^*)$ is the weighted hamming distance between the gold structure $\mathbf{y}^*$ and the valid structure $\mathbf{y}$. The hamming loss is decomposable on the mention candidates, which enables efficient inferences. We set the hamming loss weight to 0.2 after a preliminary search. Note that the number of parameters in our composition model is large. Thus, we include an L2 regularizer on these parameters, which is omitted from Equation 11 for brevity. The evaluation of the loss function corresponds to the loss-augmented inference problem:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}_\mathbf{x}} (\Delta(\mathbf{y}, \mathbf{y}^*) + s(\mathbf{x}, \mathbf{y}, u)), \quad (12)$$

which can be solved by the above non-overlapping inference algorithm. We employ vanilla SGD algorithm to optimize all the parameters. The numbers of training epochs are determined by early stopping (at most 1000 epochs). Training takes 6-8 hours on 4 threads.

5 Experiments

In this section, we evaluate NTEL on the NEEL and TACL datasets as described in § 2, focusing on investigating whether social information can improve the task. We also compare NTEL with the previous state-of-the-art system.

5.1 Social Network Expansion

We utilize Twitter follower, mention, and retweet social networks to train user embeddings. We were able to identify 2,312 authors for the tweets of the two datasets in March 2016. We then used the Twitter API to crawl their friend links and timelines, from which we can induce the networks. We find the numbers of social connections (bidirectional links) between these users are relatively small. In order to learn better user embeddings, we expand the set of author nodes by including nodes that will do the most to densify the author networks. For the follower network, we add additional individuals who are followed by at least twenty authors in the original set. For the mention or retweet networks, we add all users who have mentioned or retweeted by at least ten authors in the original set. The statistics of the resulting networks are presented in Table 3.

Network	# Author	# Relation
FOLLOWER+	8,772	286,800
MENTION+	6,119	57,045
RETWEET+	7,404	59,313

Table 3: Statistics of author social networks used for training user embeddings.

5.2 Experimental Settings

Following Yang and Chang (2015), we train all the models with the NEEL-train dataset and evaluate different systems on the NEEL-test and TACL datasets. In addition, 800 tweets from the NEEL-train dataset are sampled as our development set to perform parameter tuning. Note that Yang and Chang (2015) also attempt to optimize F1 scores by balancing precision and recall scores on the development set; we do not fine tune our F1 in this way, so that we can apply a single trained system across different test sets.

Metrics We follow prior work (Guo et al., 2013a; Yang and Chang, 2015) and perform the standard evaluation for an end-to-end entity linking system, computing precision, recall, and F1 score according to the entity references and the system outputs. An output entity is considered as correct if it matches the gold entity and the mention boundary overlaps with the gold mention boundary. More details about the metrics are described by Carmel et al. (2014).

Competitive systems Our first baseline system, NTEL-nonstruct, ignores the structure information and makes the entity assignment decision for each mention candidate individually. For NTEL, we start with a baseline system using the surface features, and then incorporate the two bilinear functions

[5] We fixed the pre-trained embedding matrices during loss-augmented training.

(user-entity and mention-entity) described in Equation 5 incrementally. Our main evaluation uses the RETWEET+ network, since the retweet network had the greatest entity homophily; an additional evaluation compares across network types.

Parameter tuning We tune all the hyper-parameters on the development set, and then re-train the models on the full training data with the best parameters. We choose the number of hidden units for the MLP from $\{20, 30, 40, 50\}$, and the regularization penalty for our composition model from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. The sizes of user embeddings and word embeddings are selected from $\{50, 100\}$ and $\{200, 400, 600\}$ respectively. The pre-trained Freebase entity embedding size is 1000. The learning rate for the SGD algorithm is set as 0.01. During training, we check the performance on the development set regularly to perform early stopping.

5.3 Results

Table 4 summarizes the empirical findings for our approach and S-MART (Yang and Chang, 2015) on the tweet entity linking task. For the systems with user-entity bilinear function, we report results obtained from embeddings trained on RETWEET+ in Table 4, and other results are available in Table 5. The best hyper-parameters are: the number of hidden units for the MLP is 40, the L2 regularization penalty for the composition parameters is 0.005, and the user embedding size is 100. For the word embedding size, we find 600 offers marginal improvements over 400 but requires longer training time. Thus, we choose 400 as the size of word embeddings.

As presented in Table 4, NTEL-nonstruct performs 2.7% F1 worse than the NTEL baseline on the two test sets, which indicates the non-overlapping inference improves system performance on the task. With structured inference but without embeddings, NTEL performs roughly the same as S-MART, showing that a feedforward neural network offers similar expressivity to the regression trees employed by Yang and Chang (2015).

Performance improves substantially with the incorporation of low-dimensional author, mention, and entity representations. As shown in Table 4, by learning the interactions between mention and entity representations, NTEL with mention-entity bilinear function outperforms the NTEL baseline system by 1.8% F1 on average. Specifically, the bilinear function results in considerable performance gains in recalls, with small compromise in precisions on the datasets.

Social information helps to increase about 1% F1 on top of both the NTEL baseline system and the NTEL system with mention-entity bilinear composition. In contrast to the mention-entity composition model, which mainly focuses on improving the baseline system on recall scores, the user-entity composition model increases around 2.5% recalls, without much sacrifice in precisions.

Our best system achieves the state-of-the-art results on the NEEL-test dataset and the TACL dataset, outperforming S-MART by 0.9% and 5.4% F1 scores respectively. To establish the statistical significance of the results, we obtain 100 bootstrap samples for each test set, and compute the F1 score on each sample for each algorithm. Two-tail paired t-test is then applied to determine if the F1 scores of two algorithms are significantly different. NTEL significantly outperforms S-MART on the NEEL-test dataset and the TACL dataset under $p < 0.01$ level, with t-statistics equal to 11.5 and 33.6 respectively.

As shown in Table 5, MENTION+ and RETWEET+ perform slightly better than FOLLOWER+. Puniyani et al. (2010) show that the mention network has stronger linguistic properties than the follower network, as it gives better correlations on each author's distribution over latent topics as induced by latent Dirichlet allocation (Blei et al., 2003). Our results suggest that the properties hold with respect to the authors' interests on real world entities.

5.4 Error Analysis & Discussion

We examine the outputs of different systems, focusing on investigating what errors are corrected by the two bilinear functions. The results reveal that the mention-entity composition improves the system ability to tackle mentions that are abbreviations such as 'WSJ' (*The Wall Street Journal*) and 'SJSU' (*San Jose State University*), which leads to higher recall scores. The mention-entity model also helps to eliminate errors that incorrectly link non-entities to popular entities. For example, the NTEL baseline

System	user -entity	mention -entity	NEEL-test			TACL			Avg. F1
			P	R	F1	P	R	F1	
Our approach									
NTEL-nonstruct			80.0	68.0	73.5	64.7	62.3	63.5	68.5
NTEL			**82.8**	69.3	75.4	68.0	66.0	67.0	71.2
NTEL	✓		82.3	71.8	76.7	66.9	68.7	67.8	72.2
NTEL		✓	80.2	**75.8**	77.9	66.9	**69.3**	68.1	73.0
NTEL	✓	✓	81.9	75.6	**78.6**	**69.0**	69.0	**69.0**	**73.8**
Best published results									
S-MART			80.2	75.4	77.7	60.1	67.7	63.6	70.7

Table 4: Evaluation results on the NEEL-test and TACL datasets for different systems. The best results are in **bold**.

Network	NEEL-test			TACL		
	P	R	F1	P	R	F1
FOLLOWER+	82.2	75.1	78.5	67.8	68.7	68.2
MENTION+	**82.5**	**76.0**	**79.1**	67.5	**69.3**	68.4
RETWEET+	81.9	75.6	78.6	**69.0**	69.0	**69.0**

Table 5: Comparison of different social networks with our full model. The best results are in **bold**.

system links 'sec' in the tweet 'I'm a be in Miami for sec to hit da radio!' to *Southeastern Conference*, which is corrected by the mention-entity composition model. The word semantic information encoded in the mention representations alleviates the biased entity information given by the surface features.

The user-entity composition model is good at handling highly ambiguous mentions. For example, our full model successfully disambiguates entities for mentions such as 'Sox' (*Boston Red Sox* vs. *Chicago White Sox*), 'Sanders' (*Bernie Sanders* vs. *Barry Sanders*), and 'Memphis' (*Memphis Grizzlies* vs. *Memphis, Tennessee*), which are mistakenly linked to the other entities or **Nil** by the mention-entity model. Another example is that the social network information helps the system correctly link 'Kim' to *Lil' Kim* instead of *Kim Kardashian*, despite that the latter entity's wikipedia page is considerably more popular.

6 Related Work

Tweet entity linking Previous work on entity linking mainly focuses on well-written documents (Bunescu and Pasca, 2006; Cucerzan, 2007; Milne and Witten, 2008), where entity disambiguation is usually performed by maximizing the global topical coherence between entities. However, these approaches often yield unsatisfactory performance on Twitter messages, due to the short and noisy nature of the tweets. To tackle this problem, collective tweet entity linking methods that leverage enriched context and metadata information have been proposed (Huang et al., 2014). Guo et al. (2013b) search for textually similar tweets for a target tweet, and encourage these Twitter messages to contain similar entities through label propagation. Shen et al. (2013) employ Twitter user account information to improve entity linking, based on the intuition that all tweets posted by the same user share an underlying topic distribution. Fang and Chang (2014) demonstrate that spatial and temporal signals are critical for the task, and they advance the performance by associating entity prior distributions with different timestamps and locations. Our work overcomes the difficulty by leveraging social relations — socially connected individuals are assumed to share similar interests on entities. As the Twitter post information is often sparse for some users, our assumption enables the utilization of more relevant information that helps to improve the task.

NLP with social relations Most previous work on incorporating social relations for NLP problems focuses on Twitter sentiment analysis, where the existence of social relations between users is considered as a clue that the sentiment polarities of messages from the users should be similar. Speriosu et al. (2011) construct a heterogeneous network with tweets, users, and n-grams as nodes, and the sentiment label distributions associated with the nodes

are refined by performing label propagation over social relations. Tan et al. (2011) and Hu et al. (2013) leverage social relations for sentiment analysis by exploiting a factor graph model and the graph Laplacian technique respectively, so that the tweets belonging to social connected users share similar label distributions. We work on entity linking in Twitter messages, where the label space is much larger than that of sentiment classification. The social relations can be more relevant in our problem, as it is challenging to obtain the entity prior distribution for each individual.

7 Conclusion

We present a neural based structured learning architecture for tweet entity linking, leveraging the tendency of socially linked individuals to share similar interests on named entities — the phenomenon of *entity homophily*. By modeling the compositions of vector representations of author, entity, and mention, our approach is able to exploit the social network as a source of contextual information. This vector-compositional model is combined with nonlinear feature combinations of surface features, via a feedforward neural network. To avoid predicting overlapping entity mentions, we employ a structured prediction algorithm, and train the system with loss-augmented decoding.

Social networks arise in other settings besides microblogs, such as webpages and academic research articles; exploiting these networks is a possible direction for future work. We would also like to investigate other metadata attributes that are relevant to the task, such as spatial and temporal signals.

Acknowledgments We thank the EMNLP reviewers for their constructive feedback. This research was supported by the National Science Foundation under awards IIS-1111142 and RI-1452443, by the National Institutes of Health under award number R01-GM112697-01, and by the Air Force Office of Scientific Research.

References

Sitaram Asur and Bernardo A Huberman. 2010. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 492–499.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

R. C Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Amparo E Cano, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2014. Making sense of microposts (# microposts2014) named entity extraction & linking challenge. *Making Sense of Microposts (# Microposts2014)*.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. Erd'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, pages 63–77.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Yuan Fang and Ming-Wei Chang. 2014. Entity linking on microblogs with spatial and temporal signals. *Transactions of the Association for Computational Linguistics (ACL)*.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013a. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Atlanta, GA.

Yuhang Guo, Bing Qin, Ting Liu, and Sheng Li. 2013b. Microblog entity linking by leveraging extra posts. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Seattle, WA.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 752–762, Beijing, China.

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the sixth ACM international conference on Web search and data mining (WSDM)*, pages 537–546.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media? In *Proceedings of the Conference on World-Wide Web (WWW)*, pages 591–600, New York. ACM.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *arXiv preprint arXiv:1510.05198*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO.

Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD)*, pages 1155–1158.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444.

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

D. Milne and I. H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 380–390, Atlanta, GA.

Kriti Puniyani, Jacob Eisenstein, Shay Cohen, and Eric P. Xing. 2010. Social links from latent topics in microblogs. In *Proceedings of NAACL Workshop on Social Media*, Los Angeles.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2013. Linking named entities in tweets with knowledge base via user interest modeling. In *Proceedings of Knowledge Discovery and Data Mining (KDD)*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Neural Information Processing Systems (NIPS)*, Lake Tahoe.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 53–63.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of Knowledge Discovery and Data Mining (KDD)*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the Conference on World-Wide Web (WWW)*.

Shaomei Wu, Jake M Hofman, Winter A Mason, and Duncan J Watts. 2011. Who says what to whom on twitter. In *Proceedings of the Conference on World-Wide Web (WWW)*, pages 705–714.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the Association for Computational Linguistics (ACL)*, Beijing, China.

Yi Yang and Jacob Eisenstein. 2015. Putting things in context: Community-specific embedding projections for sentiment analysis. *arXiv preprint arXiv:1511.06052*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Association for Computational Linguistics (ACL)*, Beijing, China.

Phonologically Aware Neural Model for Named Entity Recognition in Low Resource Transfer Settings

Akash Bharadwaj David Mortensen Chris Dyer Jaime G. Carbonell
{akashb, dmortens, cdyer, jgc}@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University

Abstract

Named Entity Recognition is a well established information extraction task with many state of the art systems existing for a variety of languages. Most systems rely on language specific resources, large annotated corpora, gazetteers and feature engineering to perform well monolingually. In this paper, we introduce an attentional neural model which only uses language universal phonological character representations with word embeddings to achieve state of the art performance in a monolingual setting using supervision and which can quickly adapt to a new language with minimal or no data. We demonstrate that phonological character representations facilitate cross-lingual transfer, outperform orthographic representations and incorporating both attention and phonological features improves statistical efficiency of the model in 0-shot and low data transfer settings with no task specific feature engineering in the source or target language.

1 Introduction

Named Entity Recognition (NER) (Nadeau and Sekine, 2007; Marrero et al., 2013) is an information extraction task that deals with finding and classifying entities in text into a fixed set of types of interest. It is challenging for a variety of reasons. Named Entities (NEs) can be arbitrarily synthesized (eg. people's/organization's names). NEs are often not subject to uniform cross-linguistic spelling conventions: compare *France* (English) and *Frantsiya* (Uzbek). NEs occur rarely in data which makes generalization difficult. Skewed class statistics necessitate measures to prevent models from merely favoring a majority class.

Named entities must also be annotated in context (eg. "[New York Times]$_{\text{ORG}}$" vs. "[New York]$_{\text{LOC}}$"). Lexical ambiguity (Turkey—country vs. bird), semantic ambiguity ("I work at the [New York Times]$_{\text{ORG}}$" vs. "I read the New York Times") and sparsity induced by morphology add complexity.

Despite the challenges mentioned above, competent monolingual systems that rely on having sufficient annotated data, knowledge and resources available for engineering features have been developed. A more challenging task is to design a model that retains competence in monolingual scenarios and can easily be transferred to a low resource language with minimum overhead in terms of data annotation requirements and feature engineering. This transfer setting introduces additional challenges such as varying character usage conventions across languages with same script, differing scripts, lack of NE transliteration, varying morphology, different lexicons and mutual non-intelligibility to name a few.

We propose the following changes over prior work (Lample et al., 2016) to address the challenges of the low-resource transfer setting. We use:

1. Language universal phonological character representations instead of orthographic ones.

2. Attention over characters of a word while labeling it with an NE category.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1462–1472,
Austin, Texas, November 1-5, 2016.

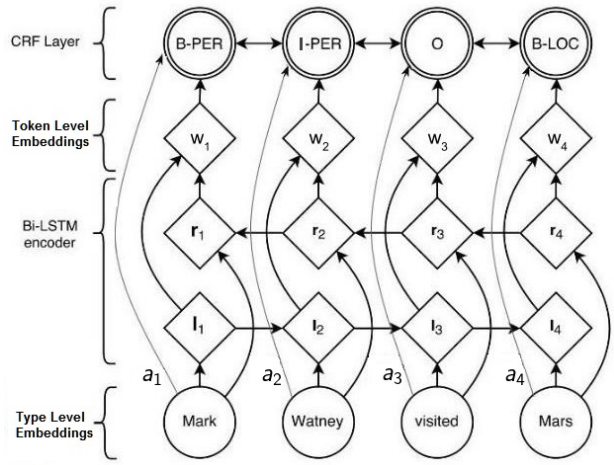


Figure 1: Attentional LSTM-CRF architecture. l_i denotes the encoding of a word and its left context (forward LSTM) while r_i includes only right context (backward LSTM). Inputs to word LSTMs are obtained using character LSTMs and word-embeddings. a_i denotes an attentional context vector concatenated with l_i and r_i.

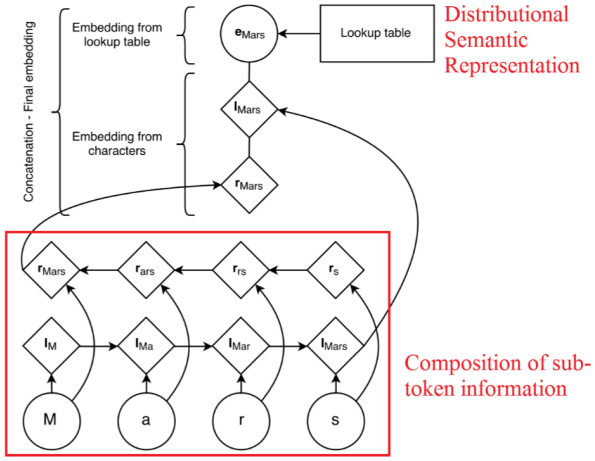


Figure 2: Type level word representations - l denotes a word prefix encoding (by forward char LSTM) while r denotes a word suffix encoding (by backward char LSTM).

We show that using phonological character representations instead does not negatively impact performance on two languages: Spanish and Turkish. We then show that using global phonological representations enables model transfer from one or more source languages to a target language with no extra effort, even when the languages use different scripts. We demonstrate that while attention over characters of words has marginal utility in monolingual and high resource settings, it greatly improves the statistical efficiency of the model in 0-shot and low resource transfer settings. We do require a mapping from a language's script to phonological feature space which is script specific and not task specific. This presents little or no overhead due to existence of tools like PanPhon (Littell et al., 2016).

2 Our Approach

Figure 1 provides a high level overview of our model. We model the words of a sentence at the type level and the token level. At the type level (ignorant of sentential context), we use bidirectional character LSTMs as in figure 2 to compose characters of a word to obtain its word representation and concatenate this with a word embedding that captures distributional semantics. This can memorize entities or capture morphological and suffixal clues that can help at a discriminative task like NER. We compose type level word representations with bidirectional LSTMs to obtain token level (cognizant of sentential context) representations. Using token level word representations along with an attentional context vector for each word based on the sequence of characters it contains, we generate score functions used by a Conditional Random Field (CRF) for inference. To facilitate transfer across languages with different scripts, we use Epitran [1] and PanPhon (Littell et al., 2016).

Epitran is a straightforward orthography-to-IPA (International Phonetic Alphabet [language universal]) system including a collection of preprocessors and grapheme-to-phoneme mappings for a variety of language-script pairs. It converts a word from its native script into a sequence of IPA characters, each of which approximately corresponds to a phoneme. PanPhon is a database of IPA-to-phonological feature vector mappings and a library for querying, manipulating, and exploiting this database. It consumes the output of Epitran and produces feature vectors (21 dimensions) of phonological characteristics such as whether a phoneme is articulated with (accompanied by) vibration of the vocal folds (voiced), with the tongue in a high, low, back, or front position, with the lips rounded or unrounded, with tongue tip or blade (coronal), etc. Figure 3 depicts the sequence of operations applied to the same NE

[1] https://github.com/dmort27/epitran

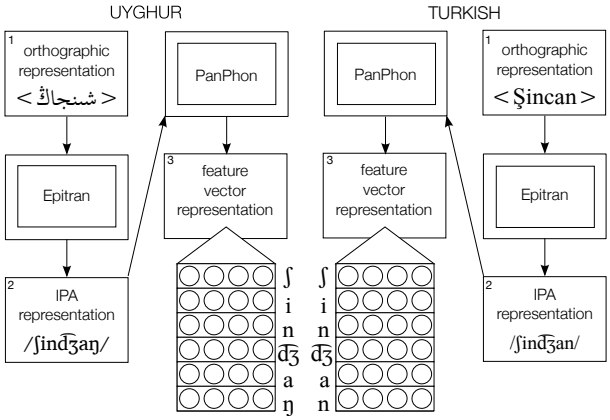


Figure 3: Use of Epitran and PanPhon to enable transfer across orthographies

in Uyghur (Perso-Arabic script) and Turkish (Latin script), thus making the equivalence across scripts apparent. We concatenate the feature vectors from PanPhon and 1-hot encodings of the corresponding IPA characters and use these as inputs to the character bi-LSTMs.

This shift to IPA space is motivated by prior work (Tsvetkov et al., 2015; Tsvetkov and Dyer, 2015) which demonstrated the value of projecting orthographic surface forms of words into a phonological space for detecting loan words, transliteration and cognates even in language pairs that exhibit significant typological, morphological and phonological differences. Our underlying assumption is that named entities are likely to be transliterated or retain pronunciation patterns across languages. Additionally, phenomena such as vowel harmony manifest explicitly in IPA representation and can potentially be helpful for NER. Foreign named entities for example, need not obey vowel harmony rules prevalent in languages like Turkish. A powerful sequence model such as a LSTM could be tolerant to the noise created by lexical aberrations, lack of spelling conventions, vowel raising etc. when given a phonological representation of an NE in different languages.

Our second proposed change is to incorporate attention over the sequence of IPA segments in a word when predicting scores for its possible labels. Attention is an unsupervised alternative to convolution or feature engineering to capture helpful localized phenomena like capitalization of first letter, presence of case markers, special characters, helpful morphological suffixes etc. or the conjunction of multiple such phenomena. Such features have been the mainstay of most prior work for NER. Most of these features are subtle and occur at the type level, whereas the CRF performs inference at the token level. We show (empirically) that attention makes the CRF more sensitive to such useful type level phenomena during inference and improves the statistical efficiency of the model in certain scenarios. Having described our intuitions, we now provide mathematical details of our model.

2.1 Model Description

2.1.1 LSTM

Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) belongs to a special breed of neural networks called Recurrent Neural Networks (RNNs) which are capable of processing sequential input of unbounded and arbitrary length. This makes them suitable for a sequence labeling task. LSTMs incorporate gating functions at each time step to allow the network to forget, remember and update contextual memory and mitigate problems like vanishing gradient. We use the following implementation:

$$
\begin{aligned}
i_t =& \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\
c_t =& (1 - i_t) \odot c_{t-1} + \\
& i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t =& \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\
h_t =& o_t \odot \tanh(c_t)
\end{aligned}
$$

where $\odot$ indicates element-wise product and σ indicates element-wise sigmoid function.

In practice we use bi-directional LSTMs (each with its own parameters) to mitigate cases where the LSTM may be biased towards the last few inputs it reads. This is done both at the word-level and the character level. Let the hidden state at time step **t** of the forward LSTM be denoted by $\overrightarrow{h_t}$ and the corresponding state of the backward LSTM be denoted by $\overleftarrow{h_t}$. At the character level, for a word with L characters, we only take the final hidden states in each direction i.e. $[\overrightarrow{h_L}; \overleftarrow{h_0}]$ and concatenate them to get a representation of the word that comprises these characters. At the word level, we concatenate corresponding forward and backward LSTM states for each word X_t to get $[\overrightarrow{h_t}; \overleftarrow{h_t}]$ which is the token

level representation for the t^{th} word in a sentence X. We use this to generate un-normalized energy/score functions for the CRF layer.

2.2 Attention

Let $w_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$ indicate the concatenated word bi-LSTM output (of dimension d_1) at step t corresponding to the t^{th} word (X_t) in the input sequence X. Let M_t be the matrix containing the concatenated bi-directional character LSTM outputs for each character of X_t. It has dimensions (l_t, d_2) where d_2 is the dimension of the concatenated bi-directional character LSTM hidden states and l_t is the number of characters in X_t. Let m_t^i denote the i^{th} row of M_t. Let P be a parameter matrix of dimension (d_1, d_2) and *p* be a bias vector of length d_2. We follow (Bahdanau et al., 2014) in the formulation of attention context vector a_t:

$$w'_t = \tanh(w_t \cdot P + p)$$
$$\alpha_i = \frac{\exp(w'_t \cdot m_t^i)}{\sum_{j=1}^{l_t} \exp(w'_t \cdot m_t^j)}$$
$$a_t = \sum_{i=1}^{l_t} (\alpha_i * m_t^i)$$

The attentional context vector a_t is then appended to w_t to obtain concatenated vector u_t = $[a_t; w_t]$. We apply a linear transform U (matrix of dimension $(d_1 + d_2, k)$ where *k* is the number of unique NER tags). This gives us:

$$e_t = u_t \cdot U \tag{1}$$

where e_t is a vector of un-normalized energy/score functions indicating the compatibility between word X_t and each of the *k* possible NER labels it can be given. Note that each word has a separate attention context vector obtained using the character LSTM hidden states generated by its constituent characters.

2.3 Conditional Random Field

Unlike Hidden Markov Models, CRFs do not enforce any independence assumptions among observed data and directly model the likelihood of a labeling hypothesis discriminatively. They also model adjacency compatibility between NER tags which can capture strong constraints like an 'I-label' tag not following an O tag without a 'B-label' tag in between them (see section 2.6). In our model, the CRF is parametrized as follows:

For a word sequence X = $(x_1, x_2, ...x_N)$, let **E** be a matrix of dimension (k,N) where k is the number of unique NER tags and N is the sequence length. The t^{th} column is the vector e_t obtained in equation 1. Let **T** be the square transition matrix of size (k+2, k+2) which captures transition score between the k NER tags, the start and the end symbols. Let Y = $(y_1, y_2, ...y_N)$ be the label sequence associated with the input word sequence, each y_i being an index into the ordered set of unique NER tags. Let y_0 be the start symbol and y_{N+1} be the end symbol. The score of this sequence is evaluated as:

$$S(X, Y) = \sum_{i=1}^{N} E_{y_i, i} + \sum_{j=0}^{N} T_{y_j, y_{j+1}}$$

Let $\mathcal{Y}_\mathcal{X}$ indicate the exponential space of all possible labelings of this sequence X. The partition function associated with this CRF is then evaluated as:

$$Z = \sum_{Y \in \mathcal{Y}_\mathcal{X}} e^{S(X,Y)}$$

The probability of a specific labeling $Y \in \mathcal{Y}_\mathcal{X}$:

$$P(Y|X) = \frac{e^{S(X,Y)}}{Z}$$

The training objective is to maximize conditional log probability of the correct labeling sequence Y^*:

$$log(P(Y^*|X)) = S(X, Y^*) - log\,(Z) \tag{2}$$

The decoding criteria for an input sequence X is:

$$Y^* = \underset{Y \in \mathcal{Y}_\mathcal{X}}{\arg\max}\, S(X, Y) \tag{3}$$

Normally, evaluating the partition function over the exponential space of all possible labelings would be intractable. However, as described in (Lafferty et al., 2001), this can be done efficiently for linear chain CRFs using the forward backward algorithm.

2.4 Word Representations

The inputs to our model are in the form of type level word representations (figure 2). Motivated by the distributional hypothesis (Harris, 1954; Firth, 1957)

we use word embeddings as inputs. In the monolingual scenario, we use structured skipgram word embeddings (Ling et al., 2015a). For the transfer scenario, embeddings can optionally be trained using techniques like multi CCA described in (Ammar et al., 2016). By learning a linear transformation from a shared vector space between languages, the model may acquire some transfer capability to the target language.

We use character bi-LSTMs to handle the Out Of Vocabulary (OOV) problem as in (Ling et al., 2015b). However, just as a distributional hypothesis exists for words, prior work (Tsvetkov and Dyer, 2015; Tsvetkov et al., 2015) suggests phonological character representations capture inherent similarities between characters that are not apparent from orthogonal one-hot orthographic character representations and can serve as a language universal surrogate for character representations. This is also useful for multi-lingual named entity recognition as explained in section 2. Therefore we make use of Epitran and PanPhon as in figure 3 to obtain both 1-hot IPA character encodings and phonological feature vectors capturing similarity between IPA characters. These form the inputs to the character bi-LSTMs. The mapping from orthographic character segments to IPA is sometimes many to one (ambiguous) and unarticulated characters (like periods in NE abbreviations) and capitalization information is lost by default. Given the importance of such orthographic features for NER and the ambiguity introduced, a drop in performance may be expected by using phonological character representations.

2.5 Training

Our model parametrization is similar to (Lample et al., 2016). The weights to be trained include the the CRF transition matrix **T**, the projection parameters are used to generate matrix **E** (P and U), the LSTM parameters and word and character embedding matrices. The objective is to maximize the log probability of the correct labeling sequence as given in equation 2. This objective is fully differentiable and standard back-propagation is used to learn weights. We use Stochastic Gradient Descent with a learning rate of 0.05 and gradients clipped at 5.0 providing best performance. Using Adadelta or Adam leads to faster convergence but slightly worse performance. Word level LSTMs use a hidden layer size of 100, orthographic character LSTMs (if used) used a hidden layer of size 25 while phonological character LSTMs used a hidden layer of size 15 due to the smaller phonetic alphabet. Tuning these did not have a major effect on performance. Dropout of 0.5 is applied after concatenation of the word embeddings and character LSTM outputs. Best dropout value was chosen through ablation studies. For Spanish, we use word embeddings pre-trained on the Spanish Gigaword version 3. For transfer experiments, we use multilingual word embeddings trained using multi CCA described in (Ammar et al., 2016).

2.6 Entity Types and Tagging Schemes

In all of the datasets in table 1, 4 entity types are annotated:

1. **Persons (PER)**
   Real/fictional, living/dead people. Aliases and family names are also annotated. E.g. Barack Obama, the [Kennedys], Puff Daddy etc.
2. **Locations (LOC)**
   Geographical locations without a dedicated population and government. E.g. Nile river, Sahara desert, Mt. Everest, Asia etc.
3. **Geo-Political Entities (GPE)**
   Geographical regions with corresponding population and government. Mentions can be nominal (e.g. India, E.U., Britain etc.) or adjectival (e.g. [British] army, [French] government etc.).
4. **Organizations (ORG)**
   Names of entities with organization and managerial structure. E.g. Democratic Party, Google, JetBlue, etc.

A BIO tagging scheme is used for all annotations. All non-entity tokens are annotated as 'O'. The first token of an entity span is annotated as 'B-label' and all remaining tokens in the entity span are annotated as 'I-label'.

3 Experiments

We conduct four different experiments:

1. CoNLL 2002 Spanish NER (Sang., 2002) task. This demonstrates the monolingual competence of phonological character representations vs. orthographic representations.

2. Turkish NER using the Linguistic Data Consortium's LDC2014E115 BOLT Turkish Language Pack [2]. This demonstrates the utility of phonological character representations and attention in a morphologically rich, low resource language. We compare against a state-of-the-art monolingual model (Lample et al., 2016) that uses orthographic character LSTMs.
3. Transfer Experiments from Uzbek to Turkish using LDC2014E112 BOLT [3] data pack for Uzbek and LDC2014E115 BOLT data pack for Turkish. These two languages have similar syntax and word order (Dryer, 2013) but vary significantly in morphology, vowel harmony and phonology, can use different scripts (Uzbek-Latin/Cyrilic, Turkish-Latin) and are not mutually intelligible.
4. Transfer Experiments from Uzbek and Turkish into Uyghur using LDC2014E112 and LDC2014E115 BOLT data pack for Uzbek and Turkish respectively and Uyghur data provided as part of DARPA LORELEI[4]. These languages all have different scripts, lexicons, morphology and phonology. Results are reported by NIST [5] on an unseen test set.

3.1 Results

Tables 2 and 3 report results from monolingual experiments in Spanish. In table 3, we report the performance of our best model against other state-of-the-art models for the Spanish CoNLL 2002 NER task (Sang., 2002). Our model performs marginally better than other benchmarks with the optimal configuration of hyper-parameters and using pre-trained word embeddings. Table 2 report ablation study results, which reveal that using pre-trained word embeddings without using character LSTMs yields a very strong baseline that already out-performs various previous benchmarks. Using character LSTMs that compose orthographic character representations yields a +0.91 improvement in F1 score and a further

[2]http://opencatalog.darpa.mil/BOLT.html

[3]BOLT contains newswire, discussion forum, social media and chat data

[4]http://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

[5]https://www.nist.gov

[6]Sparse features for character capitalization and character type (digit, punctuation etc.)

Language	# Sentences	# Entities
Spanish	8323	18798
Turkish	5065	4883
Uzbek	10078	7960
Uyghur	2161	2668*

Table 1: Dataset Statistics. * indicates non-gold annotations produced by a non-speaker linguist.

Phono Chars	Ortho Chars	Word Vecs	Cap+ Cat[6]	Ortho Attn	Phono Attn	F1
No	No	Yes	No	No	No	83.61
No	Yes	Yes	No	No	No	84.52
No	Yes	Yes	No	Yes	No	84.64
No	Yes	Yes	Yes	No	No	84.91
No	Yes	Yes	Yes	Yes	No	85.25
Yes	No	Yes	No	No	No	84.08
Yes	No	Yes	No	No	Yes	84.88
Yes	No	Yes	Yes	No	No	84.89
Yes	No	Yes	Yes	No	Yes	**85.81**
Yes	Yes	Yes	No	No	Yes	84.53
Yes	Yes	Yes	Yes	No	No	84.92
Yes	Yes	Yes	Yes	Yes	Yes	84.75
Yes	Yes	Yes	Yes	No	Yes	84.84
Yes	Yes	Yes	Yes	Yes	No	85.32

Table 2: Ablation Tests on Spanish CoNLL 2002. **Bold** indicates the best model.

Model	F1
Carreras et al. (2002)*	81.39
dos Santos et al. (2015)	82.21
Gillick et al. (2015)	81.83
Gillick et al. (2015)*	82.95
Lample et al. (2016)	85.75
Yang et al. (2016)	85.77
Our Best	**85.81**

Table 3: Comparison with benchmarks. * indicates a model that uses external labeled data

Phono Chars	Ortho Chars	Word vecs	Cap+ Cat	Ortho Attn	Phono Attn	F1
No	No	Yes	No	No	No	49.2
No	Yes	Yes	No	No	No	65.41
No	Yes	Yes	No	Yes	No	64.76
No	Yes	Yes	Yes	No	No	60.57
No	Yes	Yes	Yes	Yes	No	60.87
Yes	No	Yes	No	No	No	63.04
Yes	No	Yes	No	No	Yes	**66.07**
Yes	No	Yes	Yes	No	No	59.08
Yes	No	Yes	Yes	No	Yes	62.46
Yes	Yes	Yes	No	No	Yes	63.43
Yes	Yes	Yes	Yes	No	No	63.46
Yes	Yes	Yes	Yes	Yes	Yes	**66.47**

Table 4: Ablation Tests on Turkish **Bold** indicates the best transfer eligible (66.07) and transfer ineligible models (66.47)

Model	F1
Lample et al. (2016)	61.11
Lample et al. (2016) with pretrained embeddings	65.41
Our Best model	**66.47**
Our Best transfer-eligible model	**66.07**

Table 5: Comparison with state-of-the-art monolingual Turkish model

Model	F1
Lample et al. (2016)	37.1
Our best transfer model*	**51.2**

Table 6: NIST evaluations for Uyghur. * indicates transfer from Uzbek and Turkish

+0.12 F1 with attention. Using phonological character representations instead yields an improvement of +0.47 F1 and a further +0.8 F1 with attention. Thus, phonological representations benefit more from attention applied over them to beat out orthographic representations in that scenario. Using sparse features indicating the character category (alphabet vs. number vs. punctuation/non-phonetic) and capitalization in conjunction with with phonological character representations and word embeddings with attention over phonological characters yields the best configuration that slightly outperforms the best published models so far. Given that many previous benchmarks used features that rely heavily on orthography, this is an encouraging result since one would expect to lose some performance by using more abstract phonological representations as explained in section 2.4.

Tables 4 and 5 highlight results from monolingual experiments on Turkish. This dataset is much more challenging since the annotated training courpus is significantly smaller than the CoNLL 2002 shared task dataset and because Turkish is an agglutinative language exhibiting sparsity inducing morphology which leads to huge vocabulary size. As a competitive baseline, we train the LSTM CRF described in (Lample et al., 2016) due to its documented ability to obtain state-of-the-art monolingual results for many languages with minimal feature engineering. Our best model from the Turkish ablation study outperforms this baseline. We also see a stark contrast between the ablation study results for Turkish compared to Spanish. Firstly, word embeddings alone perform rather poorly due to the challenges of reliably estimating them for a large vocabulary given a small dataset. Character composed representations of words provide a significant performance boost (+17.27 F1 for the best model). Secondly, usage of sparse character features (like capitalization) seems to hurt performance in all but the last model in table 4. Thirdly, phonological and orthographic character representations are complementary in the case of Turkish, unlike Spanish. This is not too surprising since Turkish exhibits phonological phenomena like vowel harmony. Lack of vowel harmony in a word could give-away a foreign word or a named entity for example. Results show that attention is helpful as well. We would also like to point out that the only models in the ablation studies eligible for transfer are those that do not use orthographic character representations. Among these, the model that uses phonological representation with attention and word vectors performs the best and also outperforms the baseline system.

Our next experiments on model transfer are arguably the most interesting. Tables 7 and 8 document single source (Uzbek to Turkish) transfer results. We find that using sparse character category and capitalization features in conjunction with attention and phonological features yields the best 0-shot transfer performance (no training labels in the target language). Specifically, attention provides +6 F1 in 0-shot and 5% labeled-target language data scenarios. It is interesting to note that using multilingual word embeddings for the source and target languages alone accounts for very poor transfer. We also find that with as little as 20% of the data, we approach the performance of a monolingual target model that was trained on all the data. We also notice that the transfer models retain a consistent advantage over a monolingually trained target language model across all data availability scenarios. Lastly, we note that while attention provides a significant improvement in 0-shot and 5% data availability scenarios, a model without attention (or sparse features like capitalization) eventually does better with more data. This indicates that the model is competent enough to leverage transfer via phonology alone. This could also possibly be because attention patterns from Uzbek could bring in a bias that is eventually sub-optimal for Turkish due to dif-

Phono Chars	Word vecs	Cap+ Cat	Phono Attn	Uzbek Source F1	Target 0-shot F1	5% data	20% data	40% data	60% data	80% data	All data
No	Yes	No	No	41.87	2.09	23.44	35	42.75	46.32	48.81	50.34
Yes	Yes	No	Yes	61.24	11.9	34.06	47.84	56.1	53.5	64.72	65.2
Yes	Yes	No	No	60.92	15.55	39.42	**60.14**	**63.23**	**62.54**	**65.24**	**65.63**
Yes	Yes	Yes	No	64.89	22.14	41.19	54.02	57.06	59.26	60.84	61.72
Yes	Yes	Yes	Yes	61.85	**26.92**	**47.21**	58.58	60.32	60.7	62.84	63.58

Table 7: Model Transfer from Uzbek (Source) to Turkish (Target) at different target data availability thresholds

Model	0-shot	5% data	20% data	40% data	60% data	80% data	All data
LSTM-CRF (Lample et al., 2016)	0	33.44	50.61	53.25	57.41	60	61.11
S-LSTM (Lample et al., 2016)	0	15.41	39.33	42.99	51.92	51.55	56.58

Table 8: Monolingual Turkish baseline at different data availability thresholds

ferent morphology and phonology. In future work, we shall perform more insightful error analysis to explain these trends.

Table 6 documents NIST evaluation results on an unseen Uyghur test set (with gold annotations) for the best transfer model configuration jointly trained on Turkish and Uzbek gold annotations and Uyghur training annotations produced by a non-speaker linguist (non-gold). Since Uyghur lacks helpful type-level orthographic features such as capitalization, our transfer model in table 6 does not use any sparse features or attention but benefits from transfer via the phonological character representations we've proposed. Despite the noisy supervision provided in the target language, transferring from Turkish and Uzbek provides a +14.1 F1 improvement over a state of the art monolingual model trained on the same Uyghur annotations. It is worth pointing out that this transfer was achieved across 3 languages each with different scripts, morphology, phonology and lexicons.

4 Prior Work

NER is a well-studied sequence-labeling problem for which many different approaches have been proposed. Early works had a monolingual focus and relied heavily on feature engineering. Approaches include maximum entropy models (Chieu and Ng, 2003), hierarchically smoothed tries (Cucerzan and Yarowsky, 1999), decision trees (Carreras et al., 2002) and models incorporating syntactic, semantic and world knowledge (Wakao et al., 1996). Each of these models brings in a bias of its own. Florian et al. (2003) successfully tried ensembling multiple classifiers and improved performance.

Since NER is a sequence labeling problem, there are local dependencies both among NE labels associated with words and among the words themselves, that could aid the labeling process. To explicitly deal with these sequential characteristics, Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) became very popular. (Klein et al., 2003; Florian et al., 2003; McCallum and Li, 2003; Ratinov and Roth, 2009; Chandra et al., 1981; Lin and Wu, 2009; Lample et al., 2016; Yang et al., 2016; Ma and Hovy, 2016). CRFs eventually became more popular because they are discriminative models that directly model the required posterior probability of a labeling sequence using parametrized functions of features. They do not model the probability of the observed sentence itself, avoid Markovian independence assumptions made by HMMs and avoid the label bias problem.

Most of the work cited so far makes use of hand engineered features. The following approaches minimize the use of features while still maintaining a monolingual focus. Collobert et al. (2011), Turian et al. (2010), and Ando and Zhang (2005) use unsupervised features in conjunction with engineered features capturing capitalization, character categories and gazetteer matches. Collobert et al. (2011) use a Convolutional Neural Network (CNN) over the sequence of word embeddings. Huang et al. (2015) instead use bi-directional LSTMs over the sequence of words, along with spelling and orthographic features.

The most recent work eliminates feature engineering altogether and combines CRFs with LSTMs

which can model long sequences while remembering appropriate past context. Lample et al. (2016) proposed an architecture that uses both character and word level LSTMs to produce score functions for CRF inference conditioned on global context. Ma and Hovy (2016) replace the character LSTMs of Lample et al. (2016) with a CNN instead. Yang et al. (2016) follow a very similar architecture to Lample et al. (2016), replacing the LSTMs with Gated Recurrent Units (Cho et al., 2014). However, Yang et al. (2016) also tackle multi task and multi-lingual joint training scenarios.

Most of the models cited so far are monolingual either because they use hand crafted features and language specific resources or because of deep-seated assumptions. For example a change in orthography, lexicon, script, word order or addition of complex morphology makes transfer impossible. This is the central challenge that we tackle. There has been much less work catering to this scenario. Kim et al. (2012) use weak annotations from Wikipedia metadata and parallel data for multi lingual NER. Yang et al. (2016) addresses the use case of multilingual joint training, which assumes there is sufficient data available in all languages. Nothman et al. (2013) also operate under the assumption of availability of Wikipedia data.

To the best of our knowledge, a scenario involving transfer of a model trained in one (or more) source language(s) to another language with little or no labeled data, different script, different morphology, different lexicon, lack of transliteration, non-mutual intelligibility etc. has not been addressed recently.

5 Conclusion

In this paper, we presented two improvements over a state-of-the-art monolingual model to address Named Entity Recognition in transfer settings. The first seeks to reconcile various dimensions of variability between languages such as varying script, orthographic conventions, phonological phenomena etc. by representing words as sequences of IPA (International Phonetic Alphabet) segments consistent across all languages, rather than sequences of characters specific to a particular language. Secondly, we exploit the one-to-one mapping between input sequence words and output labels and advocate for the use of attention over the character/IPA sequence of a word when predicting its label. We show empirically that these two improvements 1) achieve at least state-of-the-art performance on a monolingual NER task in Spanish, 2) handle complex morphology in languages such as Turkish, Uzbek and Uyghur better than state of the art, 3) provide 0-shot performance in a transfer scenario to a related new language, well above that possible using multilingual word embeddings alone, and 4) are capable of generalizing to a new language with much less training data than a monolingually trained model. Moreover, all of this is achieved without any extra feature engineering specific to the task or language, apart from mapping characters in that language to IPA. We believe these results to be encouraging and look forward to replicating these results on more language pairs in different language families and further automating the process of obtaining phonological character representations.

6 Acknowledgement

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C-0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 160–163. Edmonton, Canada.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

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

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the 1999 Joint SIGDAT Conference on EMNLP and VLC*, pages 90–99.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Matthew S. Dryer, 2013. *Order of Subject, Object and Verb*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

John Rupert Firth. 1957. A synopsis of linguistic theory. In *In Studies in Linguistic Analysis*. Oxford: Philological Societ.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

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

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

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 694–702. Association for Computational Linguistics.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 180–183. Edmonton, Canada.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Patrick Littell, David Mortensen, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Bridge-language capitalization inference in western iranian: Sorani, kurmanji, zazaki, and tajik. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC16)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, and Juan Miguel Gómez-Berbís. 2013. Named entity recognition: fallacies,

challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482–489.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Languageindependent named entity recognition. *In Proc. CoNLL*.

Yulia Tsvetkov and Chris Dyer. 2015. Cross-lingual bridges with models of lexical borrowing. *JAIR*.

Yulia Tsvetkov, Waleed Ammar, and Chris Dyer. 2015. Constraint-based models of lexical borrowing. *NAACL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Takahiro Wakao, Robert Gaizauskas, and Yorick Wilks. 1996. Evaluation of an algorithm for the recognition and classification of proper names. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 418–423. Association for Computational Linguistics.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Long-Short Range Context Neural Networks for Language Modeling

Youssef Oualil[1,2] and **Mittul Singh**[1,3] and **Clayton Greenberg**[1,2,3] and **Dietrich Klakow**[1,2,3]
[1]Spoken Language Systems (LSV)
[2]Collaborative Research Center on Information Density and Linguistic Encoding
[3]Graduate School of Computer Science
Saarland University, Saarbrücken, Germany
{firstname.lastname}@lsv.uni-saarland.de

Abstract

The goal of language modeling techniques is to capture the statistical and structural properties of natural languages from training corpora. This task typically involves the learning of short range dependencies, which generally model the syntactic properties of a language and/or long range dependencies, which are semantic in nature. We propose in this paper a new multi-span architecture, which separately models the short and long context information while it dynamically merges them to perform the language modeling task. This is done through a novel recurrent Long-Short Range Context (LSRC) network, which explicitly models the local (short) and global (long) context using two separate hidden states that evolve in time. This new architecture is an adaptation of the Long-Short Term Memory network (LSTM) to take into account the linguistic properties. Extensive experiments conducted on the Penn Treebank (PTB) and the Large Text Compression Benchmark (LTCB) corpus showed a significant reduction of the perplexity when compared to state-of-the-art language modeling techniques.

1 Introduction

A high quality Language Model (LM) is considered to be an integral component of many systems for speech and language technology applications, such as machine translation (Brown et al., 1990), speech recognition (Katz, 1987), etc. The goal of an LM is to identify and predict probable sequences of predefined linguistic units, which are typically words. These predictions are typically guided by the semantic and syntactic properties encoded by the LM.

In order to capture these properties, classical LMs were typically developed as fixed (short) context techniques such as, the word count-based methods (Rosenfeld, 2000; Kneser and Ney, 1995), commonly known as N-gram language models, as well as the Feedforward Neural Networks (FFNN) (Bengio et al., 2003), which were introduced as an alternative to overcome the exponential growth of parameters required for larger context sizes in N-gram models.

In order to overcome the short context constraint and capture long range dependencies known to be present in language, Bellegarda (1998a) proposed to use Latent Semantic Analysis (LSA) to capture the global context, and then combine it with the standard N-gram models, which capture the local context. In a similar but more recent approach, Mikolov and Zweig (2012) showed that Recurrent Neural Network (RNN)-based LM performance can be significantly improved using an additional global topic information obtained using Latent Dirichlet Allocation (LDA). In fact, although recurrent architectures theoretically allow the context to indefinitely cycle in the network, Hai Son et al. (2012) have shown that, in practice, this information changes quickly in the classical RNN (Mikolov et al., 2010) structure, and that it is experimentally equivalent to an 8-gram FFNN. Another alternative to model linguistic dependencies, Long-Short Term Memory (LSTM) (Sundermeyer et al., 2012), addresses some learning issues from the original RNN by controlling the longevity of context information in the net-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1473–1481,
Austin, Texas, November 1-5, 2016.

work. This architecture, however, does not particularly model long/short context but rather uses a single state to model the global linguistic context.

Motivated by the works in (Bellegarda, 1998a; Mikolov and Zweig, 2012), this paper proposes a novel neural architecture which explicitly models 1) the local (short) context information, generally syntactic, as well as 2) the global (long) context, which is semantic in nature, using two separate recurrent hidden states. These states evolve in parallel within a long-short range context network. In doing so, the proposed architecture is particularly adapted to model natural languages that manifest local-global context information in their linguistic properties.

We proceed as follows. Section 2 presents a brief overview of short vs long range context language modeling techniques. Section 3 introduces the novel architecture, Long-Short Range Context (LSRC), which explicitly models these two dependencies. Then, Section 4 evaluates the proposed network in comparison to different state-of-the-art language models on the PTB and the LTCB corpus. Finally, we conclude in Section 5.

2 Short vs Long Context Language Models

The goal of a language model is to estimate the probability distribution $p(w_1^T)$ of word sequences $w_1^T = w_1, \cdots, w_T$. Using the chain rule, this distribution can be expressed as

$$p(w_1^T) = \prod_{t=1}^{T} p(w_t|w_1^{t-1}) \quad (1)$$

This probability is generally approximated under different simplifying assumptions, which are typically derived based on different linguistic observations. All these assumptions, however, aim at modeling the optimal context information, be it syntactic and/or semantic, to perform the word prediction. The resulting models can be broadly classified into two main categories: long and short range context models. The rest of this section presents a brief overview of these categories with a particular focus on Neural Network (NN)-based models.

2.1 Short Range Context

This category includes models that approximate (1) based on the Markov dependence assumption of order $N-1$. That is, the prediction of the current word depends only on the last $N-1$ words in the history. In this case, (1) becomes

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t|w_{t-N+1}^{t-1}) \quad (2)$$

The most popular methods that subscribe in this category are the N-gram models (Rosenfeld, 2000; Kneser and Ney, 1995) as well as the FFNN model (Bengio et al., 2003), which estimates each of the terms involved in this product, i.e, $p(w_t|w_{t-N+1}^{t-1})$ in a single bottom-up evaluation of the network.

Although these methods perform well and are easy to learn, the natural languages that they try to encode, however, are not generated under a Markov model due to their dynamic nature and the long range dependencies they manifest. Alleviating this assumption led to an extensive research to develop more suitable modeling techniques.

2.2 Long Range Context

Conventionally, N-gram related LMs have not been built to capture long linguistic dependencies, although significant word triggering information is still available for large contexts. To illustrate such triggering correlations spread over a large context, we use correlation defined over a distance d, given by $c_d(w_1, w_2) = \frac{P_d(w_1,w_2)}{P(w_1)P(w_2)}$. A value greater than 1 shows that it is more likely that the word w_1 follows w_2 at a distance d than expected without the occurrence of w_2. In Figure 1, we show the variation of this correlation for pronouns with the distance d. It can be observed that seeing another "he" about twenty words after having seen a first "he" is much more likely. A similar observation can be made for the word "she". It is, however, surprising that seeing "he" after "he" is three times more likely than seeing "she" after "she", so "he" is much more predictive. In the cases of cross-word triggering of "he" $\rightarrow$ "she" and "she" $\rightarrow$ "he", we find that the correlation is suppressed in comparison to the same word triggering for distances larger than three. In summary, Figure 1 demonstrates that word triggering information exists at large distances, even up to one thousand words. These conclusions were confirmed by similar correlation experiments that we conducted

for different types of words and triggering relations.

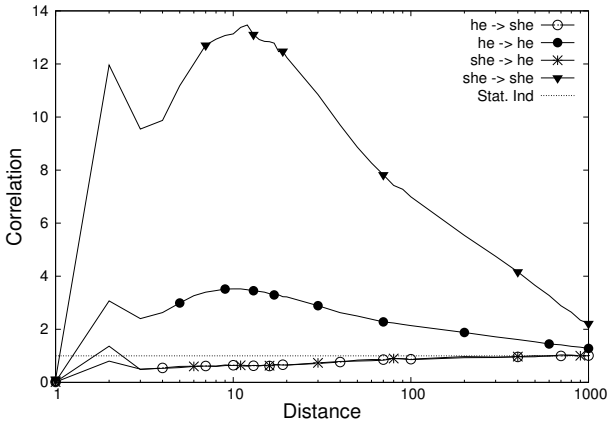


Figure 1: Variation of word triggering correlations for pronouns over large distances.

In order to model this long-term correlation and overcome the restrictive Markov assumption, recurrent language models have been proposed to approximate (1) according to

$$p(w_1^T) \approx \prod_{t=1}^{T} p(w_t|w_{t-1}, h_{t-1}) = \prod_{t=1}^{T} p(w_t|h_t) \quad (3)$$

In NN-based recurrent models, h_t is a context vector which represents the complete history, and modeled as a hidden state that evolves within the network.

2.2.1 Elman-Type RNN-based LM

The classical RNN (Mikolov et al., 2010) estimates each of the product terms in (3) according to

$$H_t = f\left(X_{t-1} + V \cdot H_{t-1}\right) \quad (4)$$
$$P_t = g\left(W \cdot H_t\right) \quad (5)$$

where X_{t-1} is a continuous representation (i.e, embedding) of the word w_{t-1}, V encodes the recurrent connection weights and W is the hidden-to-output connection weights. These parameters define the network and are learned during training. Moreover, $f(\cdot)$ is an activation function, whereas $g(\cdot)$ is the softmax function. Figure (2) shows an example of the standard RNN architecture.

Theoretically, the recurrent connections of an RNN allow the context to indefinitely cycle in the network and thus, modeling long context. In practice, however, Hai Son et al. (2012) have shown that this information changes quickly over time, and that it is experimentally equivalent to an 8-gram FFNN. This observation was confirmed by the experiments that we report in this paper.

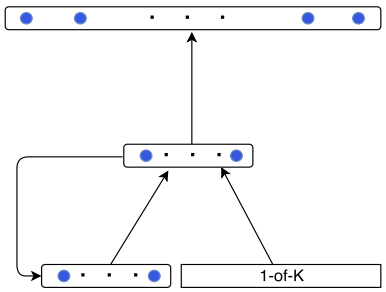


Figure 2: Elman RNN architecture.

2.2.2 Long-Short Term Memory Network

In order to alleviate the rapidly changing context issue in standard RNNs and control the longevity of the dependencies modeling in the network, the LSTM architecture (Sundermeyer et al., 2012) introduces an internal memory state C_t, which explicitly controls the amount of information, to forget or to add to the network, before estimating the current hidden state. Formally, this is done according to

$$\{i, f, o\}_t = \sigma\left(U^{i,f,o} \cdot X_{t-1} + V^{i,f,o} \cdot H_{t-1}\right) \quad (6)$$
$$\tilde{C}_t = f\left(U^c \cdot X_{t-1} + V^c \cdot H_{t-1}\right) \quad (7)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (8)$$
$$H_t = o_t \odot f\left(C_t\right) \quad (9)$$
$$P_t = g\left(W \cdot H_t\right) \quad (10)$$

where $\odot$ is the element-wise multiplication operator, $\tilde{C}_t$ is the memory candidate, whereas i_t, f_t and o_t are the input, forget and output gates of the network, respectively. Figure 3 illustrates the recurrent module of an LSTM network. Learning of an LSTM model requires the training of the network parameters $U^{i,f,o,c}$, $V^{i,f,o,c}$ and W.

Although LSTM models have been shown to outperform classical RNN in modeling long range dependencies, they do not explicitly model long/short context but rather use a single state to encode the global linguistic context.

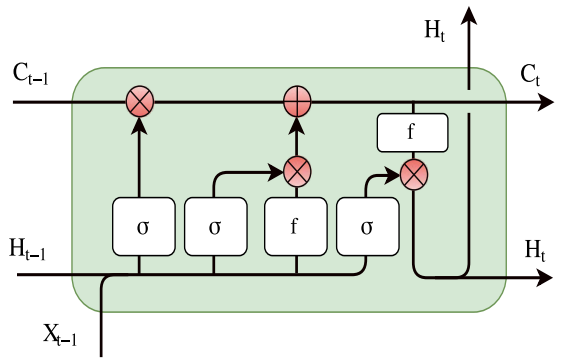


Figure 3: Block diagram of the recurrent module of an LSTM network.

3 Multi-Span Language Models

The attempts to learn and combine short and long range dependencies in language modeling led to what is known as multi-span LMs (Bellegarda, 1998a). The goal of these models is to learn the various constraints, both local and global, that are present in a language. This is typically done using two different models, which separately learn the local and global context, and then combine their resulting linguistic information to perform the word prediction. For instance, Bellegarda (1998b) proposed to use Latent Semantics Analysis (LSA) to capture the global context, and then combine it with the standard N-gram models, which capture the local context, whereas Mikolov and Zweig (2012) proposed to model the global topic information using Latent Dirichlet Allocation (LDA), which is then combined with an RNN-based LM. This idea is not particular to language modeling but has been also used in other Natural Language Processing (NLP) tasks, e.g., Anastasakos et al. (2014) proposed to use a local/global model to perform a spoken language understanding task.

3.1 Long-Short Range Context Network

Following the line of thoughts in (Bellegarda, 1998b; Mikolov and Zweig, 2012), we propose a new multi-span model, which takes advantage of the LSTM ability to model long range context while, simultaneously, learning and integrating the short context through an additional recurrent, local state. In doing so, the resulting Long-Short Range Context (LSRC) network is able to separately model the short/long context while it dynamically combines them to perform the next word prediction task. Formally, this new model is defined as

$$H_t^l = f\left(X_{t-1} + U_l^c \cdot H_{t-1}^l\right) \quad (11)$$

$$\{i,f,o\}_t = \sigma\left(V_l^{i,f,o} \cdot H_t^l + V_g^{i,f,o} \cdot H_{t-1}^g\right) \quad (12)$$

$$\tilde{C}_t = f\left(V_l^c \cdot H_t^l + V_g^c \cdot H_{t-1}^g\right) \quad (13)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (14)$$

$$H_t^g = o_t \odot f\left(C_t\right) \quad (15)$$

$$P_t = g\left(W \cdot H_t^g\right) \quad (16)$$

Learning of an LSRC model requires the training of the local parameters $V_l^{i,f,o,c}$ and U_l^c, the global parameters $V_g^{i,f,o,c}$ and the hidden-to-output connection weights W. This can be done using the standard Back-Propagation Through Time (BPTT) algorithm, which is typically used to train recurrent networks.

The proposed approach uses two hidden states, namely, H_t^l and H_t^g to model short and long range context, respectively. More particularly, the local state H_t^l evolves according to (11) which is nothing but a simple recurrent model as it is defined in (4). In doing so, H_t^l is expected to have a similar behavior to RNN, which has been shown to capture local/short context (up to 10 words), whereas the global state H_t^g follows the LSTM model, which is known to capture longer dependencies (see example in Figure 5). The main difference here, however, is the dependence of the network modules (gates and memory candidate) on the previous local state H_t^l instead of the last seen word X_{t-1}. This model is based on the assumption that the local context carries more linguistic information, and is therefore, more suitable to combine with the global context and update LSTM, compared to the last seen word. Figure 4 illustrates the recurrent module of an LSRC network. It is worth mentioning that this model was not particularly developed to separately learn syntactic and semantic information. This may come, however, as a result of the inherent local and global nature of these two types of linguistic properties.

3.2 Context Range Estimation

For many NLP applications, capturing the global context information can be a crucial component to develop successful systems. This is mainly due to

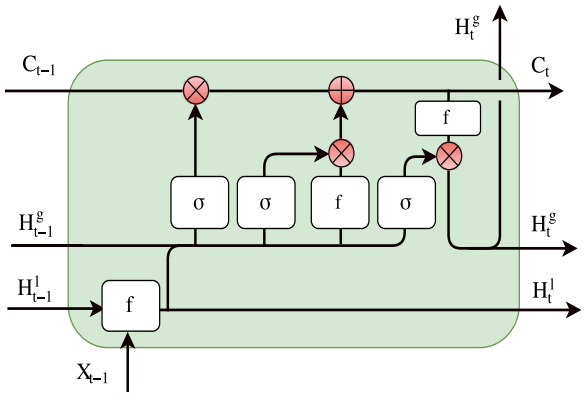


Figure 4: Block diagram of the recurrent module of an LSRC network.

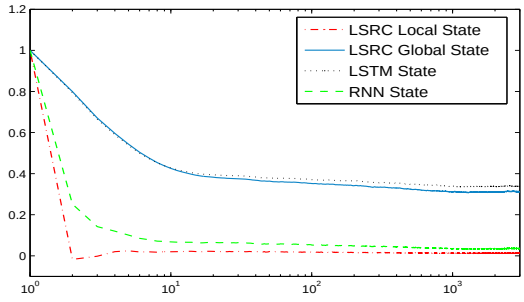


Figure 5: Temporal correlation of the proposed network in comparison to LSTM and RNN.

the inherent nature of languages, where a single idea or topic can span over few sentences, paragraphs or a complete document. LSA-like approaches take advantage of this property, and aim at extracting some hidden "concepts" that best explain the data in a low-dimension "semantic space". To some extent, the hidden layer of LSRC/LSTM can be seen as a vector in a similar space. The information stored in this vector, however, changes continuously based on the processed words. Moreover, interpreting its content is generally difficult. As an alternative, measuring the temporal correlation of this hidden vector can be used as an indicator of the ability of the network to model short and long context dependencies. Formally, the temporal correlation of a hidden state H over a distance d is given by

$$c_d = \frac{1}{D}\sum_{t=1}^{t=D} SM(H_t, H_{t+d}) \qquad (17)$$

where D is the test data size in words and SM is a similarity measure such as the *cosine similarity*. This measure allows us to evaluate how fast does the information stored in the hidden state change over time.

In Figure 5, we show the variation of this temporal correlation for the local and global states of the proposed LSRC network in comparison to RNN and LSTM for various values of the distance d (up to 3000). This figure was obtained on the test set of the Penn Treebank (PTB) corpus, described in Section (4). The main conclusion we can draw from this figure is the ability of the LSRC local and global states (trained jointly) to behave in a similar fashion to RNN and LSTM states (trained separately), respectively. We can also conclude that the LSRC global state and LSTM are able to capture long range correlations, whereas the context changes rapidly over time in RNN and LSRC local state.

4 Experiments and Results

4.1 Experimental Setup

We evaluated the proposed architecture on two different benchmark tasks. The first set of experiments was conducted on the commonly used Penn Treebank (PTB) corpus using the same experimental setup adopted in (Mikolov et al., 2011) and (Zhang et al., 2015). Namely, sections 0-20 are used for training while sections 21-22 and 23-24 are used for validation an testing, respectively. The vocabulary was limited to the most 10k frequent words while the remaining words were mapped to the token <unk>.

In order to evaluate how the proposed approach performs on large corpora in comparison to other methods, we run a second set of experiments on the Large Text Compression Benchmark (LTCB) (Mahoney, 2011). This corpus is based on the enwik9 dataset which contains the first 10^9 bytes of enwiki-20060303-pages-articles.xml. We adopted the same training-test-validation data split as well as the the same data processing[1] which were used in (Zhang et al., 2015). The vocabulary is limited to the most 80k

[1]All the data processing steps described here for PTB and LTCB were performed using the FOFE toolkit in (Zhang et al., 2015), which is available at `https://wiki.eecs.yorku.ca/lab/MLL/_media/projects:fofe:fofe-code.zip`

frequent words with all remaining words replaced by <unk>. Details about the sizes of these two corpora can be found in Table 1.

Corpus	Train	Dev	Test
PTB	930K	74K	82K
LTCB	133M	7.8M	7.9M

Table 1: Corpus size in number of words.

Similarly to the RNN LM toolkit[2] (Mikolov et al., 2011), we have used a single end sentence tag between each two consecutive sentences, whereas the begin sentence tag was not included[3].

4.2 Baseline Models

The proposed LSRC architecture is compared to different LM approaches that model short or long range context. These include the commonly used N-gram Kneser-Ney (KN) (Kneser and Ney, 1995) model with and without cache (Kuhn and De Mori, 1990), as well as different feedforward and recurrent neural architectures. For short (fixed) size context models, we compare our method to 1) the FFNN-based LM (Bengio et al., 2003), as well as 2) the Fixed-size Ordinally Forgetting Encoding (FOFE) approach, which is implemented in (Zhang et al., 2015) as a sentence-based model. For these short size context models, we report the results of different history window sizes (1, 2 and 4). The 1^{st}, 2^{nd} and 4^{th}-order FOFE results were either reported in (Zhang et al., 2015) or obtained using the freely available FOFE toolkit [1].

For recurrent models that were designed to capture long term context, we compared the proposed approach to 3) the full RNN (without classes) (Mikolov et al., 2011), 4) to a deep RNN (D-RNN)[4] (Pascanu et al., 2013), which investigates different approaches to construct mutli-layer RNNs, and finally 5) to the LSTM model (Sundermeyer et al., 2012), which explicitly regulates the amount of information that propagates in the network. The recurrent models results are reported for different numbers of hidden layers (1 or 2). In order to investigate the impact of deep models on the LSRC architecture, we added a single hidden, non-recurrent layer (of size 400 for PTB and 600 for the LTCB experiments) to the LSRC model (D-LSRC). This was sufficient to improve the performance with a negligible increase in the number of model parameters.

4.3 PTB Experiments

For the PTB experiments, the FFNN and FOFE models use a word embedding size of 200, whereas the hidden layer(s) size is fixed at 400, with all hidden units using the Rectified Linear Unit (ReLu) i.e., $f(x) = max(0, x)$ as activation function. We also use the same learning setup adopted in (Zhang et al., 2015). Namely, we use the stochastic gradient descent algorithm with a mini-batch size of 200, the learning rate is initialized to 0.4, the momentum is set to 0.9, the weight decay is fixed at 4×10^{-5}, whereas the training is done in epochs. The weights initialization follows the normalized initialization proposed in (Glorot and Bengio, 2010). Similarly to (Mikolov et al., 2010), the learning rate is halved when no significant improvement of the validation data log-likelihood is observed. Then, we continue with seven more epochs while halving the learning rate after each epoch.

Regarding the recurrent models, we use $f = tanh(\cdot)$ as activation function for all recurrent layers, whereas "$f = sigmoid(\cdot)$" is used for the input, forget and output gates of LSTM and LSRC. The additional non-recurrent layer in D-LSRC, however, uses the ReLu activation function. The word embedding size was set to 200 for LSTM and LSRC whereas it is the same as the hidden layer size for RNN (result of the RNN equation 4). In order to illustrate the effectiveness of the LSRC model, we also report the results when the embedding size is fixed at 100, LSRC(100). The training uses the BPTT algorithm for 5 time steps. Similarly to short context models, the mini-batch was set to 200. The learning rate, however, was set to 1.0 and the weight decay to 5×10^{-5}. The use of momentum did not lead to any additional improvement. Moreover, the data is processed sequentially without any sentence independence assumption. Thus, the recurrent mod-

[2]The RNN LM toolkit is available at `http://www.rnnlm.org/`

[3]This explains the difference in the corpus size compared to the one reported in (Zhang et al., 2015).

[4]The deep RNN results were obtained using L_p and maxout units, dropout regularization and gradient control techniques, which are known to significantly improve the performance. None of these techniques, however, were used in our experiments.

els will be able to capture long range dependencies that exist beyond the sentence boundary.

In order to compare the model sizes, we also report the Number of Parameters (NoP) to train for each of the models above.

	model			model+KN5			NoP
N-1=	1	2	4	1	2	4	4
KN	186	148	141	—	—	—	—
KN+cache	168	134	129	—	—	—	—
	1 Hidden Layer						
FFNN	176	131	119	132	116	107	6.32M
FOFE	123	111	112	108	100	101	6.32M
	Recurrent Models (1 Layer)						
RNN	117			104			8.16M
LSTM (1L)	113			99			6.96M
LSRC(100)	109			96			5.81M
LSRC(200)	104			94			7.0M
	2 Hidden Layers						
FFNN	176	129	114	132	114	102	6.96M
FOFE	116	108	109	104	98	97	6.96M
	Deep Recurrent Models						
D-LSTM (2L)	110			97			8.42M
D-RNN[4] (3L)	107.5			NR			6.16M
D-LSRC(100)	103			93			5.97M
D-LSRC(200)	102			92			7.16M

Table 2: LMs performance on the PTB test set.

Table 2 shows the perplexity evaluation on the PTB test set. As a first observation, we can clearly see that the proposed approach outperforms all other models for all configurations, in particular, RNN and LSTM. This observation includes other models that were reported in the literature, such as random forest LM (Xu and Jelinek, 2007), structured LM (Filimonov and Harper, 2009) and syntactic neural network LM (Emami and Jelinek, 2004). More particularly, we can conclude that LSRC, with an embedding size of 100, achieves a better performance than all other models while reducing the number of parameters by $\approx 29\%$ and $\approx 17\%$ compared to RNN and LSTM, respectively. Increasing the embedding size to 200, which is used by the other models, improves significantly the performance with a resulting NoP comparable to LSTM. The significance of the improvements obtained here over LSTM were confirmed through a statistical significance t-test, which led to p-values $\leq 10^{-10}$ for a significance level of 5% and 0.01%, respectively.

The results of the deep models in Table 2 also show that adding a single non-recurrent hidden layer to LSRC can significantly improve the performance. In fact, the additional layer bridges the gap between the LSRC models with an embedding size of 100 and 200, respectively. The resulting architectures outperform the other deep recurrent models with a significant reduction of the number of parameters (for the embedding size 100), and without usage of dropout regularization, L_p and maxout units or gradient control techniques compared to the deep RNN[4](D-RNN).

We can conclude from these experiments that the explicit modeling of short and long range dependencies using two separate hidden states improves the performance while significantly reducing the number of parameters.

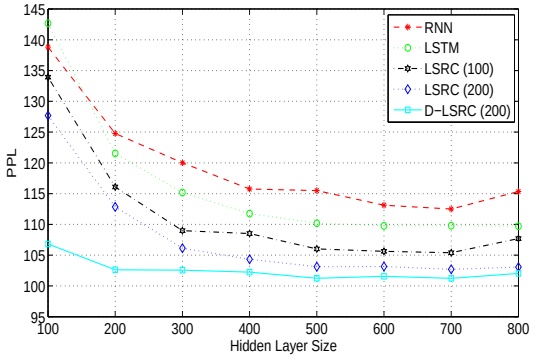


Figure 6: Perplexity of the different NN-based LMs with different hidden layer sizes on the PTB test set.

In order to show the consistency of the LSRC improvement over the other recurrent models, we report the variation of the models performance with respect to the hidden layer size in Figure 6. This figure shows that increasing the LSTM or RNN hidden layer size could not achieve a similar performance to the one obtained using LSRC with a small layer size (e.g., 300). It is also worth mentioning that this observation holds when comparing a 2-recurrent layers LSTM to LSRC with an additional non-recurrent layer.

4.4 LTCB Experiments

The LTCB experiments use the same PTB setup with minor modifications. The results shown in Table 3 follow the same experimental setup proposed in (Zhang et al., 2015). More precisely, these results were obtained without use of momentum or weight decay (due to the long training time required for this corpus), the mini-batch size was set to 400, the learning rate was set to 0.4 and the BPTT step was fixed at 5. The FFNN and FOFE architectures use 2 hidden layers of size 600, whereas RNN, LSTM and LSRC have a single hidden layer of size 600. Moreover, the word embedding size was set to 200 for all models except RNN, which was set to 600. We also report results for an LSTM with 2 recurrent layers as well as for LSRC with an additional non-recurrent layer. The recurrent layers are marked with an "R" in Table 3.

	model			NoP
Context Size M=N-1	1	2	4	4
KN	239	156	132	—
KN+cache	188	127	109	—
FFNN [M*200]-600-600-80k	235	150	114	64.84M
FOFE [M*200]-600-600-80k	112	107	100	64.84M
RNN [600]-R600-80k	85			96.36M
LSTM [200]-R600-80k	66			65.92M
LSTM [200]-R600-R600-80k	61			68.80M
LSRC [200]-R600-80k	63			65.96M
LSRC [200]-R600-600-80k	59			66.32M

Table 3: LMs performance on the LTCB test set.

The results shown in Table 3 generally confirm the conclusions we drew from the PTB experiments above. In particular, we can see that the proposed LSRC model largely outperforms all other models. In particular, LSRC clearly outperforms LSTM with a negligible increase in the number of parameters (resulting from the additional $200 \times 200 = 0.04M$ local connection weights U_l^c) for the single layer results. We can also see that this improvement is maintained for deep models (2 hidden layers), where the LSRC model achieves a slightly better performance while reducing the number of parameters by $\approx 2.5M$ and speeding up the training time by $\approx 20\%$ compared to deep LSTM.

The PTB and LTCB results clearly highlight the importance of recurrent models to capture long range dependencies for LM tasks. The training of these models, however, requires large amounts of data to significantly outperform short context models. This can be seen in the performance of RNN and LSTM in the PTB and LTCB tables above. We can also conclude from these results that the explicit modeling of long and short context in a multi-span model can lead to a significant improvement over state-of-the are models.

5 Conclusion and Future Work

We investigated in this paper the importance, followed by the ability, of standard neural networks to encode long and short range dependencies for language modeling tasks. We also showed that these models were not particularly designed to, explicitly and separately, capture these two linguistic information. As an alternative solution, we proposed a novel long-short range context network, which takes advantage of the LSTM ability to capture long range dependencies, and combines it with a classical RNN network, which typically encodes a much shorter range of context. In doing so, this network is able to encode the short and long range linguistic dependencies using two separate network states that evolve in time. Experiments conducted on the PTB and the large LTCB corpus have shown that the proposed approach significantly outperforms different state-of-the are neural network architectures, including LSTM and RNN, even when smaller architectures are used. This work, however, did not investigate the nature of the long and short context encoded by this network or its possible applications for other NLP tasks. This is part of our future work.

Acknowledgments

This work was in part supported by the Cluster of Excellence for Multimodal Computing and Interaction, the German Research Foundation (DFG) as part of SFB 1102, the EU FP7 Metalogue project (grant agreement number: 611073) and the EU Malorca project (grant agreement number: 698824).

References

[Anastasakos et al.2014] Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continu-

ous word representations for mono and multi-lingual spoken language understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 3246–3250.

[Bellegarda1998a] J. R. Bellegarda. 1998a. A multi-span language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6(5):456–467, Sep.

[Bellegarda1998b] Jerome R. Bellegarda. 1998b. Exploiting both local and global constraints for multi-span statistical language modeling. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98, Seattle, Washington, USA, May 12-15, 1998*, pages 677–680.

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar.

[Brown et al.1990] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, Jun.

[Emami and Jelinek2004] Ahmad Emami and Frederick Jelinek. 2004. Exact training of a neural syntactic language model. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 245–248, Montreal, Quebec, Canada, May.

[Filimonov and Harper2009] Denis Filimonov and Mary P. Harper. 2009. A joint language model with fine-grain syntactic tags. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP), A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1114–1123, Singapore, Aug.

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, May.

[Hai Son et al.2012] Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Katz1987] S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, Mar.

[Kneser and Ney1995] Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, pages 181–184, Detroit, Michigan, USA, May.

[Kuhn and De Mori1990] Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(6):570–583.

[Mahoney2011] Matt Mahoney. 2011. Large text compression benchmark.

[Mikolov and Zweig2012] Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1045–1048, Makuhari, Chiba, Japan, Sep.

[Mikolov et al.2011] T. Mikolov, S. Kombrink, L. Burget, J. ernock, and S. Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531, May.

[Pascanu et al.2013] Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026.

[Rosenfeld2000] Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? In *Proceedings of the IEEE*, volume 88, pages 1270–1278.

[Sundermeyer et al.2012] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, OR, USA, sep.

[Xu and Jelinek2007] Peng Xu and Frederick Jelinek. 2007. Random forests and the data sparseness problem in language modeling. *Computer Speech & Language*, 21(1):105–152.

[Zhang et al.2015] Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Li-Rong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing ACL*, volume 2, pages 495–500, july.

Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration

Changsong Liu[1*], **Shaohua Yang**[1*], **Sari Saba-Sadiya**[1],
Nishant Shukla[2], **Yunzhong He**[2], **Song-Chun Zhu**[2], and **Joyce Y. Chai**[1]

[1]*Department of Computer Science and Engineering*
Michigan State University, East Lansing, MI 48824
[2]*Center for Vision, Cognition, Learning, and Autonomy*
University of California, Los Angeles, CA 90095
{cliu,yangshao,sadiyasa,jchai}@cse.msu.edu
{shukla,yunzhong}@cs.ucla.edu,sczhu@stat.ucla.edu

Abstract

To enable language-based communication and collaboration with cognitive robots, this paper presents an approach where an agent can learn task models jointly from language instruction and visual demonstration using an And-Or Graph (AoG) representation. The learned AoG captures a hierarchical task structure where linguistic labels (for language communication) are grounded to corresponding state changes from the physical environment (for perception and action). Our empirical results on a cloth-folding domain have shown that, although state detection through visual processing is full of uncertainties and error prone, by a tight integration with language the agent is able to learn an effective AoG for task representation. The learned AoG can be further applied to infer and interpret on-going actions from new visual demonstration using linguistic labels at different levels of granularity.

1 Introduction

Given tremendous advances in robotics, computer vision, and natural language processing, a new generation of cognitive robots have emerged that aim to collaborate with humans in joint tasks. To facilitate natural and efficient communication with these physical agents, natural language processing will need to go beyond traditional symbolic representations, but rather ground language to sensors (e.g., visual perception) and actuators (e.g., lower-level control systems) of physical agents. The internal task representation will need to capture both higher-level concepts (for language communication) and lower-level visual features (for perception and action).

To address this need, we have developed an approach on learning *procedural tasks* jointly from language instruction and visual demonstration. In particular, we use And-Or Graph (AoG), which has been used in many computer vision tasks and robotic applications (Zhao and Zhu, 2013; Li et al., 2016; Xiong et al., 2016), to represent a hierarchical task model that not only captures symbolic concepts (extracted from language instructions) but also the corresponding visual state changes from the physical environment (detected by computer vision algorithms).

Different from previous works that ground language to perception (Liu et al., 2012; Matuszek et al., 2012; Kollar et al., 2013; Yu and Siskind, 2013; Yang et al., 2016), a key innovation in our framework is that language is no longer grounded just to perceived objects in the environment, but is further grounded to a hierarchical structure of *state changes* where the states are perceived from the environment during visual demonstration. The state of environment is an important notion in robotic systems as the change of states drives planning for lower-level robotic actions. Thus, connecting language concepts to state changes, our learned AoG provides a unified representation that integrates language and vision to not only support language-based communication but also facilitate robot action planning and execution in the future.

More specifically, within this AoG framework, we have developed and evaluated our algorithms in the

* The first two authors contributed equally to this paper.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1482–1492,
Austin, Texas, November 1-5, 2016.

context of learning a *cloth-folding* task. Although cloth-folding appears simple and intuitive for humans, it represents significant challenges for both vision and robotics systems. Furthermore, although symbolic language processing in this domain is easy due to limited use of vocabulary, grounded language understanding is particularly challenging. A simple phrase (e.g., "fold in half") could have different grounded meanings (e.g., lower-level representation) given different contexts. Thus, this cloth-folding domain is a good starting point to focus on grounding language to task structures.

Our empirical results have shown that, although state detection from the physical world can be extremely noisy, our learning algorithm that tightly incorporates language is capable of acquiring an effective and meaningful task model to compensate the uncertainties in visual processing. Once the AoG for the task is learned, it can be applied by our inference algorithm, for example, to infer on-going actions from new visual demonstration and generate linguistic labels at different levels of granularity to facilitate human-agent communication.

2 Related Work

Recent years have seen an increasing amount of work on grounding language to visual perception (Liu et al., 2012; Matuszek et al., 2012; Yu and Siskind, 2013; Kollar et al., 2013; Naim et al., 2015; Yang et al., 2016; Gao et al., 2016). Furthermore, the robotics community made significant efforts to utilize novel grounding techniques to facilitate task execution given natural language instructions (Chen et al., 2010; Kollar et al., 2010; Tellex et al., 2011; Misra et al., 2014) and task learning from demonstration (Saunders et al., 2006; Chernova and Veloso, 2008).

Research on Learning from Demonstration (LfD) employed various approaches to model the tasks (Argall et al., 2009), such as state-to-action mapping (Chernova and Veloso, 2009), predicate calculus (Hofmann et al., 2016), and Hierarchical Task Networks (Nejati et al., 2006; Hogg et al., 2009). However, aspiring to enable human robot communication, the framework developed in this paper focuses on task representation using language grounded to a structure of state changes detected from the physical world. As demonstrated in recent work (She et al., 2014a; Misra et al., 2015; She and Chai, 2016), explicitly modeling change of states is an important step towards interacting with robots in the physical world.

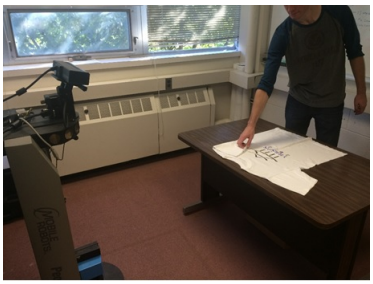


Figure 1: The setting of our situated task learning where a human teacher teaches the robot how to fold a T-shirt through both task demonstrations and language instructions.

Additionally, there has also been an increasing amount of work that learns new tasks either using methods like supervised learning on large corpus of data (Branavan et al., 2010; Branavan et al., 2012; Tellex et al., 2014; Misra et al., 2014), or by learning from humans through dialogue (Cantrell et al., 2012; Mohan et al., 2013; Kirk and Laird, 2013; She et al., 2014b; Mohseni-Kabir et al., 2015). In this paper, we focus on jointly learning new tasks through visual demonstration and language instruction. The learned task model is explicitly represented by an AoG, a hierarchical structure consisting of both linguistic labels and corresponding changes of states from the physical world. This rich task model will facilitate not only language-based communication, but also lower-level action planning and execution.

3 Task and Data

In this paper, we use cloth-folding (e.g., teaching a robot how to fold a T-shirt) as the task to demonstrate and evaluate our joint task learning approach. As mentioned earlier, cloth-folding, although simple for humans, represents a challenging task for the robotics community due to the complex state and action space.

Figure 1 illustrates the setting for our situated task learning. A human teacher can teach a robot how to fold a T-shirt through simultaneous verbal instructions and visual demonstrations. A Microsoft Kinect2 camera is mounted on the robot to record

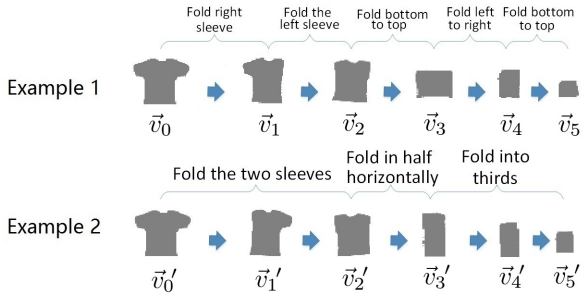


Figure 2: Examples of our parallel data where language instructions are paired with a sequence of visual states detected from the video.

the human's visual demonstration, and the human's corresponding verbal instructions are recorded by Kinect2's embedded microphone array.

A recorded video of task demonstration and its corresponding verbal instruction become one training example for our task learning system. Figure 2 shows two examples of such "parallel data". The visual demonstration is processed into a sequence of visual states, where each state is a numeric vector ($\vec{v}_i$) capturing the visual features of the T-shirt at a particular time (see later Section 5.1 for details). The recorded verbal instructions are then aligned with the sequence of visual states based on the timing information.

During teaching the task, we specifically requested the demonstrator to describe and do each step at roughly the same time. This greatly simplified the alignment problem. Since our ultimate goal is to enable humans to teach the robot through natural language dialogue and demonstration, our hypothesis is that the alignment issue can be alleviated by certain dialogue mechanism (e.g., ask to repeat the action, ask for step-by-step aligned instructions, etc.). As it is human's best interest that the robot gets the clearest instructions, we also anticipate during dialogue human teachers will be collaborative and provide mostly aligned instructions. Certainly, these hypotheses will need to be validated in the dialogue setting in our future work.

In our collected data, each *change of state*, i.e., a transition between two visual states, is caused by one or more physical actions. Some language descriptions align with only a single-step change of state. For instance, "*fold right sleeve*" is aligned with the change ($\vec{v}_0 \rightarrow \vec{v}_1$) and "*fold left sleeve*" is aligned with ($\vec{v}_1 \rightarrow \vec{v}_2$) in Example 1. This kind of single-step change of state is considered as a **primitive action**. Other language descriptions are aligned with a sequence of multiple state changes. For instance, "*fold the two sleeves*" in Example 2 is aligned with two consecutive changes: ($\vec{v}_0', \vec{v}_1', \vec{v}_2'$). This kind of sequence of state changes is considered as a **complex action**, which can be decomposed into partially ordered primitive actions. A complex action can also be concisely represented by the change from the initial state to the end state in the sequence, such as ($\vec{v}_0' \rightarrow \vec{v}_2'$) in Example 2.

These parallel data are used to train and test our learning and inference algorithms presented later.

4 And-Or Graph Representation

We use AoG as the formal model of a procedural task. Figure 3 shows an example AoG for the cloth-folding task. It is a hierarchical structure that explicitly captures the compositionality and reconfigurability of a procedural task. The terminal nodes capture state changes associated with primitive actions of this task, and non-terminal nodes capture state changes associated with complex actions which are further composed by lower-level actions.

In addition to state changes, the learned AoG is also labeled with linguistic information (e.g., verb frames) capturing the causes of the corresponding state changes. The state changes are also considered as grounded meanings of these verb frames. For example, Figure 3 shows two "*fold the t-shirt*" labels at the top layer. Note that although symbolically, these two phrases have the same meaning (e.g., same verb frames), their grounded meanings are different as they correspond to different changes of state. Being able to represent differences or ambiguities in grounded meanings is crucial to connect language to perception and action.

Formally, an AoG is defined as a 5-tuple $\mathcal{G} = (S, \Sigma, N, R, \Theta)$, where

- S is a root node (or a start symbol) representing a complete task.
- Σ is a set of terminal nodes, each of which represents a change of state associated with a primitive action.
- $N = N^{AND} \cup N^{OR}$ is a set of non-terminal nodes, which is divided into two disjoint sub-

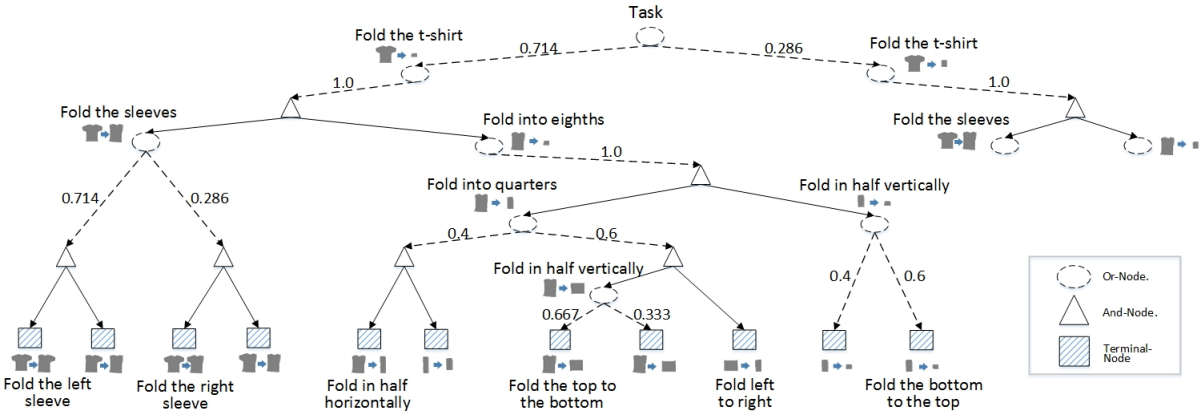


Figure 3: An example of the learned AoG

sets of And-nodes and Or-nodes.

- R is a "child-parent" relation (many-to-one mapping), i.e., $R(n^{ch}) = n^{pa}$ (meaning n^{pa} is the parent node of n^{ch}), where $n^{ch} \in \Sigma \cup N$ and $n^{pa} \in N \cup \{S\}$.
- Θ is a set of conditional probabilities $p(n^{ch}|n^{pa})$, where $n^{pa} \in N^{OR}$, $n^{ch} \in \{n \mid R(n) = n^{pa}\}$. Namely, for each Or-node, Θ defines a probability distribution over the set of all its children nodes.

In essence, our AoG model is equivalent to Probabilistic Context-Free Grammar (PCFG). An AoG can be converted into a PCFG:

- Each And-node and its children form a production rule

$$n^{AND} \rightarrow n_1^{ch} \wedge n_2^{ch} \wedge \dots$$

that represents the decomposition of a complex action into sequentially ordered sub-actions.

- Each Or-node and its children form a production rule

$$n^{OR} \rightarrow n_1^{ch} \mid n_2^{ch} \mid \dots$$

that represents all the alternative ways of accomplishing an action. Each alternative also comes with a probability as specified in Θ.

5 Method

5.1 Vision and Language Processing

The input data to our AoG learning algorithm consist of co-occurring visual demonstrations and language instructions as described in Section 3. Based on the RGB-D information provided by the Kinect2 sensor, we developed a vision processing system to keep track of human's actions and statuses of the T-shirt object.

To learn a meaningful task structure, the most important visual information are those key statuses that the object goes through. Therefore, our vision system processes each visual demonstration into a sequence of *states*. Each state $\vec{v}$ is a multi-dimensional numeric vector that encodes the geometric information of the detected T-shirt, such as its smallest bounding rectangle and largest inscribed contour-fitting rectangle. These key states are detected by tracking the human's folding actions. Namely, whenever a folding action is detected[1], we append the new state caused by the action to the sequence of observed states, till the end of the demonstration.

The verbal instructions given by the demonstrators were mainly verb phrases such as "fold *which-part*", "fold to *which-position*", or "fold *in-what-manner*". A semantic parser[2] is applied to parse each instruction text into a canonical verb-frame representation, such as

$$FOLD: \quad [PART: left\ sleeve] \\ [POSITION: middle].$$

Through the vision and language processing, each task demonstration becomes two parallel sequences, i.e., a sequence of extracted visual states and a sequence of parsed language instructions. The align-

[1]The vision system keeps track of human's hands, and detects a folding action as a gripping action followed by moving and releasing the hand(s).

[2]We use the CMU's Phoenix parser: http://wiki.speech.cs.cmu.edu/olympus/index.php/Phoenix

ment between these two sequences is also extracted from their co-occurrence timing information. Thus, an instance of a task demonstration is formally represented as a 3-tuple $x = (D, L, \Delta)$, where $D = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_M\}$ is the sequence of visual states, $L = \{l_1, l_2, \ldots, l_K\}$ is the sequence of linguistic verb-frames, and $\Delta(k) = (i, j)$ is an "alignment function" specifying the correspondence between a linguistic verb-frame l_k and a single or a sub-sequence of visual state(s) $\{\vec{v}_i, \ldots, \vec{v}_j\}$ $(i \leq j)$.

Then, given a dataset $\mathcal{X}$ of such task demonstrations, our AoG learning algorithm learns an AoG $\mathcal{G}$ as defined in Section 4. The next section describes our learning algorithm in detail.

5.2 AoG Learning Algorithm

Learning an AoG $\mathcal{G} = (S, \Sigma, N, R, \Theta)$ is carried out in two stages. Firstly, we learn a set of terminal nodes Σ to represent the primitive actions (i.e., the actions that can be preformed in a single step). This is done through clustering the observed visual states. Secondly, the hierarchical structure (i.e., N and R) and parameters Θ of the AoG is learned using an iterative grammar induction algorithm.

5.2.1 Learning Terminal Nodes

A terminal node in the AoG represents a primitive action, which causes the object to directly change from one state to another. Thus we represent a terminal node as a 2-tuple of states (or a "change of state"). Since the visual states detected by computer vision are numeric vectors with continuous values, we first apply a clustering algorithm to form a finite set of discrete state representations. Each cluster then represents a unique situation that one can encounter in a task. Since when learning a new task we usually do not know how many unique situations exist, here we employ a greedy clustering algorithm (Ng and Cardie, 2002), which does not assume a fixed number of clusters.

As the greedy clustering algorithm relies on the pairwise similarities of all the visual states, we also train an SVM classifier on a separate dataset of 22 T-shirt folding videos and use its classification output to measure the similarity between two visual states. The SVM classifier takes two numeric vectors as an input, and predicts whether these two vectors represent the same status of a T-shirt. We then apply this SVM classifier on each pair of detected visual states in our new dataset (i.e., the dataset for learning the AoG), and use the SVM's output class label (1 or -1) multiplies its classification confidence score as the similarity measurement between two visual states.

After clustering all the observed visual states in the data, we then replace each numeric vector state representation with the cluster "ID" it belongs to. Thus each visual demonstration now becomes a sequence of symbolic values, denoted as $D' = \{s_1, s_2, \ldots, s_M\}$. And we further transform it into an equivalent *change of state* sequence $C = \{(s_1, s_2), (s_2, s_3), \ldots, (s_{M-1}, s_M)\}$, in which each change of state essentially represents a primitive action in this task. These change of state pairs then form the set of terminal nodes Σ.

5.2.2 Learning the Structure and Parameters

With the sequences of numeric vector states replaced by the "symbolic" change of state sequences in the first stage, we can further learn the structure and parameters of an AoG. Namely, to learn N, R, and Θ that maximize the posterior probability:

$$\begin{aligned} & \arg\max_{N,R,\Theta} P(N, R, \Theta | \mathcal{X}, \Sigma) \\ = & \arg\max_{N,R,\Theta} P(N, R | \mathcal{X}, \Sigma)\, P(\Theta | \mathcal{X}, \Sigma, N, R). \end{aligned}$$

Following the iterative grammar induction paradigm (Tu et al., 2013; Xiong et al., 2016), we employ an iterative procedure that alternatively solves $\arg\max_{N,R} P(N, R | \mathcal{X}, \Sigma)$ and $\arg\max_{\Theta} P(\Theta | \mathcal{X}, \Sigma, N, R)$.

To solve the first term, we use greedy or beam search with a heuristic function similar to (Solan et al., 2005). To solve the second term, we estimate the probability of each branch of an Or-node by computing the frequency of that branch, which is essentially a maximum likelihood estimation similar to (Pei et al., 2013).

In detail, the learning procedure first initializes empty N, R, and Θ, then iterates through the following two steps until no further update can be made.

Step (1): search for new And-nodes.

This step searches for new And-node candidates from $\Sigma \cup N$, and update N and R with the top-ranked candidates. Specifically, we denote an And-node candidate to be searched as $A = (s_l \rightarrow s_m \rightarrow s_r)$.

Here s_l is the initial state of an existing node, whose end state is s_m. And s_r is the end state of another existing node, whose initial state is s_m. Thus an And-node candidate always has two child nodes, and represents a pattern of sub-sequences which starts from state s_l, ends at s_r, and has s_m occurred somewhere in the middle.

Using the above notation, the heuristic function for ranking And-node candidates is defined as

$$h(A) = (1 - \lambda)P_{state}(A) + \lambda P_{label}(A)$$

where $P_{state}(A)$ captures the prevalence of a particular And-node candidate based on the observed state change sequences:

$$P_{state}(A) = \frac{P_R(A) + P_L(A)}{2}$$

and $P_R(A)$ is the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears and the number of times $(s_l \rightarrow s_m)$ appears, and $P_L(A)$ is the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears and the number of times $(s_m \rightarrow s_r)$ appears.

The component $P_{label}(A)$ captures the prevalence of linguistic labels associated with the sequential state change patterns. It is computed as the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ co-occurs with a linguistic instruction[3] and the total number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears.

We specially define two AoG learning settings based on the role that language plays:

- *Tight* language integration: incorporate heuristics on linguistic labels (i.e., $\lambda = 0.5$). In this setting, the learned AoG prefers And-nodes that not only happen frequently, but also can be described by a linguistic label.
- *Loose* language integration: without incorporating the heuristics on linguistic labels ($\lambda = 0$). Each And-node is learned only based on the frequency of its state change pattern. The learned node can still acquire a linguistic label if there happen to be a co-occurring one, but the chance is lower than the "tight" setting.

Step (2): search for new Or-nodes or new branches of existing Or-nodes, then update Θ.

Once new And-nodes are added by the previous step, the next step is to search for Or-nodes that can be created or updated. An Or-node in the AoG essentially represents the set of all And-nodes that share the same initial and end states, denoted as $(s_l \rightarrow s_r)$ here (s_l and s_r are the common initial and end states, respectively). Suppose $(s_l \rightarrow s'_m \rightarrow s_r)$ is a newly added And-node, it is then assigned as a child of the Or-node $(s_l \rightarrow s_r)$. To further update Θ, the branching probability is computes as the ratio between the number of times $(s_l \rightarrow s'_m \rightarrow s_r)$ appears and the number of times $(s_l \rightarrow s_r)$ appears.

5.3 Inference Using AoG

Once a task AoG is learned, it can be applied to interpret and explain new visual demonstrations using linguistic labels. Due to the noises and uncertainties from computer vision processing, one key challenge in interpreting the visual demonstration is to reliably identify the different states of the T-shirt.

To tackle this issue, we formulate a joint inference problem. Namely, given a task demonstration video, we first process it into a sequence of numeric vector states $D = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_M\}$ as described in Section 5.1. Then the goal of inference is to find the most-likely parse tree T and a sequence of "symbolic states" $D' = \{s_1, s_2, \ldots, s_M\}$ based on the AoG $\mathcal{G}$ and the input D:

$$(T^*, D'^*) = \arg\max_{T, D'} P(T, D' \mid \mathcal{G}, D)$$

We apply a chart parsing algorithm (Klein and Manning, 2001) to efficiently solve this problem. Furthermore, to accommodate the ambiguities in mapping a numeric vector state $\vec{v}_m$ to a symbolic state, we take into consideration the top-k hypotheses measured by the similarity between $\vec{v}_m$ and a symbolic state s_k.[4] For each state mapping hypothesis, we add a completed edge between indices m and $m + 1$ in the chart, with s_k as its symbol and a probability p based on the similarity between $\vec{v}_m$ and s_k. Based on the given AoG, the chart parsing algorithm then uses Dynamic Programming to search the best parse tree that maximizes the joint probability of $P(T, D'|\mathcal{G}, D)$.

Figure 4 illustrates the input and output of our inference algorithm. As illustrated by this example,

[3] Such information is encoded in the Δ function as mentioned in Section 5.1.

[4] A symbolic state is represented by a cluster of numeric vector states learned from the training data.

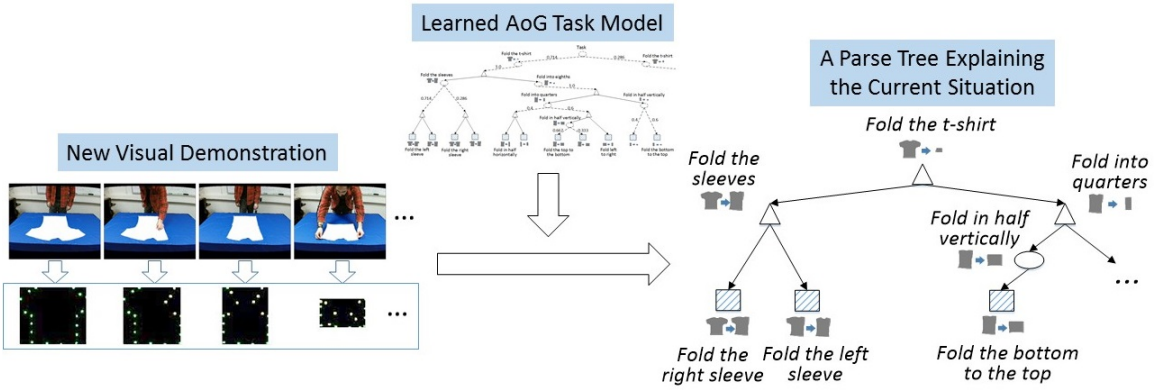


Figure 4: An illustration of the input and output of our AoG-based inference algorithm.

the parse tree represents a hierarchical structure underlying the observed task procedure, and the linguistic labels associated with the nodes can be used to describe the primitive and complex actions involved in the procedure.

6 Evaluation

Using the setting as described in Section 3, we collected 45 T-shirt folding demonstrations from 6 people to evaluate our AoG learning and inference methods. More specifically, we conducted a 5-fold cross validation. In each fold, 36 demonstrations were used for training to learn a task AoG. Then the remaining 9 demonstrations were used for testing, in which the learned AoG is further applied to process each of the testing visual demonstrations.

Motivated by earlier work on plan/activity recognition using CFG-based models (Carberry, 1990; Pynadath and Wellman, 2000), we use an extrinsic task that automatically assigns linguistic labels to new demonstrations to evaluate the quality of the learned AoG and the effectiveness of the inference algorithm. This involves three steps: (1) parse the video using the learned AoG; (2) identify linguistic labels associated with terminal or nonterminal nodes in the parse tree; and (3) compare the identified linguistic labels with the manually annotated labels.

We conduct the evaluation at two levels:

- **Primitive actions:** use linguistic labels associated with terminal nodes to describe the primitive actions in each video. This level provides detailed descriptions on how the observed task procedure is performed step-by-step.
- **Complex actions:** use linguistic labels associated with nonterminal nodes to describe complex actions. This provides a high-level "summary" of the detailed low-level actions.

The capability to recognize fine-grained primitive actions as well as high-level complex actions in a task procedure and to communicate those in language is important for many real-world AI applications such as human-robot collaboration (Mohseni-Kabir et al., 2015) and visual question answering (Tu et al., 2014).

6.1 Primitive Actions

We first compare the performance of interpreting primitive actions using the learned AoG with a baseline. The baseline applies a memory-based (or similarity-base) approach. Given a testing video, it extracts all the different visual states and maps each state to the nearest cluster learned from the training data (see Section 5.2.1). It then pairs each two consecutive states as a change of state instance, and uses the linguistic label corresponding to the identical change of state found in the training data as the label of a primitive action.

We measure the primitive action recognition performance in terms of the *normalized Minimal Edit Distance (MED)*. Namely, for each testing demonstration we calculate the MED between the ground-truth sequence of primitive action labels and the automatically generated sequence of labels, and divide the MED value by the length of the ground-truth sequence to produce a normalized score (a smaller score indicates better performance in recognizing the primitive actions).

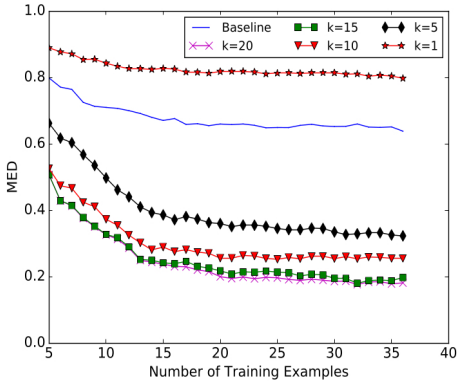


Figure 5: Performance of interpreting primitive actions. Different number of state mapping hypotheses (k) are used in the inference algorithm. The x-axis is the number of training examples used for learning the AoG.

The performances of the baseline and our AoG-based approach are shown in Figure 5. For the AoG-based approach, Figure 5 also shows the performances of incorporating different number of state mapping hypotheses (i.e., $k = 1, 5, 10, 15, 20$) into the inference algorithm (Section 5.3). Here we only report the performance of using AoG learned with the *tight* language integration (see Section 5.2.2), since there is no difference in performance between the tight and loose language integration settings in recognizing primitive actions[5].

As Figure 5 shows, the baseline performance is rather weak (i.e., high MED scores). This is largely due to the noise in state clustering and mapping from vision. After manually inspecting the collected demonstration videos, we found 18 unique statuses associated with folding a T-shirt. However the computer vision based clustering on average produces more than 30 clusters when all the 36 training examples are used. This makes it difficult to directly match the state changes as in the baseline. For our AoG-based method, when the inference algorithm only takes the single best state mapping hypothesis into consideration (i.e., $k = 1$), it yields a very weak performance because the observed state change sequence often cannot be parsed using the learned AoG.

However, the performance of the AoG-based method is significantly improved when multiple state mapping hypotheses are incorporated into the inference process. When the top-5 ($k = 5$) state mapping hypotheses are incorporated into the AoG-based inference, its MED score has already outperformed the baseline by a 0.3 gap ($p < 0.001$ using the Wilcoxon signed-rank test). When $k = 20$, the MED score has dropped by more than 0.6 compared to $k = 1$ ($p < 0.001$).

These results indicate that our AoG-based method is capable of learning useful task structure from small data. When multiple hypotheses of visual state mapping are incorporated, the learned AoG can compensate the uncertainties in vision processing and identify highly reliable primitive actions from unseen demonstrations.

6.2 Complex Actions

We further evaluate the performance of interpreting complex actions using the learned AoG. The baseline for comparison is similar to the one used in the previous section. It first converts a test video into a sequence of "symbolic" states by mapping each detected visual state to its nearest cluster. It then enumerates all the possible segments that consist of more than two consecutive states and search for the identical segments in the training data. If a matching segment is found, then the corresponding linguistic label (if any) is used as the label for a complex action. Since complex actions correspond to nonterminal nodes in the parse tree generated by AoG-based inference, and some of them may have linguistic labels while others may not. We use *precision*, *recall*, and *F-score* to measure how well the generated linguistic labels match the manually segmented and annotated complex actions in testing videos.

Figure 6 shows the F-scores of recognizing complex actions using the AoG learned from the *loose* and the *tight* language integration, respectively. In this figure, results are based on $k = 20$ state mapping hypotheses incorporated into the inference algorithm. As shown here, performances from both settings are significantly better than the baseline ($p < 0.001$). The AoG learned based on the tight integration with language yields significantly better performance than the loose integration (over 0.2 gain on F-score, $p < 0.001$).

This result indicates that the tight integration of

[5]Because the linguistic labels generated for primitive actions are all from terminal nodes, and the two different AoG learning settings only affect nonterminal nodes.

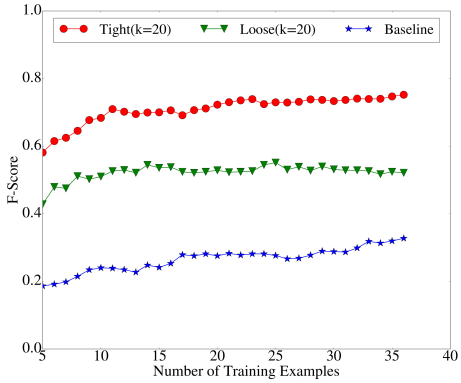


Figure 6: Performances (F-score) of recognizing complex actions. The lowest curve shows the performance from the baseline. Two other curves represent the performance using the AoG learned from the *loose* integration and the *tight* integration with language respectively (where $k = 20$ is used in inference).

language during AoG learning favors And-node patterns that are more likely to be described by natural language (or more consistent with human conceptualization of the task structure)[6]. Such an AoG representation can lead to recognition of video segments that can be better explained or summarized by human language. This capability of learning explicit and language-oriented task representations is important to link language and vision for enabling situated human-agent communication/collaboration.

Table 6.2 further shows the results from different numbers of state mapping hypotheses that are incorporated into the inference algorithm. As shown here, the trend of performance improvement with the increase in k is again observed. When multiple state mapping hypotheses are incorporated in inference, the learned AoG is capable of compensating uncertainties in vision processing and producing better parses for unseen visual demonstrations.

7 Conclusion and Future Work

This paper presents an approach on task learning where an agent can learn a grounded task model from human demonstrations and language instructions. A key innovation of this work is grounding language to a perceived structure of state changes based on AoG representation. Once the task model is acquired, it can be used as a basis to support collaboration and communication between humans and agents/robots. Using cloth-folding as an example, our empirical results have demonstrated that tightly integrating language with vision can effectively produce task structures in AoG that can generalize well to new demonstrations.

Although we have only made an initial attempt on a small task, our approach can be naturally extended to more complex tasks such like assembling and cooking. Both the AoG representation and the task learning approach are general and applicable to different domains. What needs to be adapted is the representation of the visual states and computer vision algorithms to detect these states for a specific task.

Grounding language to a structure of perceived state changes will provide an important stepping stone towards integrating language, perception, and action for human-robot communication and collaboration. Currently, our algorithms learn the task structures based on offline parallel data. Our future work will explore incremental learning through human-agent dialogue to acquire grounded task structures.

Table 1: Performance of recognizing complex actions using the AoG learned from the loose and tight integration of language as described in Section 5.2. Different (k) number of state mapping hypotheses are used in the inference algorithm.

		k=1	k=5	k=10	k=15	k=20
Precision	Loose	0.34	0.76	0.79	0.84	0.84
	Tight	0.34	0.8	0.86	0.89	0.9
Recall	Loose	0.12	0.33	0.35	0.38	0.38
	Tight	0.12	0.51	0.59	0.64	0.65
F-Score	Loose	0.17	0.46	0.49	0.52	0.52
	Tight	0.18	0.63	0.70	0.74	0.75

Acknowledgments

The authors are grateful to Sarah Fillwock and James Finch for their help on data collection and processing, to Mun Wai Lee for his helpful discussions, and to anonymous reviewers for their valuable comments and suggestions. This work was supported in part by N66001-15-C-4035 from the DARPA SIMPLEX program, and IIS-1208390 and IIS-1617682 from the National Science Foundation.

[6]By further investigating the learned AoG under the two different settings, we found that the nonterminal nodes learned from the tight language integration setting is more likely to acquire a linguistic label (33%) than the nonterminal nodes learned from the loose setting (18%).

References

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1268–1277.

S. R. K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 126–135.

Rehj Cantrell, J. Benton, Kartik Talamadupula, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. 2012. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 471–478.

Sandra Carberry. 1990. *Plan recognition in natural language dialogue*. MIT press.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1):397–436.

Sonia Chernova and Manuela Veloso. 2008. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1183–1186.

Sonia Chernova and Manuela Veloso. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y. Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Volume 1: Long Papers, pages 1814–1824.

Till Hofmann, Tim Niemueller, Jens Claßen, and Gerhard Lakemeyer. 2016. Continual planning in golog. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Chad Hogg, Ugur Kuter, and Héctor Muñoz-Avila. 2009. Learning hierarchical task networks for nondeterministic planning domains. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1708–1714.

James R. Kirk and John E. Laird. 2013. Learning task formulations through situated interactive instruction. In *Proceedings of the Second Annual Conference on Advances in Cognitive Systems (ACS)*, volume 219, page 236.

Dan Klein and Christopher D. Manning. 2001. An o(n^3) agenda-based chart parser for arbitrary probabilistic context-free grammars. *Stanford Technical Report*.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction (HRI)*, pages 259–266.

Thomas Kollar, Jayant Krishnamurthy, and Grant P. Strimel. 2013. Toward interactive grounded language acqusition. In *Robotics: Science and Systems*.

Bo Li, Tianfu Wu, Caiming Xiong, and Song-Chun Zhu. 2016. Recognizing car fluents from video. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Changsong Liu, Rui Fang, and Joyce Y. Chai. 2012. Towards mediating shared perceptual basis in situated dialogue. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 140–149.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proceedings of Robotics: Science and Systems (RSS)*.

Dipendra K. Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 992–1002.

Shiwali Mohan, James Kirk, and John Laird. 2013. A computational model for situated task learning with interactive instruction. In *Proceedings of the 12th International Conference on Cognitive Modeling (ICCM)*.

Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. 2015. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 205–212.

Iftekhar Naim, Young Chol Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. 2015. Discriminative unsupervised alignment of natural language instructions with corresponding video

segments. In *North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*.

Negin Nejati, Pat Langley, and Tolga Konik. 2006. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 665–672.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 104–111.

Mingtao Pei, Zhangzhang Si, Benjamin Z Yao, and Song-Chun Zhu. 2013. Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding*, 117(10):1369–1383.

David V. Pynadath and Michael P. Wellman. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 507–514. Morgan Kaufmann Publishers Inc.

Joe Saunders, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction (HRI)*, pages 118–125.

Lanbo She and Joyce Y. Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lanbo She, Yu Cheng, Joyce Y. Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014a. Teaching robots new actions through natural language instructions. In *Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y. Chai, and Ning Xi. 2014b. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. 2014. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167.

Kewei Tu, Maria Pavlovskaia, and Song-Chun Zhu. 2013. Unsupervised structure learning of stochastic and-or grammars. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1322–1330.

Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. 2014. Joint video and text parsing for understanding events and answering queries. *IEEE MultiMedia*, 21(2):42–70.

Caiming Xiong, Nishant Shukla, Wenlong Xiong, and Song-Chun Zhu. 2016. Robot learning with a spatial, temporal, and causal and-or graph. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 149–159.

Haonan Yu and Jeffrey M. Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 53–63.

Yibiao Zhao and Song-Chun Zhu. 2013. Scene parsing by integrating function, geometry and appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3119–3126.

Resolving Language and Vision Ambiguities Together: Joint Segmentation & Prepositional Attachment Resolution in Captioned Scenes

Gordon Christie[1,*], **Ankit Laddha**[2,*], **Aishwarya Agrawal**[1], **Stanislaw Antol**[1]
Yash Goyal[1], **Kevin Kochersberger**[1], **Dhruv Batra**[3,1]
[1]Virginia Tech [2]Carnegie Mellon University [3]Georgia Institute of Technology
ankit1991laddha@gmail.com
{gordonac,aish,santol,ygoyal,kbk,dbatra}@vt.edu

Abstract

We present an approach to simultaneously perform semantic segmentation and prepositional phrase attachment resolution for captioned images. Some ambiguities in language cannot be resolved without simultaneously reasoning about an associated image. If we consider the sentence "I shot an elephant in my pajamas", looking at language alone (and not using common sense), it is unclear if it is the person or the elephant wearing the pajamas or both. Our approach produces a diverse set of plausible hypotheses for both semantic segmentation and prepositional phrase attachment resolution that are then jointly reranked to select the most consistent pair. We show that our semantic segmentation and prepositional phrase attachment resolution modules have complementary strengths, and that joint reasoning produces more accurate results than any module operating in isolation. Multiple hypotheses are also shown to be crucial to improved multiple-module reasoning. Our vision and language approach significantly outperforms the Stanford Parser (De Marneffe et al., 2006) by 17.91% (28.69% relative) and 12.83% (25.28% relative) in two different experiments. We also make small improvements over DeepLab-CRF (Chen et al., 2015).

1 Introduction

Perception and intelligence problems are hard. Whether we are interested in understanding an image or a sentence, our algorithms must operate under tremendous levels of ambiguity. When a human reads the sentence "I eat sushi with tuna", it is clear that the preposition phrase "with tuna" modifies "sushi" and not the act of eating, but this may be ambiguous to a machine. This problem of determining whether a prepositional phrase ("with tuna") modifies a noun phrase ("sushi") or verb phrase ("eating") is formally known as Prepositional Phrase Attachment Resolution (PPAR) (Ratnaparkhi et al., 1994). Consider the captioned scene shown in Fig-

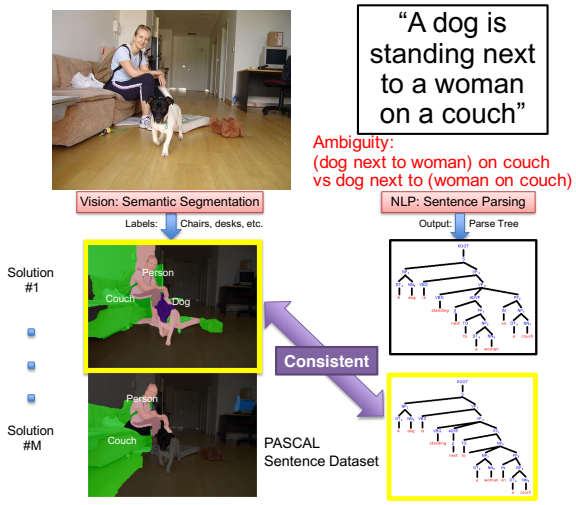


Figure 1: Overview of our approach. We propose a model for simultaneous 2D semantic segmentation and prepositional phrase attachment resolution by reasoning about sentence parses. The language and vision modules each produce M diverse hypotheses, and the goal is to select a pair of consistent hypotheses. In this example the ambiguity to be resolved from the image caption is whether the dog is standing on or next to the couch. Both modules benefit by selecting a pair of compatible hypotheses.

* Denotes equal contribution

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1493–1503,
Austin, Texas, November 1-5, 2016.

ure 1. The caption "A dog is standing next to a woman on a couch" exhibits a PP attachment ambiguity – "(dog next to woman) on couch" vs "dog next to (woman on couch)". It is clear that having access to image segmentations can help resolve this ambiguity, and having access to the correct PP attachment can help image segmentation.

There are two main roadblocks that keep us from writing a single unified model (say a graphical model) to perform both tasks: (1) Inaccurate Models – empirical studies (Meltzer et al., 2005, Szeliski et al., 2008, Kappes et al., 2013) have repeatedly found that models are often inaccurate and miscalibrated – their "most-likely" beliefs are placed on solutions far from the ground-truth. (2) Search Space Explosion – jointly reasoning about multiple modalities is difficult due to the combinatorial explosion of search space ($\{$exponentially-many segmentations$\}$ $\times$ $\{$exponentially-many sentence-parses$\}$).

Proposed Approach and Contributions. In this paper, we address the problem of simultaneous object segmentation (also called semantic segmentation) and PPAR in captioned scenes. To the best of our knowledge this is the first paper to do so.

Our main thesis is that a set of diverse plausible hypotheses can serve as a concise interpretable summary of uncertainty in vision and language 'modules' (What does the semantic segmentation module see in the world? What does the PPAR module describe?) and form the basis for tractable joint reasoning (How do we reconcile what the semantic segmentation module sees in the world with how the PPAR module describes it?).

Given our two modules with M hypotheses each, how can we integrate beliefs across the segmentation and sentence parse modules to pick the best pair of hypotheses? Our key focus is *consistency* – correct hypotheses from different modules will be correct in a consistent way, but incorrect hypotheses will be incorrect in incompatible ways. Specifically, we develop a MEDIATOR model that scores pairs for consistency and searches over all M^2 pairs to pick the highest scoring one. We demonstrate our approach on three datasets – ABSTRACT-50S (Vedantam et al., 2014), PASCAL-50S, and PASCAL-Context-50S (Mottaghi et al., 2014). We show that our vision+language approach significantly outperforms the Stanford Parser (De Marneffe et al., 2006) by 20.66% (36.42% relative) for ABSTRACT-50S, 17.91% (28.69% relative) for PASCAL-50S, and by 12.83% (25.28% relative) for PASCAL-Context-50S. We also make small but consistent improvements over DeepLab-CRF (Chen et al., 2015).

2 Related Work

Most works at the intersection of vision and NLP tend to be 'pipeline' systems, where vision tasks take 1-best inputs from NLP (*e.g.*, sentence parsings) without trying to improve NLP performance and vice-versa. For instance, Fidler et al. (2013) use prepositions to improve object segmentation and scene classification, but only consider the most-likely parse of the sentence and do not resolve ambiguities in text. Analogously, Yatskar et al. (2014) investigate the role of object, attribute, and action classification annotations for generating human-like descriptions. While they achieve impressive results at generating descriptions, they assume perfect vision modules to generate sentences. Our work uses current (still imperfect) vision and NLP modules to reason about images and provided captions, and simultaneously improve both vision and language modules. Similar to our philosophy, an earlier work by Barnard and Johnson (2005) used images to help disambiguate word senses (*e.g.* piggy banks vs snow banks). In a more recent work, Gella et al. (2016) studied the problem of reasoning about an image and a verb, where they attempt to pick the correct sense of the verb that describes the action depicted in the image. Berzak et al. (2015) resolve linguistic ambiguities in sentences coupled with videos that represent different interpretations of the sentences. Perhaps the work closest to us is Kong et al. (2014), who leverage information from an RGBD image and its sentential description to improve 3D semantic parsing and resolve ambiguities related to coreference resolution in the sentences (*e.g.*, what "it" refers to). We focus on a different kind of ambiguity – the Prepositional Phrase (PP) attachment resolution. In the classification of parsing ambiguities, coreference resolution is considered a discourse ambiguity (Poesio and Artstein, 2005) (arising out of two different words across sentences for the same object), while PP attachment is considered a syntactic ambiguity (arising out of multiple valid sentence

structures) and is typically considered much more difficult to resolve (Bach, 2016, Davis, 2016).

A number of recent works have studied problems at the intersection of vision and language, such as Visual Question Answering (Antol et al., 2015, Geman et al., 2014, Malinowski et al., 2015), Visual Madlibs (Yu et al., 2015), and image captioning (Vinyals et al., 2015, Fang et al., 2015). Our work falls in this domain with a key difference that we produce *both* vision and NLP outputs.

Our work also has similarities with works on 'spatial relation learning' (Malinowski and Fritz, 2014, Lan et al., 2012), *i.e*. learning a visual representation for noun-preposition-noun triplets ("car on road"). While our approach can certainly utilize such spatial relation classifiers if available, the focus of our work is different. Our goal is to improve semantic segmentation and PPAR by jointly reranking segmentation-parsing solution pairs. Our approach implicitly learns spatial relationships for prepositions ("on", "above") but these are simply emergent latent representations that help our reranker pick out the most consistent pair of solutions.

Our work utilizes a line of work (Batra et al., 2012, Batra, 2012, Prasad et al., 2014) on producing diverse plausible solutions from probabilistic models, which has been successfully applied to a number of problem domains (Guzman-Rivera et al., 2013, Yadollahpour et al., 2013, Gimpel et al., 2013, Premachandran et al., 2014, Sun et al., 2015, Ahmed et al., 2015).

3 Approach

In order to emphasize the generality of our approach, and to show that our approach is compatible with a wide class of implementations of semantic segmentation and PPAR modules, we present our approach with the modules abstracted as "black boxes" that satisfy a few general requirements and minimal assumptions. In Section 4, we describe each of the modules in detail, making concrete their respective features, and other details.

3.1 What is a Module?

The goal of a module is to take input variables $\mathbf{x} \in \mathcal{X}$ (images or sentences), and predict output variables $\mathbf{y} \in \mathcal{Y}$ (semantic segmentation) and $\mathbf{z} \in \mathcal{Z}$ (prepositional attachment expressed in sentence parse). The two requirements on a module are that it needs to be able to produce *scores* $S(\mathbf{y}|\mathbf{x})$ for potential solutions and a list of *plausible hypotheses* $\mathbf{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^M\}$.

Multiple Hypotheses. In order to be useful, the set $\mathbf{Y}$ of hypotheses must provide an accurate summary of the score landscape. Thus, the hypotheses should be plausible (*i.e*., high-scoring) and mutually non-redundant (*i.e*., diverse). Our approach (described next) is applicable to any choice of diverse hypothesis generators. In our experiments, we use the k-best algorithm of Huang and Chiang (2005) for the sentence parsing module and the DivMBest algorithm (Batra et al., 2012) for the semantic segmentation module. Once we instantiate the modules in Section 4, we describe the diverse solution generation in more detail.

3.2 Joint Reasoning Across Multiple Modules

We now show how to intergrate information from both segmentation and PPAR modules. Recall that our key focus is *consistency* – correct hypotheses from different modules will be correct in a consistent way, but incorrect hypotheses will be incorrect in incompatible ways. Thus, our goal is to search for a pair (semantic segmentation, sentence parsing) that is mutually consistent.

Let $\mathbf{Y} = \{\mathbf{y}^1, \ldots, \mathbf{y}^M\}$ denote the M semantic segmentation hypotheses and $\mathbf{Z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^M\}$ denote the M PPAR hypotheses.

MEDIATOR Model. We develop a "mediator" model that identifies high-scoring hypotheses across modules in agreement with each other. Concretely, we can express the MEDIATOR model as a factor graph where each node corresponds to a module (semantic segmentation and PPAR). Working with such a factor graph is typically completely intractable because each node $\mathbf{y}, \mathbf{z}$ has exponentially-many states (image segmentations, sentence parsing). As illustrated in Figure 2, in this factor-graph view, the hypothesis sets $\mathbf{Y}, \mathbf{Z}$ can be considered 'delta-approximations' for reducing the size of the output spaces.

Unary factors $S(\cdot)$ capture the score/likelihood of each hypothesis provided by the corresponding module for the image/sentence at hand. Pairwise factors $C(\cdot, \cdot)$ represent consistency factors. Impor-

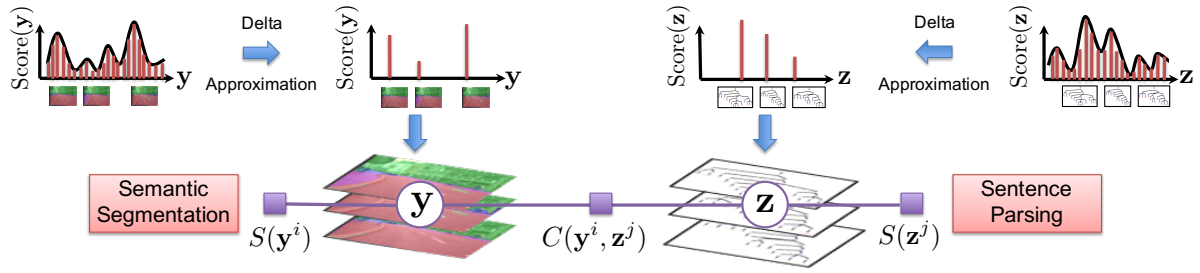


Figure 2: Illustrative inter-module factor graph. Each node takes exponentially-many or infinitely-many states and we use a 'delta approximation' to limit support.

tantly, since we have restricted each module variables to just M states, we are free to capture *arbitrary domain-specific high-order relationships* for consistency, without any optimization concerns. In fact, as we describe in our experiments, these consistency factors may be designed to exploit domain knowledge in fairly sophisticated ways.

Consistency Inference. We perform exhaustive inference over all possible tuples.

$$\underset{i,j\in\{1,\ldots,M\}}{\operatorname{argmax}} \Big\{\mathcal{M}(\mathbf{y}^i,\mathbf{z}^j) = S(\mathbf{y}^i) + S(\mathbf{z}^j) + C(\mathbf{y}^i,\mathbf{z}^j)\Big\}. \tag{1}$$

Notice that the search space with M hypotheses each is M^2. In our experiments, we allow each module to take a different value for M, and typically use around 10 solutions for each module, leading to a mere 100 pairs, which is easily enumerable. We found that even with such a small set, at least one of the solutions in the set tends to be *highly accurate*, meaning that the hypothesis sets have relatively high recall. This shows the power of using a small set of diverse hypotheses. For a large M, we can exploit a number of standard ideas from the graphical models literature (*e.g.* dual decomposition or belief propagation). In fact, this is one reason we show the factor in Figure 2; there is a natural decomposition of the problem into modules.

Training MEDIATOR. We can express the MEDIATOR score as $\mathcal{M}(\mathbf{y}^i,\mathbf{z}^j) = \mathbf{w}^\intercal\boldsymbol{\phi}(\mathbf{x},\mathbf{y}^i,\mathbf{z}^j)$, as a linear function of *score and consistency features* $\boldsymbol{\phi}(\mathbf{x},\mathbf{y}^i,\mathbf{z}^j) = [\phi_S(\mathbf{y}^i);\phi_S(\mathbf{z}^j);\phi_C(\mathbf{y}^i,\mathbf{z}^j)]$, where $\phi_S(\cdot)$ are the single-module (semantic segmentation and PPAR module) score features, and $\phi_C(\cdot,\cdot)$ are the inter-module consistency features. We describe these features in detail in the experiments. We learn these consistency weights $\mathbf{w}$ from a dataset annotated with ground-truth for the two modules $\mathbf{y},\mathbf{z}$. Let $\{\mathbf{y}^*,\mathbf{z}^*\}$ denote the `oracle` pair, composed of the most accurate solutions in the hypothesis sets. We learn the MEDIATOR parameters in a discriminative learning fashion by solving the following Structured SVM problem:

$$\min_{\mathbf{w},\xi_{ij}} \quad \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + C\sum_{ij}\xi_{ij} \tag{2a}$$

$$s.t. \quad \underbrace{\mathbf{w}^\intercal\boldsymbol{\phi}(\mathbf{x},\mathbf{y}^*,\mathbf{z}^*)}_{\text{Score of } \texttt{oracle} \text{ tuple}} - \underbrace{\mathbf{w}^\intercal\boldsymbol{\phi}(\mathbf{x},\mathbf{y}^i,\mathbf{z}^j)}_{\text{Score of any other tuple}}$$

$$\geq \underbrace{1}_{\text{Margin}} - \underbrace{\frac{\xi_{ij}}{\mathcal{L}(\mathbf{y}^i,\mathbf{z}^j)}}_{\text{Slack scaled by loss}} \qquad \forall i,j \in \{1,\ldots,M\}. \tag{2b}$$

Intuitively, we can see that the constraint (2b) tries to maximize the (soft) margin between the score of the `oracle` pair and all other pairs in the hypothesis sets. Importantly, the slack (or violation in the margin) is scaled by the loss of the tuple. Thus, if there are other good pairs not too much worse than the `oracle`, the margin for such tuples will not be tightly enforced. On the other hand, the margin between the `oracle` and bad tuples will be very strictly enforced.

This learning procedure requires us to define the loss function $\mathcal{L}(\mathbf{y}^i,\mathbf{z}^j)$, *i.e.*, the cost of predicting a tuple (semantic segmentation, sentence parsing). We use a weighted average of individual losses:

$$\mathcal{L}(\mathbf{y}^i,\mathbf{z}^j) = \alpha\ell(\mathbf{y}^{gt},\mathbf{y}^i) + (1-\alpha)\ell(\mathbf{z}^{gt},\mathbf{z}^j) \tag{3}$$

The standard measure for evaluating semantic segmentation is average Jaccard Index (or Intersection-over-Union) (Everingham et al., 2010), while for evaluating sentence parses w.r.t. their prepositional phrase attachment, we use the fraction of prepositions correctly attached. In our experiments, we report results with such a convex combination of module loss functions (for different values of α).

4 Experiments

We now describe the setup of our experiments, provide implementation details of the modules, and describe the consistency features.

Datasets. Access to rich annotated image + caption datasets is crucial for performing quantitative evaluations. Since this is the first paper to study the problem of joint segmentation and PPAR, no standard datasets for this task exist so we had to curate our own annotations for PPAR on three image caption datasets – ABSTRACT-50S (Vedantam et al., 2014), PASCAL-50S (Vedantam et al., 2014) (expands the UIUC PASCAL sentence dataset (Rashtchian et al., 2010) from 5 captions per image to 50), and PASCAL-Context-50S (Mottaghi et al., 2014) (which uses the PASCAL Context image annotations and the same sentences as PASCAL-50S). Our annotations are publicly available on the authors' webpages. To curate the PASCAL-Context-50S PPAR annotations, we first select all sentences that have preposition phrase attachment ambiguities. We then plotted the distribution of prepositions in these sentences. The top 7 prepositions are used, as there is a large drop in the frequencies beyond these. The 7 prepositions are: "on", "with", "next to", "in front of", "by", "near", and "down". We then further sampled sentences to ensure uniform distribution across prepositions. We perform a similar filtering for PASCAL-50S and ABSTRACT-50S (using the top-6 prepositions for ABSTRACT-50S). Details are in the supplement. We consider a preposition ambiguous if there are at least two parsings where one of the two objects in the preposition dependency is the same across the two parsings while the other object is different (*e.g.* (dog on couch) and (woman on couch)). To summarize the statistics of all three datasets:

1. **ABSTRACT-50S** (Vedantam et al., 2014): 25,000 sentences (50 per image) with 500 images from abstract scenes made from clipart. Filtering for captions containing the top-6 prepositions resulted in 399 sentences describing 201 unique images. These 6 prepositions are: "with", 'next to", "on top of", "in front of", "behind", and "under". Overall, there are 502 total prepositions, 406 ambiguous prepositions, 80.88% ambiguity rate and 60 sentences with multiple ambiguous prepositions.
2. **PASCAL-50S** (Vedantam et al., 2014): 50,000 sentences (50 per image) for the images in the UIUC PASCAL sentence dataset (Rashtchian et al., 2010). Filtering for the top-7 prepositions resulted in a total of 30 unique images, and 100 image-caption pairs, where ground-truth PPAR were carefully annotated by two vision + NLP graduate students. Overall, there are 213 total prepositions, 147 ambiguous prepositions, 69.01% ambiguity rate and 35 sentences with multiple ambiguous prepositions.
3. **PASCAL-Context-50S** (Mottaghi et al., 2014): We use images and captions from PASCAL-50S, but with PASCAL Context segmentation annotations (60 categories instead of 21). This makes the vision task more challenging. Filtering this dataset for the top-7 prepositions resulted in a total of 966 unique images and 1,822 image-caption pairs. Ground truth annotations for the PPAR were collected using Amazon Mechanical Turk. Workers were shown an image and a prepositional attachment (extracted from the corresponding parsing of the caption) as a phrase ("woman on couch"), and asked if it was correct. A screenshot of our interface is available in the supplement. Overall, there are 2,540 total prepositions, 2,147 ambiguous prepositions, 84.53% ambiguity rate and 283 sentences with multiple ambiguous prepositions.

Setup. *Single Module:* We first show that visual features help PPAR by using the ABSTRACT-50S dataset, which contains clipart scenes where the extent and position of all the objects in the scene is known. This allows us to consider a scenario with a perfect vision system.

Multiple Modules: In this experiment we use imperfect language and vision modules, and show improvements on the PASCAL-50S and PASCAL-Context-50S datasets.

Module 1: Semantic Segmentation (SS) $\mathbf{y}$. We use DeepLab-CRF (Chen et al., 2015) and DivMBest (Batra et al., 2012) to produce M diverse segmentations of the images. To evaluate we use image-level class-averaged Jaccard Index.

Module 2: PP Attachment Resolution (PPAR)

z. We use a recent version (v3.3.1; released 2014) of the PCFG Stanford parser module (De Marneffe et al., 2006, Huang and Chiang, 2005) to produce M parsings of the sentence. In addition to the parse trees, the module can also output *dependencies*, which make syntactical relationships more explicit. Dependencies come in the form *dependency_type(*$word_1$*,* $word_2$*)*, such as the preposition dependency *prep_on(woman-8, couch-11)* (the number indicates the word position in sentence). To evaluate, we count the percentage of preposition attachments that the parse gets correct.

Baselines:

- INDEP. In our experiments, we compare our proposed approach (MEDIATOR) to the highest scoring solution predicted independently from each module. For semantic segmentation this is the output of DeepLab-CRF (Chen et al., 2015) and for the PPAR module this is the 1-best output of the Stanford Parser (De Marneffe et al., 2006, Huang and Chiang, 2005). Since our hypothesis lists are generated by greedy M-Best algorithms, this corresponds to predicting the $(\mathbf{y}^1, \mathbf{z}^1)$ tuple. This comparison establishes the importance of joint reasoning. To the best of our knowledge, there is no existing (or even natural) joint model to compare to.
- DOMAIN ADAPTATION. We learn a reranker on the parses. Note that domain adaptation is only needed for PPAR since the Stanford parser is trained on Penn Treebank (Wall Street Journal text) and not on text about images (such as image captions). Such domain adaptation is not necessary for semantic segmentation. This is a competitive single-module baseline. Specifically, we use the *same* parse-based features as our approach, and learn a reranker over the M_z parse trees ($M_z = 10$).

Our approach (MEDIATOR) significantly outperforms both baselines. The improvements over INDEP show that joint reasoning produces more accurate results than any module (vision or language) operating in isolation. The improvements over DOMAIN ADAPTATION establish the source of improvements is indeed vision, and not the reranking step. Simply adapting the parse from its original training domain (Wall Street Journal) to our domain (image captions) is not enough.

Ablative Study. Ours-CASCADE: This ablation studies the importance of multiple hypothesis. For each module (say $\mathbf{y}$), we feed the single-best output of the other module $\mathbf{z}^1$ as input. Each module learns its own weight $\mathbf{w}$ using *exactly the same* consistency features and learning algorithm as MEDIATOR and predicts one of the plausible hypotheses $\hat{\mathbf{y}}^{\text{CASCADE}} = \text{argmax}_{\mathbf{y}\in\mathbf{Y}}\, \mathbf{w}^\intercal \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z}^1)$. This ablation of our system is similar to (Heitz et al., 2008) and helps us in disentangling the benefits of multiple hypothesis and joint reasoning.

Finally, we note that Ours-CASCADE can be viewed as special cases of MEDIATOR. Let MEDIATOR-$(M_\mathbf{y}, M_\mathbf{z})$ denote our approach run with $M_\mathbf{y}$ hypotheses for the first module and $M_\mathbf{z}$ for the second. Then INDEP corresponds to MEDIATOR-$(1, 1)$ and CASCADE corresponds to predicting the $\mathbf{y}$ solution from MEDIATOR-$(M_\mathbf{y}, 1)$ and the $\mathbf{z}$ solution from MEDIATOR-$(1, M_\mathbf{z})$. To get an upper-bound on our approach, we report `oracle`, the accuracy of the most accurate tuple in 10×10 tuples.

In the main paper, our results are presented where MEDIATOR was trained with equally weighted loss ($\alpha = 0.5$), but we provide additional results for varying values of α in the supplement.

MEDIATOR and Consistency Features. Recall that we have two types of features – (1) score features $\phi_S(y^i)$ and $\phi_S(z^j)$, which try to capture how likely solutions y^i and z^j are respectively, and (2) consistency features $\phi_C(y^i, z^j)$, which capture how consistent the PP attachments in z^j are with the segmentation in y^i. For each ($object_1$, *preposition*, $object_2$) in z^j, we compute 6 features between $object_1$ and $object_2$ segmentations in y^i. Since the humans writing the captions may use multiple synonymous words (*e.g.* dog, puppy) for the same visual entity, we use word2vec (Mikolov et al., 2013) similarities to map the nouns in the sentences to the corresponding dataset categories.

- **Semantic Segmentation Score Features ($\phi_S(y^i)$) (2-dim)**: We use ranks and solution scores from DeepLab-CRF (Chen et al., 2015).
- **PPAR Score Features ($\phi_S(z^i)$) (9-dim)**: We use ranks and the log probability of parses from (De Marneffe et al., 2006), and 7 binary indicators for PASCAL (6 for ABSTRACT-50S) denoting which prepositions are present in the parse.

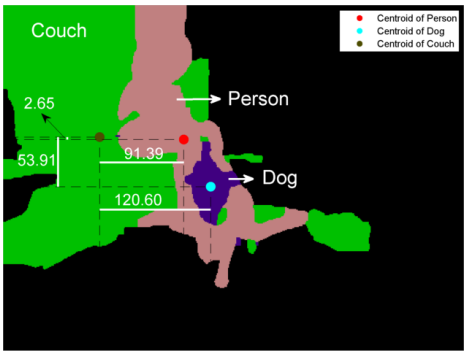


Figure 3: Example on PASCAL-50S ("A dog is standing next to a woman on a couch."). The ambiguity in this sentence "(dog next to woman) on couch" vs "dog next to (woman on couch)". We calculate the horizontal and vertical distances between the segmentation centers of "person" and "couch" and between the segmentation centers of "dog" and "couch". We see that the "dog" is much further below the couch (53.91) than the woman (2.65). So, if the MEDIATOR model learned that "on" means the first object is above the second object, we would expect it to choose the "person on couch" preposition parsing.

- **Inter-Module Consistency Features (56-dim)**: For each of the 7 prepositions, 8 features are calculated:
  - One feature is the Euclidean distance between the center of the segmentation masks of the two objects connected by the preposition. These two objects in the segmentation correspond to the categories with which the soft similarity of the two objects in the sentence is highest among all PASCAL categories.
  - Four features capture $\max\{0,$ (normalized -directional-distance)$\}$, where directional-distance measures above/below/left/right displacements between the two objects in the segmentation, and normalization involves dividing by height/width.
  - One feature is the ratio of sizes between object$_1$ and object$_2$ in the segmentation.
  - Two features capture the word2vec similarity between the two objects in PPAR (say 'puppy' and 'kitty') with their most similar PASCAL category (say 'dog' and 'cat'), where these features are 0 if the categories are not present in segmentation.

A visual illustration for some of these features for PASCAL can be seen in Figure 3. In the case where an object parsed from z^j is not present in the segmentation y^i, the distance features are set to 0. The ratio of areas features (area of smaller object / area of larger object) are also set to 0 assuming that the smaller object is missing. In the case where an object has two or more connected components in the segmentation, the distances are computed w.r.t. the centroid of the segmentation and the area is computed as the number of pixels in the union of the instance segmentation masks. We also calculate 20 features for PASCAL-50S and 59 features for PASCAL-Context-50S that capture that consistency between y^i and z^j, in terms of presence/absence of PASCAL categories. For each noun in PPAR we compute its word2vec similarity with all PASCAL categories. For each of the PASCAL categories, the feature is the sum of similarities (with the PASCAL category) over all nouns if the category is present in segmentation, and is -1 times the sum of similarities over all nouns otherwise. This feature set was not used for ABSTRACT-50S, since these features were intended to help improve the accuracy of the semantic segmentation module. For ABSTRACT-50S, we only use the 5 distance features, resulting in a 30-dim feature vector.

4.1 Single-Module Results

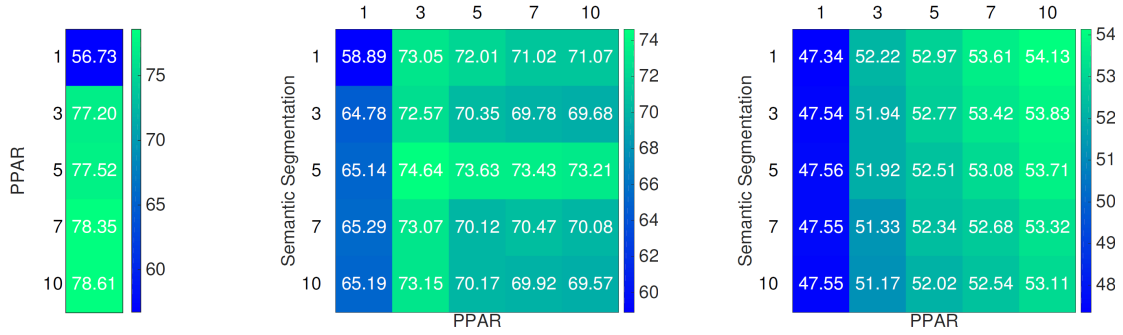
We performed a 10-fold cross-validation on the ABSTRACT-50S dataset to pick M (=10) and the weight on the hinge-loss for MEDIATOR (C). The results are presented in Table 1. Our approach significantly outpeforms 1-best outputs of the Stanford Parser (De Marneffe et al., 2006) by 20.66% (36.42% relative). This shows a need for diverse hypotheses and reasoning about visual features when picking a sentence parse. `oracle` denotes the best achievable performance using these 10 hypotheses.

Module	Stanford Parser	Domain Adaptation	Ours	`oracle`
PPAR	56.73	57.23	**77.39**	97.53

Table 1: Results on our subset of ABSTRACT-50S.

4.2 Multiple-Module Results

We performed 10-fold cross-val for our results of PASCAL-50S and PASCAL-Context-50S, with 8



(a) ABSTRACT-50S (b) PASCAL-50S (c) PASCAL-Context-50S

Figure 4: (a) Validation accuracies for different values of M on ABSTRACT-50S, (b) for different values of $M_{\mathbf{y}}, M_{\mathbf{z}}$ on PASCAL-50S, (c) for different values of $M_{\mathbf{y}}, M_{\mathbf{z}}$ on PASCAL-Context-50S.

	PASCAL-50S			PASCAL-Context-50S		
	Instance-Level Jaccard Index	PPAR Acc.	Average	Instance-Level Jaccard Index	PPAR Acc.	Average
DeepLab-CRF	66.83	-	-	**43.94**	-	-
Stanford Parser	-	62.42	-	-	50.75	-
Average	-	-	64.63	-	-	47.345
Domain Adaptation	-	72.08	-	-	58.32	-
Ours CASCADE	67.56	75.00	71.28	**43.94**	**63.58**	**53.76**
Ours MEDIATOR	**67.58**	**80.33**	**73.96**	**43.94**	**63.58**	**53.76**
`oracle`	69.96	96.50	83.23	49.21	75.75	62.48

Table 2: Results on our subset of the PASCAL-50S and PASCAL-Context-50S datasets. We are able to significantly outperform the Stanford Parser and make small improvements over DeepLab-CRF for PASCAL-50S.

`train` folds, 1 `val` fold, and 1 `test` fold, where the `val` fold was used to pick $M_{\mathbf{y}}$, $M_{\mathbf{z}}$, and C. Figure 4 shows the average combined accuracy on `val`, which was found to be maximal at $M_{\mathbf{y}} = 5, M_{\mathbf{z}} = 3$ for PASCAL-50S, and $M_{\mathbf{y}} = 1, M_{\mathbf{z}} = 10$ for PASCAL-Context-50S, which are used at test time.

We present our results in Table 2. Our approach significantly outperforms the Stanford Parser (De Marneffe et al., 2006) by 17.91% (28.69% relative) for PASCAL-50S, and 12.83% (25.28% relative) for PASCAL-Context-50S. We also make small improvements over DeepLab-CRF (Chen et al., 2015) in the case of PASCAL-50S. To measure statistical significance of our results, we performed paired t-tests between MEDIATOR and INDEP. For both modules (and average), the null hypothesis (that the accuracies of our approach and baseline come from the same distribution) can be successfully rejected at p-value 0.05. For sake of completeness, we also compared MEDIATOR with our ablated system (CASCADE) and found statistically significant differences only in PPAR.

These results demonstrate a need for each module to produce a diverse set of plausible hypotheses for our MEDIATOR model to reason about. In the case of PASCAL-Context-50S, MEDIATOR performs identical to CASCADE since $M_{\mathbf{y}}$ is chosen as 1 (which is the CASCADE setting) in cross-validation. Recall that MEDIATOR is a larger model class than CASCADE (in fact, CASCADE is a special case of MEDIATOR with $M_y = 1$). It is interesting to see that the large model class does not hurt, and MEDIATOR gracefully reduces to a smaller capacity model (CASCADE) if the amount of data is not enough to warrant the extra capacity. We hypothesize that in the presence of more training data, cross-validation may pick a different setting of $M_{\mathbf{y}}$ and $M_{\mathbf{z}}$, resulting in full utilization of the model capacity. Also note that our domain adaptation baseline achieved an accuracy higher than MAP/Stanford-Parser, but significantly lower than our approach for both PASCAL-50S and PASCAL-Context-50S. We also performed this for our single-module experiment and picked M_z (=10) with cross-validation,

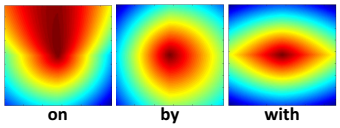


Figure 5: Visualizations for 3 different prepositions (red = high scores, blue = low scores). We can see that our model has implicitly learned spatial arrangements unlike other spatial relation learning (SRL) works.

Feature set	PASCAL-50S		PASCAL-Context-50S
	Instance-Level Jaccard Index	PPAR Acc.	PPAR Acc.
All features	67.58	80.33	63.58
Drop all consistency	66.96	66.67	61.47
Drop Euclidean distance	67.27	77.33	63.77
Drop directional distance	67.12	78.67	63.63
Drop word2vec	67.58	78.33	62.72
Drop category presence	67.48	79.25	61.19

Table 3: Ablation study of different feature combinations. Only PPAR Acc. is shown for PASCAL-Context-50S because $M_y = 1$.

which resulted in an accuracy of 57.23%. Again, this is higher than MAP/Stanford-Parser (56.73%), but significantly lower than our approach (77.39%). Clearly, domain adaptation alone is not sufficient. We also see that `oracle` performance is fairly high, suggesting that when there is ambiguity and room for improvement, MEDIATOR is able to rerank effectively.

Ablation Study for Features. Table 3 displays results of an ablation study on PASCAL-50S and PASCAL-Context-50S to show the importance of the different features. In each row, we retain the module score features and drop a single set of consistency features. We can see all consistency features contribute to the performance of MEDIATOR.

Visualizing Prepositions. Figure 5 shows a visualization for what our MEDIATOR model has implicitly learned about 3 prepositions ("on", "by", "with"). These visualizations show the score obtained by taking the dot product of distance features (Euclidean and directional) between $object_1$ and $object_2$ connected by the preposition with the corresponding learned weights of the model, considering $object_2$ to be at the center of the visualization. Notice that these were learned without explicit training for spatial learning as in spatial relation learning (SRL) works (Malinowski and Fritz, 2014, Lan et al., 2012). These were simply recovered as an intermediate step towards reranking SS + PPAR hypotheses. Also note that SRL cannot handle multiple segmentation hypotheses, which our work shows are important (Table 2 CASCADE). In addition, our approach is more general.

5 Discussions and Conclusion

We presented an approach to the simultaneous reasoning about prepositional phrase attachment resolution of captions and semantic segmentation in images that integrates beliefs across the modules to pick the best pair of a diverse set of hypotheses. Our full model (MEDIATOR) significantly improves the accuracy of PPAR over the Stanford Parser by 17.91% for PASCAL-50S and by 12.83% for PASCAL-Context-50S, and achieves a small improvement on semantic segmentation over DeepLab-CRF for PASCAL-50S. These results demonstrate a need for information exchange between the modules, as well as a need for a diverse set of hypotheses to concisely capture the uncertainties of each module. Large gains in PPAR validate our intuition that vision is very helpful for dealing with ambiguity in language. Furthermore, we see even larger gains are possible from the oracle accuracies.

While we have demonstrated our approach on a task involving simultaneous reasoning about language and vision, our approach is general and can be used for other applications. Overall, we hope our approach will be useful in a number of settings.

Acknowledgements

We thank Larry Zitnick, Mohit Bansal, Kevin Gimpel, and Devi Parikh for helpful discussions, suggestions, and feedback included in this work. A majority of this work was done while AL was an intern at Virginia Tech. This work was partially supported by a National Science Foundation CAREER award, an Army Research Office YIP Award, an Office of Naval Research grant N00014-14-1-0679, and GPU donations by NVIDIA, all awarded to DB. GC was supported by DTRA grant HDTRA1-13-1-0015 provided by KK. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the U.S. Government or any sponsor.

References

Faruk Ahmed, Dany Tarlow, and Dhruv Batra. 2015. Optimizing Expected Intersection-over-Union with Candidate-Constrained CRFs. In *ICCV*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Kent Bach. 2016. Routledge encyclopedia of philosophy entry. `http://online.sfsu.edu/kbach/ambguity.html`.

Kobus Barnard and Matthew Johnson. 2005. Word Sense Disambiguation with Pictures. *Artificial Intelligence*, 167.

Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. 2012. Diverse M-Best Solutions in Markov Random Fields. In *ECCV*.

Dhruv Batra. 2012. An Efficient Message-Passing Algorithm for the M-Best MAP Problem. In *UAI*.

Yevgeni Berzak, Andrei Barbu, Daniel Harari, Boris Katz, and Shimon Ullman. 2015. Do You See What I Mean? Visual Resolution of Linguistic Ambiguities. In *EMNLP*.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2015. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *ICLR*.

Ernest Davis. 2016. Notes on ambiguity. `http://cs.nyu.edu/faculty/davise/ai/ambiguity.html`.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC*.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2).

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From Captions to Visual Concepts and Back. In *CVPR*.

Sanja Fidler, Abhishek Sharma, and Raquel Urtasun. 2013. A Sentence is Worth a Thousand Pixels. In *CVPR*.

Spandana Gella, Mirella Lapata, and Frank Keller. 2016. Unsupervised Visual Sense Disambiguation for Verbs using Multimodal Embeddings. In *NAACL HLT*.

Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. 2014. A Visual Turing Test for Computer Vision Systems. In *PNAS*.

K. Gimpel, D. Batra, C. Dyer, and G. Shakhnarovich. 2013. A Systematic Exploration of Diversity in Machine Translation. In *EMNLP*.

Abner Guzman-Rivera, Pushmeet Kohli, and Dhruv Batra. 2013. DivMCuts: Faster Training of Structural SVMs with Diverse M-Best Cutting-Planes. In *AISTATS*.

Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. 2008. Cascaded Classification Models: Combining Models for Holistic Scene Understanding. In *NIPS*.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *IWPT*, pages 53–64.

Jörg H. Kappes, Bjoern Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Jan Lellmann, Nikos Komodakis, and Carsten Rother. 2013. A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problems. In *CVPR*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? Text-to-Image Coreference. In *CVPR*.

Tian Lan, Weilong Yang, Yang Wang, and Greg Mori. 2012. Image Retrieval with Structured Object Queries Using Latent Ranking SVM. In *ECCV*.

Mateusz Malinowski and Mario Fritz. 2014. A pooling approach to modelling spatial relations for image retrieval and annotation. *arXiv preprint arXiv:1411.5190*.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask Your Neurons: A Neural-based Approach to Answering Questions about Images. In *ICCV*.

Talya Meltzer, Chen Yanover, and Yair Weiss. 2005. Globally Optimal Solutions for Energy Minimization in Stereo Vision Using Reweighted Belief Propagation. In *ICCV*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. 2014. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*.

Massimo Poesio and Ron Artstein. 2005. Annotating (Anaphoric) ambiguity. In *Corpus Linguistics Conference*.

Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. 2014. Submodular meets Structured: Finding Diverse Subsets in Exponentially-Large Structured Item Sets. In *NIPS*.

Vittal Premachandran, Daniel Tarlow, and Dhruv Batra. 2014. Empirical Minimum Bayes Risk Prediction:

How to extract an extra few% performance from vision models with just three more parameters. In *CVPR*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting Image Annotations Using Amazon's Mechanical Turk. In *NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the workshop on Human Language Technology*. ACL.

Qing Sun, Ankit Laddha, and Dhruv Batra. 2015. Active Learning for Structured Probabilistic Models With Histogram Approximation. In *CVPR*.

Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. 2008. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors. *PAMI*, 30(6).

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2014. CIDEr: Consensus-based Image Description Evaluation. In *CVPR*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and Tell: A Neural Image Caption Generator. In *CVPR*.

Payman Yadollahpour, Dhruv Batra, and Greg Shakhnarovich. 2013. Discriminative Re-ranking of Diverse Segmentations. In *CVPR*.

Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See No Evil, Say No Evil: Description Generation from Densely Labeled Images. In *Lexical and Computational Semantics*.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual Madlibs: Fill in the Blank Description Generation and Question Answering. In *ICCV*.

CHARAGRAM: Embedding Words and Sentences via Character n-grams

John Wieting Mohit Bansal Kevin Gimpel Karen Livescu
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{jwieting,mbansal,kgimpel,klivescu}@ttic.edu

Abstract

We present CHARAGRAM embeddings, a simple approach for learning character-based compositional models to embed textual sequences. A word or sentence is represented using a character n-gram count vector, followed by a single nonlinear transformation to yield a low-dimensional embedding. We use three tasks for evaluation: word similarity, sentence similarity, and part-of-speech tagging. We demonstrate that CHARAGRAM embeddings outperform more complex architectures based on character-level recurrent and convolutional neural networks, achieving new state-of-the-art performance on several similarity tasks.[1]

1 Introduction

Representing textual sequences such as words and sentences is a fundamental component of natural language understanding systems. Many functional architectures have been proposed to model compositionality in word sequences, ranging from simple averaging (Mitchell and Lapata, 2010; Iyyer et al., 2015) to functions with rich recursive structure (Socher et al., 2011; Zhu et al., 2015; Tai et al., 2015; Bowman et al., 2016). Most work uses words as the smallest units in the compositional architecture, often using pretrained word embeddings or learning them specifically for the task of interest (Tai et al., 2015; He et al., 2015).

Some prior work has found benefit from using character-based compositional models that encode arbitrary character sequences into vectors. Examples include recurrent neural networks (RNNs) and convolutional neural networks (CNNs) on character sequences, showing improvements for several NLP tasks (Ling et al., 2015a; Kim et al., 2015; Ballesteros et al., 2015; dos Santos and Guimarães, 2015). By sharing subword information across words, character models have the potential to better represent rare words and morphological variants.

Our approach, CHARAGRAM, uses a much simpler functional architecture. We represent a character sequence by a vector containing counts of character n-grams, inspired by Huang et al. (2013). This vector is embedded into a low-dimensional space using a single nonlinear transformation. This can be interpreted as learning embeddings of character n-grams, which are learned so as to produce effective sequence embeddings when a summation is performed over the character n-grams in the sequence.

We consider three evaluations: word similarity, sentence similarity, and part-of-speech tagging. On multiple word similarity datasets, CHARAGRAM outperforms RNNs and CNNs, achieving state-of-the-art performance on SimLex-999 (Hill et al., 2015). When evaluated on a large suite of sentence-level semantic textual similarity tasks, CHARAGRAM embeddings again outperform the RNN and CNN architectures as well as the PARAGRAM-PHRASE embeddings of Wieting et al. (2016). We also consider English part-of-speech (POS) tagging using the bidirectional long short-term memory tagger of Ling et al. (2015a). The three architectures reach similar performance, though CHARAGRAM converges fastest to high accuracy.

[1] Trained models and code are available at http://ttic.uchicago.edu/~wieting.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1504–1515,
Austin, Texas, November 1-5, 2016.

We perform extensive analysis of our CHARAGRAM embeddings. We find large gains in performance on rare words, showing the empirical benefit of subword modeling. We also compare performance across different character n-gram vocabulary sizes, finding that the semantic tasks benefit far more from large vocabularies than the syntactic task. However, even for challenging semantic similarity tasks, we still see strong performance with only a few thousand character n-grams.

Nearest neighbors show that CHARAGRAM embeddings simultaneously address differences due to spelling variation, morphology, and word choice. Inspection of embeddings of particular character n-grams reveals etymological links; e.g., *die* is close to *mort*. We release our resources to the community in the hope that CHARAGRAM can provide a strong baseline for subword-aware text representation.

2 Related Work

We first review work on using subword information in word embedding models. The simplest approaches append subword features to word embeddings, letting the model learn how to use the subword information for particular tasks. Some added knowledge-based morphological features to word representations (Alexandrescu and Kirchhoff, 2006; El-Desoky Mousa et al., 2013). Others learned embeddings jointly for subword units and words, defining simple compositional architectures (often based on addition) to create word embeddings from subword embeddings (Lazaridou et al., 2013; Botha and Blunsom, 2014; Qiu et al., 2014; Chen et al., 2015).

A recent trend is to use richer functional architectures to convert character sequences into word embeddings. Luong et al. (2013) used recursive models to compose morphs into word embeddings, using unsupervised morphological analysis. Ling et al. (2015a) used a bidirectional long short-term memory (LSTM) RNN on characters to embed arbitrary word types, showing strong performance for language modeling and POS tagging. Ballesteros et al. (2015) used this model to represent words for dependency parsing. Several have used character-level RNN architectures for machine translation, whether for representing source or target words (Ling et al., 2015b; Luong and Manning, 2016), or for generating entire translations character-by-character (Chung et al., 2016).

Sutskever et al. (2011) and Graves (2013) used character-level RNNs for language modeling. Others trained character-level RNN language models to provide features for NLP tasks, including tokenization and segmentation (Chrupała, 2013; Evang et al., 2013), and text normalization (Chrupała, 2014).

CNNs with character n-gram filters have been used to embed arbitrary word types for several tasks, including language modeling (Kim et al., 2015), part-of-speech tagging (dos Santos and Zadrozny, 2014), named entity recognition (dos Santos and Guimarães, 2015), text classification (Zhang et al., 2015), and machine translation (Costa-Jussà and Fonollosa, 2016). Combinations of CNNs and RNNs on characters have also been explored (Józefowicz et al., 2016).

Most closely-related to our approach is the DSSM (instantiated variously as "deep semantic similarity model" or "deep structured semantic model") developed by Huang et al. (2013). For an information retrieval task, they represented words using feature vectors containing counts of character n-grams. Sperr et al. (2013) used a very similar technique to represent words in neural language models for machine translation. Our CHARAGRAM embeddings are based on this same idea. We show this strategy to be extremely effective when applied to both words and sentences, outperforming character LSTMs like those used by Ling et al. (2015a) and character CNNs like those from Kim et al. (2015).

3 Models

We now describe models that embed textual sequences using their characters, including our CHARAGRAM model and the baselines that we compare to. We denote a character-based textual sequence by $x = \langle x_1, x_2, ..., x_m \rangle$, which includes space characters between words as well as special start-of-sequence and end-of-sequence characters. We use x_i^j to denote the subsequence of characters from position i to position j inclusive, i.e., $x_i^j = \langle x_i, x_{i+1}, ..., x_j \rangle$, and we define $x_i^i = x_i$.

Our CHARAGRAM model embeds a character sequence x by adding the vectors of its character n-

grams followed by an elementwise nonlinearity:

$$g_{\text{CHAR}}(x) = h\left(\boldsymbol{b} + \sum_{i=1}^{m+1} \sum_{j=1+i-k}^{i} \mathbb{I}\left[x_j^i \in V\right] W^{x_j^i}\right) \quad (1)$$

where h is a nonlinear function, $\boldsymbol{b} \in \mathbb{R}^d$ is a bias vector, k is the maximum length of any character n-gram, $\mathbb{I}[p]$ is an indicator function that returns 1 if p is true and 0 otherwise, V is the set of character n-grams included in the model, and $W^{x_j^i} \in \mathbb{R}^d$ is the vector for character n-gram x_j^i.

The set V is used to restrict the model to a predetermined set (vocabulary) of character n-grams. Below, we compare several choices for V. The number of parameters in the model is $d + d|V|$. This model is based on the letter n-gram hashing technique developed by Huang et al. (2013). One can also view Eq. (1) (as they did) as first populating a vector of length $|V|$ with counts of character n-grams followed by a nonlinear transformation.

We compare the CHARAGRAM model to two other models. First we consider LSTM architectures (Hochreiter and Schmidhuber, 1997) over the character sequence x, using the version from Gers et al. (2003). We use a forward LSTM over the characters in x, then take the final LSTM hidden vector as the representation of x. Below we refer to this model as "charLSTM."

We also compare to convolutional neural network (CNN) architectures, which we refer to below as "charCNN." We use the architecture from Kim (2014) with a single convolutional layer followed by an optional fully-connected layer. We use filters of varying lengths of character n-grams, using two primary configurations of filter sets, one of which is identical to that used by Kim et al. (2015). Each filter operates over the entire sequence of character n-grams in x and we use max pooling for each filter. We tune over the choice of nonlinearity for both the convolutional filters and for the optional fully-connected layer. We give more details below about filter sets, n-gram lengths, and nonlinearities.

We note that using character n-gram convolutional filters is similar to our use of character n-grams in the CHARAGRAM model. The difference is that, in the CHARAGRAM model, the n-gram must match exactly for its vector to affect the representation, while in the CNN each filter will affect the representation of all sequences (depending on the nonlinearity being used). So the CHARAGRAM model is able to learn precise vectors for particular character n-grams with specific meanings, while there is pressure for the CNN filters to capture multiple similar patterns that recur in the data. Our qualitative analysis shows the specificity of the learned character n-gram vectors learned by the CHARAGRAM model.

4 Experiments

We perform three sets of experiments. The goal of the first two (Section 4.1) is to produce embeddings for textual sequences such that the embeddings for paraphrases have high cosine similarity. Our third evaluation (Section 4.2) is a classification task, and follows the setup of the English part-of-speech tagging experiment from Ling et al. (2015a).

4.1 Word and Sentence Similarity

We compare the ability of our models to capture semantic similarity for both words and sentences. We train on noisy paraphrase pairs from the Paraphrase Database (PPDB; Ganitkevitch et al., 2013) with an L_2 regularized contrastive loss objective function, following the training procedure of Wieting et al. (2015) and Wieting et al. (2016). More details are provided in the supplementary material.

4.1.1 Datasets

For word similarity, we focus on two of the most commonly used datasets for evaluating semantic similarity of word embeddings: WordSim-353 (WS353) (Finkelstein et al., 2001) and SimLex-999 (SL999) (Hill et al., 2015). We also evaluate our best model on the Stanford Rare Word Similarity Dataset (Luong et al., 2013).

For sentence similarity, we evaluate on a diverse set of 22 textual similarity datasets, including all datasets from every SemEval semantic textual similarity (STS) task from 2012 to 2015. We also evaluate on the SemEval 2015 Twitter task (Xu et al., 2015) and the SemEval 2014 SICK Semantic Relatedness task (Marelli et al., 2014). Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent.

Each STS task consists of 4-6 datasets covering a wide variety of domains, including newswire, tweets, glosses, machine translation outputs, web forums, news headlines, image and video captions, among others. Most submissions for these tasks use supervised models that are trained and tuned on provided training data or similar datasets from older tasks. Further details are provided in the official task descriptions (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015).

4.1.2 Preliminaries

For training data, we use pairs from PPDB. For word similarity experiments, we train on word pairs and for sentence similarity, we train on phrase pairs. PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones. The pairs in PPDB are sorted by a confidence measure and so the smaller sets contain higher precision paraphrases. PPDB is derived automatically from naturally-occurring bilingual text, and versions of PPDB have been released for many languages without the need for any manual annotation (Ganitkevitch and Callison-Burch, 2014).

Before training the CHARAGRAM model, we need to populate V, the vocabulary of character n-grams included in the model. We obtain these from the training data used for the final models in each setting, which is either the lexical or phrasal section of PPDB XXL. We tune over whether to include the full sets of character n-grams in these datasets or only those that appear more than once.

When extracting n-grams, we include spaces and add an extra space before and after each word or phrase in the training and evaluation data to ensure that the beginning and end of each word is represented. We note that strong performance can be obtained using far fewer character n-grams; we explore the effects of varying the number of n-grams and the n-gram orders in Section 4.4.

We used Adam (Kingma and Ba, 2014) with a learning rate of 0.001 to learn the parameters in the following experiments.

4.1.3 Word Embedding Experiments

Training and Tuning For hyperparameter tuning, we used one epoch on the lexical section of PPDB XXL, which consists of 770,007 word pairs. We used either WS353 or SL999 for model selection (reported below). We then took the selected hyperparameters and trained for 50 epochs to ensure that all models had a chance to converge.

Model	Tuned on	WS353	SL999
charCNN	SL999	26.31	30.64
	WS353	33.19	16.73
charLSTM	SL999	48.27	54.54
	WS353	51.43	48.83
CHARAGRAM	SL999	53.87	**63.33**
	WS353	**58.35**	60.00
inter-annotator agreement	-	75.6	78

Table 1: Word similarity results (Spearman's $\rho \times 100$). The inter-annotator agreement is the average Spearman's ρ between a single annotator and the average of all others.

Full details of our tuning procedure are provided in the supplementary material. In short, we tuned all models thoroughly, tuning the activation functions for CHARAGRAM and charCNN, as well as the regularization strength, mini-batch size, and sampling type for all models. For charCNN, we experimented with two filter sets: one uses 175 filters for each n-gram size $\in \{2, 3, 4\}$, and the other uses the set of filters from Kim et al. (2015), consisting of 25 filters of size 1, 50 of size 2, 75 of size 3, 100 of size 4, 125 of size 5, and 150 of size 6. We also experimented with using dropout (Srivastava et al., 2014) on the inputs to the final layer of charCNN in place of L_2 regularization, as well as removing the last feedforward layer. Neither variation significantly improved performance on our suite of tasks for word or sentence similarity. However, using more filters does improve performance, apparently linearly with the square of the number of filters.

Architecture Comparison The results are shown in Table 1. The CHARAGRAM model outperforms both the charLSTM and charCNN models, and also outperforms recent strong results on SL999.

We also found that the charCNN and charLSTM models take far more epochs to converge than the CHARAGRAM model. We noted this trend across experiments and explore it further in Section 4.3.

Comparison to Prior Work We found that performance of CHARAGRAM on word similarity tasks can be improved by using more character n-grams. This is explored in Section 4.4. Our best result from these experiments was obtained with the largest

Model	SL999
Hill et al. (2014)	52
Schwartz et al. (2015)	56
Faruqui and Dyer (2015)	58
Wieting et al. (2015)	66.7
CHARAGRAM (large)	**70.6**

Table 2: Spearman's $\rho \times 100$ on SL999. CHARAGRAM (large) refers to the CHARAGRAM model described in Section 4.4. This model contains 173,881 character n-grams, more than the 100,283 in the CHARAGRAM model used in Table 1.

model we considered, which contains 173,881 n-gram embeddings. When using WS353 for model selection and training for 25 epochs, this model achieves 70.6 on SL999. To our knowledge, this is the best result reported on SL999 in this setting; Table 2 shows comparable recent results. Note that a higher SL999 number is reported by Mrkšić et al. (2016), but the setting is not comparable to ours as they started with embeddings tuned on SL999.

Lastly, we evaluated our model on the Stanford Rare Word Similarity Dataset (Luong et al., 2013), using SL999 for model selection. We obtained a Spearman's ρ of 47.1, which outperforms the 41.8 result from Soricut and Och (2015) and is competitive with the 47.8 reported by Pennington et al. (2014), which used a 42B-token corpus for training.

4.1.4 Sentence Embedding Experiments

Training and Tuning We did initial training of our models using one pass through PPDB XL, which consists of 3,033,753 unique phrase pairs. Following Wieting et al. (2016), we use the annotated phrase pairs developed by Pavlick et al. (2015) as our validation set, using Spearman's ρ to rank the models. We then take the highest performing models and train on the 9,123,575 unique phrase pairs in the phrasal section of PPDB XXL for 10 epochs.

For all experiments, we fix the mini-batch size to 100, the margin δ to 0.4, and use MAX sampling (see supplementary material). For CHARAGRAM, V contains all 122,610 character n-grams ($n \in \{2, 3, 4\}$) in the PPDB XXL phrasal section. Other tuning settings are the same as Section 4.1.3.

For another baseline, we train the PARAGRAM-PHRASE model of Wieting et al. (2016), tuning its regularization strength over $\{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. The PARAGRAM-PHRASE model simply uses word averaging as its composition function, but outperforms many more complex models.

In this section, we refer to our model as CHARAGRAM-PHRASE because the input is a character sequence containing multiple words rather than only a single word as in Section 4.1.3. Since the vocabulary V is defined by the training data sequences, the CHARAGRAM-PHRASE model includes character n-grams that span multiple words, permitting it to capture some aspects of word order and word co-occurrence, which the PARAGRAM-PHRASE model is unable to do.

We encountered difficulties training the charLSTM and charCNN models for this task. We tried several strategies to improve their chance at convergence, including clipping gradients, increasing training data, and experimenting with different optimizers and learning rates. We found success by using the original (confidence-based) ordering of the PPDB phrase pairs for the initial epoch of learning, then shuffling them for subsequent epochs. This is similar to curriculum learning (Bengio et al., 2009). The higher-confidence phrase pairs tend to be shorter and have many overlapping words, possibly making them easier to learn from.

Results An abbreviated version of the sentence similarity results is shown in Table 3; the supplementary material contains the full results. For comparison, we report performance for the median (50%), third quartile (75%), and top-performing (Max) systems from the shared tasks. We observe strong performance for the CHARAGRAM-PHRASE model. It always does better than the charCNN and charLSTM models, and outperforms the PARAGRAM-PHRASE model on 15 of the 22 tasks. Furthermore, CHARAGRAM-PHRASE matches or exceeds the top-performing task-tuned systems on 5 tasks, and is within 0.003 on 2 more. The charLSTM and charCNN models are significantly worse, with the charCNN being the better of the two and beating PARAGRAM-PHRASE on 4 of the tasks.

We emphasize that there are many other models that could be compared to, such as an LSTM over word embeddings. This and many other models were explored by Wieting et al. (2016). Their PARAGRAM-PHRASE model, which simply learns word embeddings within an averaging composition

Dataset	50%	75%	Max	charCNN	charLSTM	PARAGRAM-PHRASE	CHARAGRAM-PHRASE
STS 2012 Average	54.5	59.5	70.3	56.5	40.1	58.5	**66.1**
STS 2013 Average	45.3	51.4	65.3	47.7	30.7	**57.7**	57.2
STS 2014 Average	64.7	71.4	76.7	64.7	46.8	71.5	**74.7**
STS 2015 Average	70.2	75.8	80.2	66.0	45.5	75.7	**76.1**
2014 SICK	71.4	79.9	82.8	62.9	50.3	**72.0**	70.0
2015 Twitter	49.9	52.5	61.9	48.6	39.9	52.7	**53.6**
Average	59.7	65.6	73.6	59.2	41.9	66.2	**68.7**

Table 3: Results on SemEval textual similarity datasets (Pearson's $r \times 100$). The highest score in each row is in boldface (omitting the official task score columns). The last row shows the average performance over all 22 textual similarity datasets

Model	Accuracy (%)
charCNN	97.02
charLSTM	96.90
CHARAGRAM	96.99
CHARAGRAM (2-layer)	**97.10**

Table 4: Results on part-of-speech tagging.

function, was among their best-performing models. We used this model in our experiments as a strongly-performing representative of their results.

Lastly, we note other recent work that considers a similar transfer learning setting. The FastSent model (Hill et al., 2016) uses the 2014 STS task in its evaluation and reports an average Pearson's r of 61.3. On the same data, the C-PHRASE model (Pham et al., 2015) has an average Pearson's r of 65.7.[2] Both results are lower than the 74.7 achieved by CHARAGRAM-PHRASE on this dataset.

4.2 POS Tagging Experiments

We now consider part-of-speech (POS) tagging, since it has been used as a testbed for evaluating architectures for character-level word representations. It also differs from semantic similarity, allowing us to evaluate our architectures on a syntactic task. We replicate the POS tagging experimental setup of Ling et al. (2015a). Their model uses a bidirectional LSTM over character embeddings to represent words. They then use the resulting word representations in another bidirectional LSTM that predicts the tag for each word. We replace their character bidirectional LSTM with our three architectures: charCNN, charLSTM, and CHARAGRAM.

We use the Wall Street Journal portion of the Penn Treebank, using Sections 1-18 for training, 19-21 for tuning, and 22-24 for testing. We set the dimensionality of the character embeddings to 50 and that of the (induced) word representations to 150. For optimization, we use stochastic gradient descent with a mini-batch size of 100 sentences. The learning rate and momentum are set to 0.2 and 0.95 respectively. We train the models for 50 epochs, again to ensure that all models have an opportunity to converge.

The other settings for our models are mostly the same as for the word and sentence experiments (Section 4.1). We again use character n-grams with $n \in \{2, 3, 4\}$, tuning over whether to include all 54,893 in the training data or only those that occur more than once. However, there are two minor differences from the previous sections. First, we add a single binary feature to indicate if the token contains a capital letter. Second, our tuning considers rectified linear units as the activation function for the CHARAGRAM and charCNN architectures.[3]

The results are shown in Table 4. Performance is similar across models. We found that adding a second fully-connected 150 dimensional layer to the CHARAGRAM model improved results slightly.[4]

4.3 Convergence

One observation we made during our experiments was that different models converged at significantly different rates. Figure 1 plots the performance of the word similarity and tagging tasks as a function of training epoch. For word similarity, we plot the oracle Spearman's ρ on SL999, while for tagging we plot accuracy on the tuning set. We evaluate every quarter epoch (approximately every 194,252 word pairs) for word similarity and every epoch for tag-

[2]Both the results for FastSent and C-PHRASE were computed from Table 4 in (Hill et al., 2016).

[3]We did not consider ReLU for the similarity experiments because the final embeddings are used directly to compute cosine similarities, which led to poor performance when restricting the embeddings to be non-negative.

[4]We also tried adding a second (300 dimensional) layer for the word and sentence similarity models and found it to hurt performance.

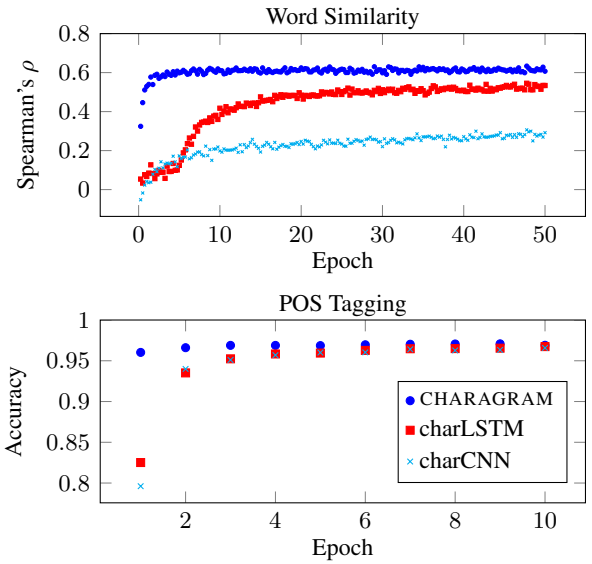


Figure 1: Plots of performance versus training epoch for word similarity and POS tagging.

ging. We only show the first 10 epochs of training in the tagging plot.

The plots show that the CHARAGRAM model converges quickly to high performance. The charCNN and charLSTM models take many more epochs to converge. Even with tagging, which uses a very high learning rate, CHARAGRAM converges significantly faster than the others. For word similarity, it appears that charCNN and charLSTM are still slowly improving at the end of 50 epochs. This suggests the possibility that these models could eventually surpass CHARAGRAM with more epochs. However, due to the large training sets available from PPDB and the computational requirements of these architectures, we were unable to explore the regime of training for many epochs. We conjecture that slow convergence could also be the reason for the inferior performance of LSTMs for similarity tasks as reported by Wieting et al. (2016).

4.4 Model Size Experiments

The default setting for our CHARAGRAM and CHARAGRAM-PHRASE models is to use all character bigram, trigrams, and 4-grams that occur in the training data at least C times, tuning C over the set $\{1, 2\}$. This results in a large number of parameters, which could be seen as an unfair advantage over the comparatively smaller charCNN and charLSTM similarity models, which have up to 881,025 and 763,200 parameters respectively (including 134 character embeddings for each).

Task	# n-grams	2	2,3	2,3,4	2,3,4,5	2,3,4,5,6
POS Tagging	100	95.52	96.09	96.15	96.13	96.16
	1,000	96.72	96.86	96.97	97.02	97.03
	50,000	96.81	97.00	97.02	97.04	**97.09**
Word Similarity	100	6.2	7.0	7.7	9.1	8.8
	1,000	15.2	33.0	38.7	43.2	43.9
	50,000	14.4	52.4	67.8	69.2	**69.5**
Sentence Similarity	100	40.2	33.8	32.5	31.2	29.8
	1,000	50.1	60.3	58.6	56.6	55.6
	50,000	45.7	64.7	**66.6**	63.0	61.3

Table 5: Results of using different numbers and different combinations of character n-grams.

However, for a given sequence, very few parameters in the CHARAGRAM model are actually used. For charCNN and charLSTM, by contrast, all parameters are used except character embeddings for characters not present in the sequence. For a 100-character sequence, the 300-dimensional CHARAGRAM model uses approximately 90,000 parameters, about one-tenth of those used by charCNN and charLSTM for the same sequence.

We performed a series of experiments to investigate how the CHARAGRAM and CHARAGRAM-PHRASE models perform with different numbers and lengths of character n-grams. For a given k, we took the k most frequent character n-grams for each value of n in use. We experimented with k values in $\{100, 1000, 50000\}$. If there were fewer than k unique character n-grams for a given n, we used all of them. For these experiments, we did very little tuning, setting the regularization strength to 0 and only tuning the activation function. For word similarity, we report performance on SL999 after 5 training epochs on the lexical section of PPDB XXL. For sentence similarity, we report the average Pearson's r over all 22 datasets after 5 training epochs on the phrasal section of PPDB XL. For tagging, we report accuracy on the tuning set after 50 training epochs.

The results are shown in Table 5. When using extremely small models with only 100 n-grams of each order, we still see relatively strong performance on tagging. However, the similarity tasks require far more n-grams to yield strong performance. Using 1000 n-grams clearly outperforms 100, and 50,000 n-grams performs best. We also found that models converged more quickly on tagging than on the similarity tasks. We suspect this is due to differences in task complexity. In tagging, the model does not

need to learn all facets of each word's semantics; it only needs to map a word to its syntactic categories. Therefore, simple surface-level features like affixes can help tremendously. However, learning representations that reflect detailed differences in word meaning is a more fine-grained endeavor and this is presumably why larger models are needed and convergence is slower.

5 Analysis

5.1 Quantitative Analysis

One of our primary motivations for character-based models is to address the issue of out-of-vocabulary (OOV) words, which were found to be one of the main sources of error for the PARAGRAM-PHRASE model from Wieting et al. (2016). They reported a negative correlation (Pearson's r of -0.45) between OOV rate and performance. We took the 12,108 sentence pairs in all 20 SemEval STS tasks and binned them by the total number of unknown words in the pairs.[5] We computed Pearson's r over each bin. The results are shown in Table 6.

Number of Unknown Words	N	PARAGRAM-PHRASE	CHARAGRAM-PHRASE
0	11,292	71.4	**73.8**
1	534	68.8	**78.8**
2	194	66.4	**72.8**
≥ 1	816	68.6	**77.9**
≥ 0	12,108	71.0	**74.0**

Table 6: Performance (Pearson's $r \times 100$) as a function of the number of unknown words in the sentence pairs over all 20 SemEval STS datasets. N is the number of sentence pairs.

The CHARAGRAM-PHRASE model has better performance for each number of unknown words. The PARAGRAM-PHRASE model degrades when more unknown words are present, presumably because it is forced to use the same unknown word embedding for all unknown words. The CHARAGRAM-PHRASE model has no notion of unknown words, as it can embed any character sequence.

We next investigated the sensitivity of the two models to length, as measured by the maximum of the lengths of the two sentences in a pair. We binned all of the 12,108 sentence pairs in the 20 SemEval STS tasks by length and then again found the Pearson's r for both the PARAGRAM-PHRASE and CHARAGRAM-PHRASE models. The results are shown in Table 7.

Max Length	N	PARAGRAM-PHRASE	CHARAGRAM-PHRASE
≤ 4	71	67.9	**72.9**
5	216	71.1	**71.9**
6	572	67.0	**69.7**
7	1,097	71.5	**74.0**
8	1,356	74.2	**74.5**
9	1,266	71.7	**72.7**
10	1,010	70.7	**74.2**
11-15	3,143	71.8	**73.7**
16-20	1,559	73.0	**75.1**
≥ 21	1,818	74.5	**75.4**

Table 7: Performance (Pearson's $r \times 100$) as a function of the maximum number of tokens in the sentence pairs over all 20 SemEval STS datasets. N is the number of sentence pairs.

Both models are robust to sentence length, achieving the highest correlations on the longest sentences. We also find that CHARAGRAM-PHRASE outperforms PARAGRAM-PHRASE at all sentence lengths.

5.2 Qualitative Analysis

Bigram	CHARAGRAM-PHRASE	PARAGRAM-PHRASE
not capable	incapable, unable, incapacity	not, capable, stalled
not able	unable, incapable, incapacity	not, able, stalled
not possible	impossible impracticable unable	not, stalled, possible
not sufficient	insufficient, sufficient, inadequate	not, sufficient, stalled
not easy	easy, difficult, tough	not, stalled, easy

Table 8: Nearest neighboring words of selected bigrams under CHARAGRAM-PHRASE and PARAGRAM-PHRASE embeddings.

Aside from OOVs, the PARAGRAM-PHRASE model lacks the ability to model word order or cooccurrence, since it simply averages the words in the sequence. We were interested to see whether CHARAGRAM-PHRASE could handle negation, since it does model limited information about word order (via character n-grams that span multiple words). We made a list of "not" bigrams that could be represented by a single word, then embedded each bigram using both models and did a nearest-neighbor search over a working vocabulary.[6] The results, in Table 8, show how the CHARAGRAM-PHRASE embeddings model negation. In all cases but one, the near-

[5] Unknown words were defined as those not present in the 1.7 million unique (case-insensitive) tokens that comprise the vocabulary for the GloVe embeddings available at `http://nlp.stanford.edu/projects/glove/`. The PARAGRAM-SL999 embeddings, used to initialize the PARAGRAM-PHRASE model, use this same vocabulary.

[6] This has all words in PPDB-XXL, our evaluations, and two other datasets: SST (Socher et al., 2013) and SNLI (Bowman et al., 2015), resulting in 93,217 unique (up-to-casing) tokens.

Word	Nearest Neighbors
vehicals	vehical, vehicles, vehicels, vehicular, cars, vehicle, automobiles, car
serious-looking	serious, grave, acute, serious-minded, seriousness, gravity, serious-faced
near-impossible	impossible, hard/impossible, audacious-impossible, impractical, unable
growths	growth, grow, growing, increases, grows, increase, rise, growls, rising
litered	liter, litering, lited, liters, literate, literature, literary, literal, lite, obliterated
journeying	journey, journeys, voyage, trip, roadtrip, travel, tourney, voyages, road-trip
babyyyyyy	babyyyyyyy, baby, babys, babe, baby.i, babydoll, babycake, darling
adirty	dirty, dirtyyyyyy, filthy, down-and-dirty, dirtying, dirties, ugly, dirty-blonde
refunding	refunds, refunded, refund, repayment, reimbursement, rebate, repay reimbursements, reimburse, repaying, repayments, rebates, rebating, reimburses
professors	professor, professorships, professorship, teachers, professorial, teacher prof., teaches, lecturers, teachings, instructors, headteachers, teacher-student
huge	enormous, tremendous, large, big, vast, overwhelming, immense, giant formidable, considerable, massive, huger, large-scale, great, daunting

Table 9: Nearest neighbors of CHARAGRAM-PHRASE embeddings. Above the double horizontal line are nearest neighbors of words that were not in our training data, and below it are nearest neighbors of words that were in our training data.

est neighbor is a paraphrase for the bigram and the next neighbors are mostly paraphrases as well. The PARAGRAM-PHRASE model, unsurprisingly, is incapable of modeling negation. In all cases, the nearest neighbor is *not*, as it carries much more weight than the word it modifies. The remaining nearest neighbors are either the modified word or *stalled*.

We did two additional nearest neighbor explorations with our CHARAGRAM-PHRASE model. First, we collected nearest neighbors for words that were not in the training data (i.e., PPDB XXL), but were in our working vocabulary. These are shown in the upper part of Table 9. In the second, we collected nearest neighbors of words that were in our training data, shown in the lower part of Table 9.

Several kinds of similarity are being captured simultaneously. One kind is similarity in terms of spelling variation, including misspellings (*vehicals*, *vehicels*) and repetition for emphasis (*babyyyyyyy*). Another kind is similarity in terms of morphological variants of a shared root (e.g., *journeying* and *journey*). We also find many synonym relationships without significant amounts of overlapping characters (e.g., *vehicles*, *cars*, *automobiles*). Words in the training data, which tend to be more commonly used, do tend to have higher precision in their nearest neighbors (e.g., neighbors of *huge*). We see occasional mistakes for words that share many characters but are not paraphrases (e.g., *litered*, a likely misspelling of *littered*).

Lastly, since our model learns embeddings for character n-grams, we show an analysis of character n-gram nearest neighbors in Table 10. They appear to be grouped into themes, such as death (row 1), food (row 2), and speed (row 3), but have different granularities. The n-grams in the last row appear in paraphrases of *2*, whereas the second-to-last row shows n-grams in words related to language.

n-gram	Nearest Neighbors
die	_dy, _die, dead,_dyi, rlif, mort, ecea, rpse, d_aw
foo	_foo,_eat, meal, alim, trit, feed, grai,_din, nutr, toe
pee	peed, hast, spee, fast, mpo_, pace, _vel, loci, ccel
aiv	waiv, aive, boli, epea, ncel, abol, lift, bort, bol
ngu	ngue, uist, ongu, tong, abic, gual, fren, ocab, ingu
_2	2_, _02, _02_,_tw,_dua,_xx,_ii_, xx, o_14, d_2

Table 10: Nearest neighbors of character n-gram embeddings from trained CHARAGRAM-PHRASE model. The underscore indicates a space, which signals the beginning or end of a word.

6 Conclusion

We performed a careful empirical comparison of character-based compositional architectures on three NLP tasks. We found a consistent trend: the simplest architecture converges fastest to high performance. These results, coupled with those from Wieting et al. (2016), suggest that practitioners should begin with simple architectures rather than moving immediately to RNNs and CNNs. We release our code and trained models so they can be used by the NLP community for general-purpose, character-based text representation.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We thank the developers of Theano (Theano Development Team, 2016) and NVIDIA Corporation for donating GPUs used in this research.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and D. Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of ACL*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. *arXiv preprint arXiv:1309.4628*.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Marta R. Costa-Jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810*.

Cicero dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*.

Cicero dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau. 2013. Morpheme-based feature-rich language models using deep neural networks for LVCSR of Egyptian Arabic. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. *arXiv preprint arXiv:1506.05230*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

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

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, 3.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*.

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

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

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

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

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

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1).

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*.

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

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*.

Length bias in Encoder Decoder Models and a Case for Global Conditioning

Pavel Sountsov
Google
siege@google.com

Sunita Sarawagi *
IIT Bombay
sunita@iitb.ac.in

Abstract

Encoder-decoder networks are popular for modeling sequences probabilistically in many applications. These models use the power of the Long Short-Term Memory (LSTM) architecture to capture the *full dependence* among variables, unlike earlier models like CRFs that typically assumed conditional independence among non-adjacent variables. However in practice encoder-decoder models exhibit a bias towards short sequences that surprisingly gets worse with increasing beam size.

In this paper we show that such phenomenon is due to a discrepancy between the full sequence margin and the per-element margin enforced by the locally conditioned training objective of a encoder-decoder model. The discrepancy more adversely impacts long sequences, explaining the bias towards predicting short sequences.

For the case where the predicted sequences come from a closed set, we show that a globally conditioned model alleviates the above problems of encoder-decoder models. From a practical point of view, our proposed model also eliminates the need for a beam-search during inference, which reduces to an efficient dot-product based search in a vector-space.

1 Introduction

In this paper we investigate the use of neural networks for modeling the conditional distribution $\Pr(\mathbf{y}|\mathbf{x})$ over sequences $\mathbf{y}$ of discrete tokens in response to a complex input $\mathbf{x}$, which can be another sequence or an image. Such models have applications in machine translation (Bahdanau et al., 2014; Sutskever et al., 2014), image captioning (Vinyals et al., 2015), response generation in emails (Kannan et al., 2016), and conversations (Khaitan, 2016; Vinyals and Le, 2015; Li et al., 2015).

The most popular neural network for probabilistic modeling of sequences in the above applications is the encoder-decoder (ED) network (Sutskever et al., 2014). A ED network first encodes an input $\mathbf{x}$ into a vector which is then used to initialize a recurrent neural network (RNN) for decoding the output $\mathbf{y}$. The decoder RNN factorizes $\Pr(\mathbf{y}|\mathbf{x})$ using the chain rule as $\prod_j \Pr(y_j|y_1, \ldots, y_{j-1}, \mathbf{x})$ where $y_1, \ldots, y_n$ denote the tokens in $\mathbf{y}$. This factorization does not entail any conditional independence assumption among the $\{y_j\}$ variables. This is unlike earlier sequence models like CRFs (Lafferty et al., 2001) and MeMMs (McCallum et al., 2000) that typically assume that a token is independent of all other tokens given its adjacent tokens. Modern-day RNNs like LSTMs promise to capture non-adjacent and long-term dependencies by summarizing the set of previous tokens in a continuous, high-dimensional state vector. Within the limits of parameter capacity allocated to the model, the ED, by virtue of exactly factorizing the token sequence, is consistent.

However, when we created and deployed an ED model for a chat suggestion task we observed several counter-intuitive patterns in its predicted outputs. Even after training the model over billions of examples, the predictions were systematically biased towards short sequences. Such bias has also been seen in translation (Cho et al., 2014). Another curious

* Work done while visiting Google Research on a leave from IIT Bombay.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1516–1525,
Austin, Texas, November 1-5, 2016.

phenomenon was that the accuracy of the predictions sometimes dropped with increasing beam-size, more than could be explained by statistical variations of a well-calibrated model (Ranzato et al., 2016).

In this paper we expose a margin discrepancy in the training loss of encoder-decoder models to explain the above problems in its predictions. We show that the training loss of ED network often underestimates the margin of separating a correct sequence from an incorrect shorter sequence. The discrepancy gets more severe as the length of the correct sequence increases. That is, even after the training loss converges to a small value, full inference on the training data can incur errors causing the model to be underfitted for long sequences in spite of low training cost. We call this the length bias problem.

We propose an alternative model that avoids the margin discrepancy by globally conditioning the $P(\mathbf{y}|\mathbf{x})$ distribution. Our model is applicable in the many practical tasks where the space of allowed outputs is closed. For example, the responses generated by the smart reply feature of Inbox is restricted to lie within a hand-screened whitelist of responses $\mathcal{W} \subset \mathcal{Y}$ (Kannan et al., 2016), and the same holds for a recent conversation assistant feature of Google's Allo (Khaitan, 2016). Our model uses a second RNN encoder to represent the output as another fixed length vector. We show that our proposed encoder-encoder model produces better calibrated whole sequence probabilities and alleviates the length-bias problem of ED models on two conversation tasks. A second advantage of our model is that inference is significantly faster than ED models and is guaranteed to find the globally optimal solution. In contrast, inference in ED models requires an expensive beam-search which is both slow and is not guaranteed to find the optimal sequence.

2 Length Bias in Encoder-Decoder Models

In this section we analyze the widely used encoder-decoder neural network for modeling $\Pr(\mathbf{y}|\mathbf{x})$ over the space of discrete output sequences. We use $y_1, \ldots, y_n$ to denote the tokens in a sequence $\mathbf{y}$. Each y_i is a discrete symbol from a finite dictionary V of size m. Typically, m is large. The length n of a sequence is allowed to vary from sequence to sequence even for the same input $\mathbf{x}$. A special token EOS $\in V$ is used to mark the end of a sequence. We use $\mathcal{Y}$ to denote the space of such valid sequences and θ to denote the parameters of the model.

2.1 The encoder-decoder network

The Encoder-Decoder (ED) network represents $\Pr(\mathbf{y}|\mathbf{x}, \theta)$ by applying chain rule to exactly factorize it as $\prod_{t=1}^{n} \Pr(y_t|y_1, \ldots, y_{t-1}, \mathbf{x}, \theta)$. First, an encoder with parameters $\theta_x \subset \theta$ is used to transform $\mathbf{x}$ into a d-dimensional real-vector $\mathbf{v}_x$. The network used for the encoder depends on the form of $\mathbf{x}$ — for example, when $\mathbf{x}$ is also a sequence, the encoder could be a RNN. The decoder then computes each $\Pr(y_t|y_1, \ldots, y_{t-1}, \mathbf{v}_x, \theta)$ as

$$\Pr(y_t|y_1, \ldots, y_{t-1}, \mathbf{v}_x, \theta) = P(y_t|\mathbf{s}_t, \theta), \qquad (1)$$

where $\mathbf{s}_t$ is a state vector implemented using a recurrent neural network as

$$\mathbf{s}_t = \begin{cases} \mathbf{v}_x & \text{if } t = 0, \\ \text{RNN}(\mathbf{s}_{t-1}, \theta_{E,y_{t-1}}, \theta_R) & \text{otherwise.} \end{cases} \qquad (2)$$

where RNN() is typically a stack of LSTM cells that captures long-term dependencies, $\theta_{E,y} \subset \theta$ are parameters denoting the embedding for token y, and $\theta_R \subset \theta$ are the parameters of the RNN. The function $\Pr(y|\mathbf{s}, \theta_y)$ that outputs the distribution over the m tokens is a softmax:

$$\Pr(y|\mathbf{s}, \theta) = \frac{e^{\mathbf{s}\theta_{S,y}}}{e^{\mathbf{s}\theta_{S,1}} + \ldots + e^{\mathbf{s}\theta_{S,m}}}, \qquad (3)$$

where $\theta_{S,y} \subset \theta$ denotes the parameters for token y in the final softmax.

2.2 The Origin of Length Bias

The ED network builds a single probability distribution over sequences of arbitrary length. For an input $\mathbf{x}$, the network needs to choose the highest probability $\mathbf{y}$ among valid candidate sequences of widely different lengths. Unlike in applications like entity-tagging and parsing where the length of the output is determined based on the input, in applications like response generation valid outputs can be of widely varying length. Therefore, $\Pr(\mathbf{y}|\mathbf{x}, \theta)$ should be well-calibrated over all sequence lengths. Indeed under infinite data and model capacity the ED model is consistent and will represent all sequence lengths faithfully. In practice when training data is

finite, we show that the ED model is biased against long sequences. Other researchers (Cho et al., 2014) have reported this bias but we are not aware of any analysis like ours explaining the reasons of this bias.

Claim 2.1. *The training loss of the ED model underestimates the margin of separating long sequences from short ones.*

Proof. Let $\mathbf{x}$ be an input for which a correct output $\mathbf{y}^+$ is of length ℓ and an incorrect output $\mathbf{y}^-$ is of length 1. Ideally, the training loss should put a positive margin between $\mathbf{y}^+$ and $\mathbf{y}^-$ which is $\log \Pr(\mathbf{y}^+|\mathbf{x}) - \log \Pr(\mathbf{y}^-|\mathbf{x})$. Let us investigate if the maximum likelihood training objective of the ED model achieves that. We can write this objective as:

$$\max_{\theta} \log \Pr(y_1^+|\mathbf{x},\theta) + \sum_{j=2}^{\ell} \log \Pr(y_j^+|y_{1\ldots j-1}^+, \mathbf{x}, \theta). \tag{4}$$

Only the first term in the above objective is involved in enforcing a margin between $\mathbf{y}^+$ and $\mathbf{y}^-$ because $\log \Pr(y_1^+|\mathbf{x})$ is maximized when $\log \Pr(y_1^-|\mathbf{x})$ is correspondingly minimized. Let $m_L(\theta) = \log \Pr(y_1^+|\mathbf{x},\theta) - \log \Pr(y_1^-|\mathbf{x},\theta)$, the local margin from the first position and $m_R(\theta) = \sum_{j=2}^{\ell} \log \Pr(y_j^+|\mathbf{y}_{1\ldots j-1}^+, \mathbf{x}, \theta)$. It is easy to see that our desired margin between $\mathbf{y}^+$ and $\mathbf{y}^-$ is $\log \Pr(\mathbf{y}^+|\mathbf{x}) - \log \Pr(\mathbf{y}^-|\mathbf{x}) = m_L + m_R$. Let $m_g = m_L + m_R$. Assuming two possible labels for the first position ($m = 2$) [1], the training objective in Equation 4 can now be rewritten in terms of the margins as:

$$\min_{\theta} \log(1 + e^{-m_L(\theta)}) - m_R(\theta)$$

We next argue that this objective is not aligned with our ideal goal of making the global margin $m_L + m_R$ positive.

First, note that m_R is a log probability which under finite parameters will be non-zero. Second, even though m_L can take any arbitrary finite value, the training objective drops rapidly when m_L is positive. When training objective is regularized and training data is finite, the model parameters θ cannot take very large values and the trainer will converge at a small positive value of m_L. Finally, we show that the value of m_R decreases with increasing sequence length. For each position j in the sequence, we add to m_R log-probability of y_j^+. The maximum value of $\log \Pr(y_j^+|\mathbf{y}_{1\ldots j-1}^+, \mathbf{x}, \theta)$ is $\log(1-\epsilon)$ where ϵ is non-zero and decreasing with the magnitude of the parameters θ. In general, $\log \Pr(y_j^+|\mathbf{y}_{1\ldots j-1}^+, \mathbf{x}, \theta)$ can be a much smaller negative value when the input $\mathbf{x}$ has multiple correct responses as is common in conversation tasks. For example, an input like $\mathbf{x}$ ='How are you?', has many possible correct outputs: $\mathbf{y} \in$ {'I am good', 'I am great', 'I am fine, how about you?', etc}. Let f_j denote the relative frequency of output y_j^+ among all correct responses with prefix $\mathbf{y}_{1\ldots j-1}^+$. The value of m_R will be upper bounded as

$$m_R \leq \sum_{j=2}^{\ell} \log \min(1-\epsilon, f_j)$$

This term is negative always and increases in magnitude as sequence length increases and the set of positive outpus have high entropy. In this situation, when combined with regularization, our desired margin m_g may not remain positive even though m_L is positive. In summary, the core issue here is that since the ED loss is optimized and regularized on the local problem it does not control for the global, task relevant margin. □

This mismatch between the local margin optimized during training and the global margin explains the length bias observed by us and others (Cho et al., 2014). During inference a shorter sequence for which m_R is smaller wins over larger sequences.

This mismatch also explains why increasing beam size leads to a drop in accuracy sometimes (Ranzato et al., 2016)[2]. When beam size is large, we are more likely to dig out short sequences that have otherwise been separated by the local margin. We show empirically in Section 4.3 that for long sequences larger beam size hurts accuracy whereas for small sequences the effect is the opposite.

2.3 Proposed fixes to the ED models

Many ad hoc approaches have been used to alleviate length bias directly or indirectly. Some resort to nor-

[1] For $m > 2$, the objective will be upper bounded by $\min_\theta \log(1 + (m-1)e^{-m_L(\theta)}) - m_R(\theta)$. The argument that follows remains largely unchanged

[2] Figure 6 in the paper shows a drop in BLEU score by 0.5 as the beam size is increased from 3 to 10.

malizing the probability by the full sequence length (Cho et al., 2014; Graves, 2013) whereas (Abadie et al., 2014) proposes segmenting longer sentences into shorter phrases. (Cho et al., 2014) conjectures that the length bias of ED models could be because of limited representation power of the encoder network. Later more powerful encoders based on attention achieved greater accuracy (Bahdanau et al., 2014) on long sequences. Attention can be viewed as a mechanism of improving the capacity of the local models, thereby making the local margin m_L more definitive. But attention is not effective for all tasks — for example, (Vinyals and Le, 2015) report that attention was not useful for conversation.

Recently (Bengio et al., 2015; Ranzato et al., 2016) propose another modification to the ED training objective where the true token y_{j-1} in the training term $\log \Pr(y_j|y_1, \ldots, y_{j-1})$ is replaced by a sample or top-k modes from the posterior at position $j-1$ via a careful schedule. Incidently, this fix also helps to indirectly alleviate the length bias problem. The sampling causes incorrect tokens to be used as previous history for producing a correct token. If earlier the incorrect token was followed by a low-entropy EOS token, now that state should also admit the correct token causing a decrease in the probability of EOS, and therefore the short sequence.

In the next section we propose our more direct fix to the margin discrepancy problem.

3 Globally Conditioned Encoder-Encoder Models

We represent $\Pr(\mathbf{y}|\mathbf{x}, \theta)$ as a globally conditioned model $\frac{e^{s(\mathbf{y}|\mathbf{x},\theta)}}{Z(\mathbf{x},\theta)}$ where $s(\mathbf{y}|\mathbf{x}, \theta)$ denotes a score for output $\mathbf{y}$ and $Z(\mathbf{x}, \theta)$ denotes the shared normalizer. We show in Section 3.3 why such global conditioning solves the margin discrepancy problem of the ED model. The intractable partition function in global conditioning introduces several new challenges during training and inference. In this section we discuss how we designed our network to address them.

Our model assumes that during inference the output has to be selected from a given whitelist of responses $\mathcal{W} \subset \mathcal{Y}$. In spite of this restriction, the problem does not reduce to multi-class classification because of two important reasons. First, during training we wish to tap all available input-output pairs including the significantly more abundant outputs that do not come from the whitelist. Second, the whitelist could be very large and treating each output sequence as an atomic class can limit generalization achievable by modeling at the level of tokens in the sequence.

3.1 Modeling $s(\mathbf{y}|\mathbf{x}, \theta)$

We use a second encoder to convert $\mathbf{y}$ into a vector $\mathbf{v}_y$ of the same size as the vector $\mathbf{v}_x$ obtained by encoding $\mathbf{x}$ as in a ED network. The parameters used to encode $\mathbf{v}_x$ and $\mathbf{v}_y$ are disjoint. As we are only interested in a fixed dimensional output, unlike in ED networks, we have complete freedom in choosing the type of network to use for this second encoder. For our experiments, we have chosen to use an RNN with LSTM cells. Experimenting with other network architectures, such as bidirectional RNNs remains an interesting avenue for future work. The score $s(\mathbf{y}|\mathbf{x}, \theta)$ is the dot-product between $\mathbf{v}_y$ and $\mathbf{v}_x$. Thus our model is

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{e^{\mathbf{v}_x^T \mathbf{v}_y}}{\sum_{\mathbf{y}' \in \mathcal{Y}} e^{\mathbf{v}_x^T \mathbf{v}_{y'}}}. \quad (5)$$

3.2 Training and Inference

During training we use maximum likelihood to estimate θ given a large set of valid input-output pairs $\{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^N, \mathbf{y}^N)\}$ where each $\mathbf{y}^i$ belongs to $\mathcal{Y}$ which in general is much larger than $\mathcal{W}$. Our main challenge during training is that $\mathcal{Y}$ is intractably large for computing Z. We decompose Z as

$$Z = e^{s(\mathbf{y}|\mathbf{x},\theta)} + \sum_{\mathbf{y}' \in \mathcal{Y} \setminus \mathbf{y}} e^{s(\mathbf{y}'|\mathbf{x},\theta)}, \quad (6)$$

and then resort to estimating the last term using importance sampling. Constructing a high quality proposal distribution over $\mathcal{Y} \setminus \mathbf{y}$ is difficult in its own right, so in practice, we make the following approximations. We extract the most common T sequences across a data set into a pool of negative examples. We estimate the empirical prior probability of the sequences in that pool, $Q(\mathbf{y})$, and then draw k samples from this distribution. We take care to remove the true sequence from this distribution so as to remove the need to estimate its prior probability.

During inference, given an input $\mathbf{x}$ we need to find $\text{argmax}_{\mathbf{y} \in \mathcal{W}} s(\mathbf{y}|\mathbf{x}, \theta)$. This task can be performed

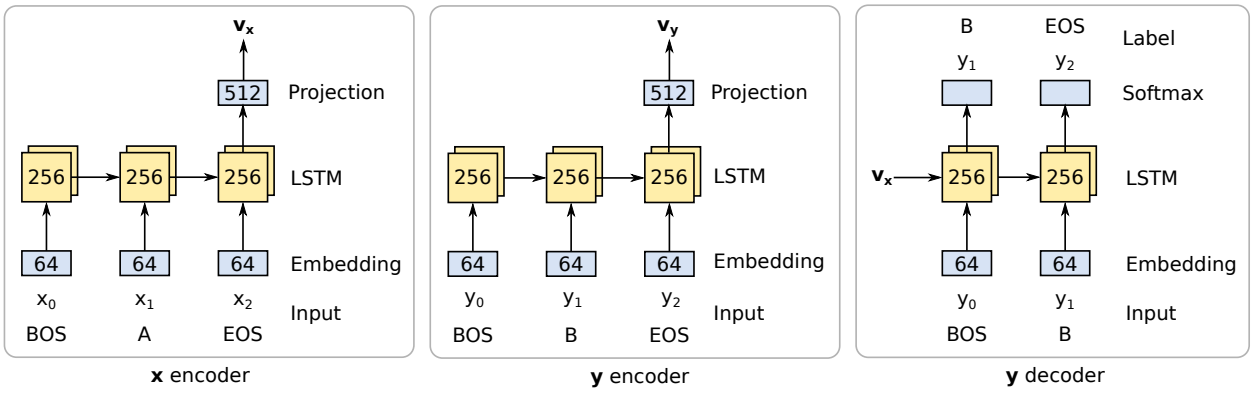


Figure 1: Neural network architectures used in our experiments. The context encoder network is used for both encoder-encoder and encoder-decoder models to encode the context sequence ('A') into a $\mathbf{v}_x$. For the encoder-encoder model, label sequence ('B') are encoded into $\mathbf{v}_y$ by the label encoder network. For the encoder-decoder network, the label sequence is decomposed using the chain rule by the decoder network.

efficiently in our network because the vectors $\mathbf{v}_y$ for the sequences $\mathbf{y}$ in the whitelist $\mathcal{W}$ can be pre-computed. Given an input $\mathbf{x}$, we compute $\mathbf{v}_x$ and take dot-product with the pre-computed vectors to find the highest scoring response. This gives us the optimal response. When $\mathcal{W}$ is very large, we can obtain an approximate solution by indexing the vectors $\mathbf{v}_y$ of $\mathcal{W}$ using recent methods specifically designed for dot-product based retrieval (Guo et al., 2016).

3.3 Margin

It is well-known that the maximum likelihood training objective of a globally normalized model is margin maximizing (Rosset et al., 2003). We illustrate this property using our set up from Claim 2.1 where a correct output $\mathbf{y}^+$ is of length ℓ and an incorrect output $\mathbf{y}^-$ is of length 1 with two possible labels for each position ($m = 2$).

The globally conditioned model learns a parameter per possible sequence and assigns the probability to each sequence using a softmax over those parameters. Additionally, we place a Gaussian prior on the parameters with a precision c. The loss for a positive example becomes:

$$\mathcal{L}_G(\mathbf{y}^+) = -\log \frac{e^{-\theta_{\mathbf{y}^+}}}{\sum_{\mathbf{y}'} e^{-\theta_{\mathbf{y}'}}} + \frac{c}{2} \sum_{\mathbf{y}'} \theta^2_{\mathbf{y}'},$$

where the sums are taken over all possible sequences.

We also train an ED model on this task. It also learns a parameter for every possible sequence, but assigns probability to each sequence using the chain rule. We also place the same Gaussian prior as above on the parameters. Let $\mathbf{y}_j$ denote the first j tokens $\{y_1, \ldots, y_j\}$ of sequence $\mathbf{y}$. The loss for a positive example for this model is then:

$$\mathcal{L}_L(\mathbf{y}^+) = -\sum_{j=1}^{\ell} \left(\log \frac{e^{-\theta_{\mathbf{y}^+_j}}}{\sum_{\mathbf{y}'_j} e^{-\theta_{\mathbf{y}'_j}}} + \frac{c}{2} \sum_{\mathbf{y}'_j} \theta^2_{\mathbf{y}'_j} \right),$$

where the inner sums are taken over all sequences of length j.

We train both models on synthetic sequences generated using the following rule. The first token is chosen to be '1' probability 0.6. If '1' is chosen, it means that this is a positive example and the remaining $\ell - 1$ tokens are chosen to be '1' with probability $0.9^{\frac{1}{\ell-1}}$. If a '0' is chosen as the first token, then that is a negative example, and the sequence generation does not go further. This means that there are $2^{\ell-1}$ unique positive sequences of length ℓ and one negative sequence of length 1. The remaining possible sequences do not occur in the training or testing data. By construction the unbiased margin between the most probable correct example and the incorrect example is length independent and positive. We sample 10000 such sequences and train both models using

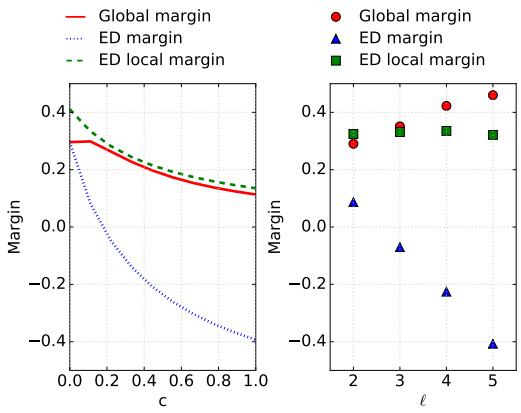


Figure 2: Comparing final margins of ED model with a globally conditioned model on example dataset of Section 3.3 as a function of regularization constant c and message length ℓ.

Adagrad (Duchi et al., 2011) for 1000 epochs with a learning rate of 0.1, effectively to convergence.

Figure 2 shows the margin for both models (between the most likely correct sequence and the incorrect sequence) and the local margin for the ED model at the end of training. On the left panel, we used sequences with $\ell = 2$ and varied the regularization constant c. When c is zero, both models learn the same global margin, but as it is increased the margin for the ED model decreases and becomes negative at $c > 0.2$, despite the local margin remaining positive and high. On the right panel we used $c = 0.1$ and varied ℓ. The ED model becomes unable to separate the sequences with length above 2 with this regularization constant setting.

4 Experiments

4.1 Datasets and Tasks

We contrast the quality of the ED and encoder-encoder models on two conversational datasets: Open Subtitles and Reddit Comments.

4.1.1 Open Subtitles Dataset

The Open Subtitles dataset consists of transcriptions of spoken dialog in movies and television shows (Lison and Tiedemann, 2016). We restrict our modeling only to the English subtitles, of which results in 319 million utternaces. Each utterance is tokenized into word and punctuation tokens, with the start and end marked by the BOS and EOS tokens. We randomly split out 90% of the utterances into the training set, placing the rest into the validation set. As the speaker information is not present in this data set, we treat each utterance as a label sequence, with the preceding utterances as context.

4.1.2 Reddit Comments Dataset

The Reddit Comments dataset is constructed from publicly available user comments on submissions on the Reddit website. Each submission is associated with a list of directed comment trees. In total, there are 41 million submissions and 501 million comments. We tokenize the individual comments in the same way as we have done with the utternaces in the Open Subtitles dataset. We randomly split 90% of the submissions and the associated comments into the training set, and the rest into the validation set. We use each comment (except the ones with no parent comments) as a label sequence, with the context sequence composed of its ancestor comments.

4.1.3 Whitelist and Vocabulary

From each dataset, we derived a dictionary of 20 thousand most commonly used tokens. Additionally, each dictionary contained the unknown token (UNK), BOS and EOS tokens. Tokens in the datasets which were not present in their associated vocabularies were replaced by the UNK token.

From each data set, we extracted 10 million most common label sequences that also contained at most 100 tokens. This set of sequences was used as the negative sample pool for the encoder-encoder models. For evaluation we created a whitelist $\mathcal{W}$ out of the 100 thousand most common sequences. We removed any sequence from this set that contained any UNK tokens to simplify inference.

4.1.4 Sequence Prediction Task

To evaluate the quality of these models, we task them to predict the true label sequence given its context. Due to the computational expense, we sub-sample the validation data sets to around 1 million context-label pairs. We additionally restrict the context-label pairs such that the label sequence is present in the evaluation set of common messages. We use recall@K as a measure of accuracy of the model predictions. It is defined as the fraction of test pairs where the correct label is within K most

probable predictions according to the model. For encoder-encoder models we use an exhaustive search over the evaluation set of common messages. For ED models we use a beam search with width ranging from 1 to 15 over a token prefix trie constructed from the sequences in $\mathcal{W}$.

4.2 Model Structure and Training Procedure

The context encoder, label encoder and decoder are implemented using LSTM recurrent networks (Hochreiter and Schmidhuber, 1997) with peephole connections (Sak et al., 2014). The context and label token sequences were mapped to embedding vectors using a lookup table that is trained jointly with the rest of the model parameters. The recurrent nets were unrolled in time up to 100 time-steps, with label sequences of greater length discarded and context sequences of greater length truncated.

The decoder in the ED model is trained by using the true label sequence prefix as input, and a shifted label sequence as output (Sutskever et al., 2014). The partition function in the softmax over tokens is estimated using importance sampling with a unigram distribution over tokens as the proposal distribution (Jean et al., 2014). We sample 512 negative examples from $Q(\mathbf{y})$ to estimate the partition function for the encoder-encoder model. See Figure 1 for connectivity and network size details.

All models were trained using Adagrad (Duchi et al., 2011) with an initial base learning rate of 0.1 which we exponentially decayed with a decade of 15 million steps. For stability, we clip the L2 norm of the gradients to a maximum magnitude of 1 as described in (Pascanu et al., 2012). All models are trained for 30 million steps with a mini-batch size of 64. The models are trained in a distributed manner on CPUs and NVidia GPUs using TensorFlow (Abadi et al., 2015).

4.3 Results

We first demonstrate the discrepancy between the local and global margin in the ED models as discussed in Section 3.3. We used a beam size of 15 to get the top prediction from our trained ED models on the test data and focussed on the subset for which the top prediction was incorrect. We measured local and global margin between the top predicted sequence ($\mathbf{y}^-$) and the correct test sequence ($\mathbf{y}^+$) as follows: Global margin is the difference in their full sequence log probability. Local margin is the difference in the local token probability of the smallest position j where $y_j^- \neq y_j^+$, that is local margin is $\Pr(y_j^+|\mathbf{y}^+_{1\ldots j-1}, \mathbf{x}, \theta) - \Pr(y_j^-|\mathbf{y}^+_{1\ldots j-1}, \mathbf{x}, \theta)$. Note the training loss of ED models directly compares only the local margin.

Global margin is much smaller than local margin In Figure 3 we show the local and global margin as a 2D histogram with color luminosity denoting frequency. We observe that the global margin values are much smaller than the local margins. The prominent spine is for $(\mathbf{y}^+, \mathbf{y}^-)$ pairs differing only in a single position making the local and global margins equal. Most of the mass is below the spine. For a significant fraction of cases (27% for Reddit, and 21% for Subtitles), the local margin is positive while the global margin is negative. That is, the ED loss for these sequences is small even though the log-probability of the correct sequence is much smaller than the log-probability of the predicted wrong sequence.

Beam search is not the bottleneck An interesting side observation from the plots in Figure 3 is that more than 98% of the wrong predictions have a negative margin, that is, the score of the correct sequence is indeed lower than the score of the wrong prediction. Improving the beam-width beyond 15 is not likely to improve these models since only in 1.9% and 1.7% of the cases is the correct score higher than the score of the wrong prediction.

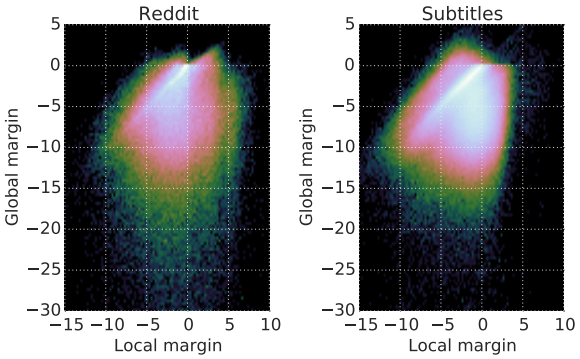


Figure 3: Local margin versus global margin for incorrectly predicted sequences. The color luminosity is proportional to frequency.

Margin discrepancy is higher for longer sequences In Figure 4 we show that this discrepancy is significantly more pronounced for longer sequences. In the figure we show the fraction of wrongly predicted sequences with a positive local margin. We find that as sequence length increases, we have more cases where the local margin is positive yet the global margin is negative. For example, for the Reddit dataset half of the wrongly predicted sequences have a positive local margin indicating that the training loss was low for these sequences even though they were not adequately separated.

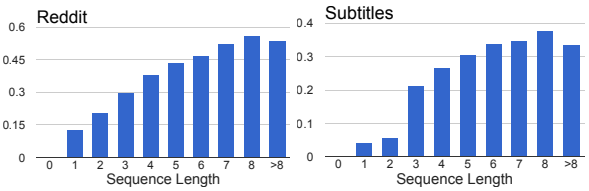


Figure 4: Fraction of incorrect predictions with positive local margin.

Increasing beam size drops accuracy for long sequences Next we show why this discrepancy leads to non-monotonic accuracies with increasing beam-size. As beam size increases, the predicted sequence has higher probability and the accuracy is expected to increase if the trained probabilities are well-calibrated. In Figure 5 we plot the number of correct predictions (on a log scale) against the length of the correct sequence for beam sizes of 1, 5, 10, and 15. For small sequence lengths, we indeed observe that increasing the beam size produces more accurate results. For longer sequences (length > 4) we observe a drop in accuracy with increasing the beam width beyond 1 for Reddit and beyond 5 for Subtitles.

Globally conditioned models are more accurate than ED models We next compare the ED model with our globally conditioned encoder-encoder (EE) model. In Figure 6 we show the recall@K values for K=1, 3 and 5 for the two datasets for increasing length of correct sequence. We find the EE model is largely better that the ED model. The most interesting difference is that for sequences of length greater than 8, the ED model has a recall@5 of zero for both datasets. In contrast, the EE model manages

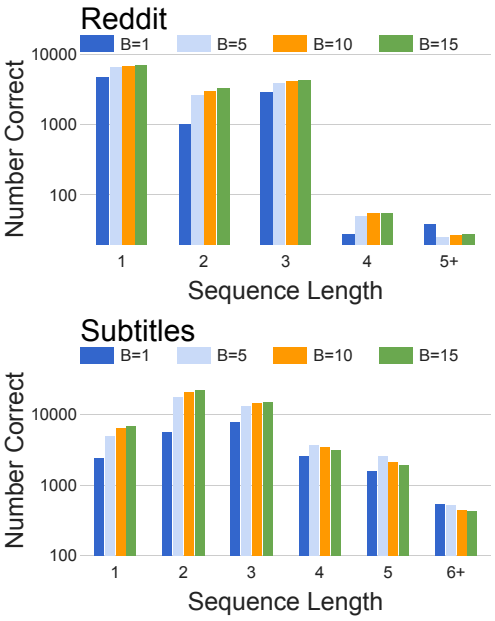


Figure 5: Effect of beam width on the number of correct predictions broken down by sequence length.

to achieve significant recall even at large sequence lengths.

Length normalization of ED models A common modification to the ED decoding procedure used to promote longer message is normalization of the prediction log-probability by its length raised to some power f (Cho et al., 2014; Graves, 2013). We experimented with two settings, $f = 0.5$ and 1.0. Our experiments show that while this indeed promotes longer sequences, it does so at the expense of reducing the accuracy on the shorter sequences.

5 Related Work

In this paper we showed that encoder-decoder models suffer from length bias and proposed a fix using global conditioning. Global conditioning has been proposed for other RNN-based sequence prediction tasks in (Yao et al., 2014) and (Andor et al., 2016). The RNN models that these work attempt to fix capture only a weak form of dependency among variables, for example they assume $\mathbf{x}$ is seen incrementally and only adjacent labels in $\mathbf{y}$ are directly dependent. As proved in (2016) these models are subject to label bias since they cannot represent a distribution that a globally conditioned model can. Thus, their fix for global dependency is using a CRFs. Such

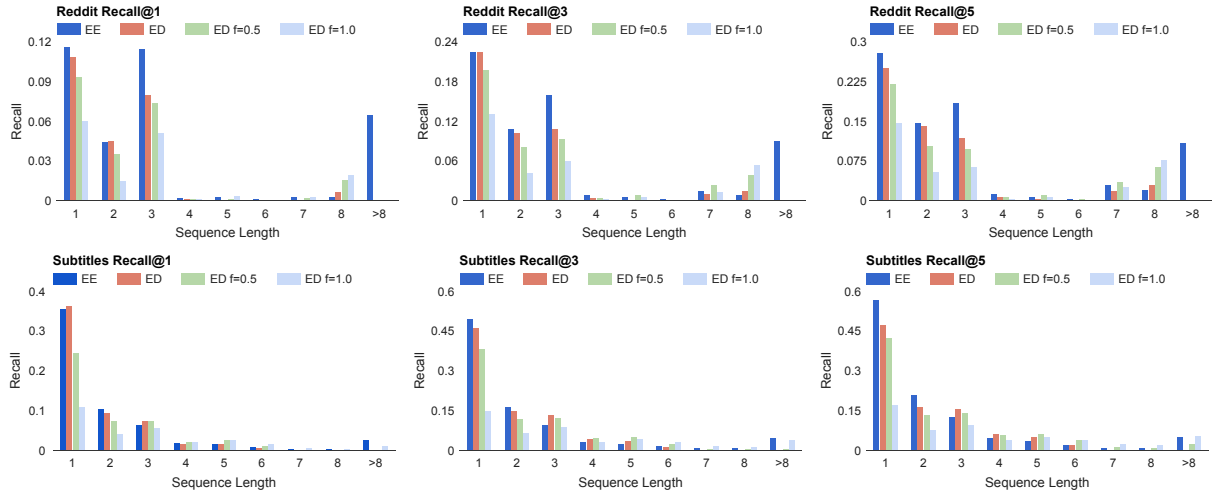


Figure 6: Comparing recall@1, 3, 5 for increasing length of correct sequence.

global conditioning will compromise a ED model which does not assume *any conditional independence* among variables. The label-bias proof of (2016) is not applicable to ED models because the proof rests on the entire input not being visible during output. Earlier illustrations of label bias of MeMMs in (Bottou, 1991; Lafferty et al., 2001) also require local observations. In contrast, the ED model transitions on the entire input and chain rule is an exact factorization of the distribution. Indeed one of the suggestions in (Bottou, 1991) to surmount label-bias is to use a fully connected network, which the ED model already does.

Our encoder-encoder network is reminiscent of the dual encoder network in (Lowe et al., 2015), also used for conversational response generation. A crucial difference is our use of importance sampling to correctly estimate the probability of a large set of candidate responses, which allows us to use the model as a standalone response generation system. Other differences include our model using separate sets of parameters for the two encoders, to reflect the assymetry of the prediction task. Lastly, we found it crucial for the model's quality to use multiple appropriately weighed negative examples for every positive example during training.

(Ranzato et al., 2016) also highlights limitations of the ED model and proposes to mix the ED loss with a sequence-level loss in a reinforcement learning framework under a carefully tuned schedule. Our method for global conditioning can capture sequence-level losses like BLEU score more easily, but may also benefit from a similar mixed loss function.

6 Conclusion

We have shown that encoder-decoder models in the regime of finite data and parameters suffer from a length-bias problem. We have proved that this arises due to the locally normalized models insufficiently separating correct sequences from incorrect ones, and have verified this empirically. We explained why this leads to the curious phenomenon of decreasing accuracy with increasing beam size for long sequences. Our proposed encoder-encoder architecture side steps this issue by operating in sequence probability space directly, yielding improved accuracy for longer sequences.

One weakness of our proposed architecture is that it cannot generate responses directly. An interesting future work is to explore if the ED model can be used to generate a candidate set of responses which are then re-ranked by our globally conditioned model. Another future area is to see if the techniques for making Bayesian networks discriminative can fix the length bias of encoder decoder networks (Peharz et al., 2013; Guo et al., 2012).

References

[Abadi et al.2015] Martín Abadi, Ashish Agarwal, Paul Barham, and Eugene Brevdo et al. 2015. TensorFlow:

Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Abadie et al.2014] J Pouget Abadie, D Bahdanau, B van Merrienboer, K Cho, and Y Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *CoRR*, abs/1409.1257.

[Andor et al.2016] D Andor, C Alberti, D Weis, A Severyn, A Presta, K Ganchev, S Petrov, and M Collins. 2016. Globally normalized transition-based neural network. *CoRR*, abs/1603.06042.

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Bengio et al.2015] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

[Bottou1991] L. Bottou. 1991. *Une approche theorique de l'apprentissage connexionniste: Applications a la recon'naissance de la parole.* Ph.D. thesis, Universitede Paris XI.

[Cho et al.2014] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

[Duchi et al.2011] John Duchi, Elan Hazad, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

[Guo et al.2012] Yuhong Guo, Dana F. Wilkinson, and Dale Schuurmans. 2012. Maximum margin bayesian networks. *CoRR*, abs/1207.1382.

[Guo et al.2016] R. Guo, S. Kumar, K. Choromanski, and D. Simcha. 2016. Quantization based fast inner product search. In *AISTATS*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Jean et al.2014] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007.

[Kannan et al.2016] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *KDD*.

[Khaitan2016] Pranav Khaitan. 2016. Chat smarter with allo. http://googleresearch.blogspot.com/2016/05/chat-smarter-with-allo.html, May.

[Lafferty et al.2001] John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

[Li et al.2015] J Li, M Galley, C Brockett, J Gao, and B Dolan. 2015. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055.

[Lison and Tiedemann2016] Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC 2016*.

[Lowe et al.2015] R Lowe, N Pow, I V Serban, and J Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructure multi-turn dialogue systems". In *SIGDial*.

[McCallum et al.2000] A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML*.

[Pascanu et al.2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

[Peharz et al.2013] Robert Peharz, Sebastian Tschiatschek, and Franz Pernkopf. 2013. The most generative maximum margin bayesian networks. In *ICML*.

[Ranzato et al.2016] M Ranzato, S Chopra, M Auli, and W Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

[Rosset et al.2003] S Rosset, J Zhu, and T Hastie. 2003. Margin maximizing loss functions. In *NIPS*.

[Sak et al.2014] Hasim Sak, Andrew Senior, and Francoise Beaufays. 2014. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In *INTERSPEECH 2014*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[Vinyals and Le2015] Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

[Vinyals et al.2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

[Yao et al.2014] K Yao, B Peng, G Zweig, D Yu, X Li, and F Gao. 2014. Recurrent conditional random field for language understanding. In *ICASSP*.

Does String-Based Neural MT Learn Source Syntax?

Xing Shi, Inkit Padhi, and Kevin Knight
Information Sciences Institute & Computer Science Department
University of Southern California
xingshi@isi.edu, ipadhi@usc.edu, knight@isi.edu

Abstract

We investigate whether a neural, encoder-decoder translation system learns syntactic information on the source side as a by-product of training. We propose two methods to detect whether the encoder has learned local and global source syntax. A fine-grained analysis of the syntactic structure learned by the encoder reveals which kinds of syntax are learned and which are missing.

1 Introduction

The sequence to sequence model (*seq2seq*) has been successfully applied to neural machine translation (NMT) (Sutskever et al., 2014; Cho et al., 2014) and can match or surpass MT state-of-art. Non-neural machine translation systems consist chiefly of phrase-based systems (Koehn et al., 2003) and syntax-based systems (Galley et al., 2004; Galley et al., 2006; DeNeefe et al., 2007; Liu et al., 2011; Cowan et al., 2006), the latter of which adds syntactic information to source side (tree-to-string), target side (string-to-tree) or both sides (tree-to-tree). As the seq2seq model first encodes the source sentence into a high-dimensional vector, then decodes into a target sentence, it is hard to understand and interpret what is going on inside such a procedure. Considering the evolution of non-neural translation systems, it is natural to ask:

1. Does the encoder learn syntactic information about the source sentence?
2. What kind of syntactic information is learned, and how much?
3. Is it useful to augment the encoder with additional syntactic information?

In this work, we focus on the first two questions and propose two methods:

- We create various syntactic labels of the source sentence and try to predict these syntactic labels with logistic regression, using the learned sentence encoding vectors (for sentence-level labels) or learned word-by-word hidden vectors (for word-level label). We find that the encoder captures both global and local syntactic information of the source sentence, and different information tends to be stored at different layers.
- We extract the whole constituency tree of source sentence from the NMT encoding vectors using a retrained linearized-tree decoder. A deep analysis on these parse trees indicates that much syntactic information is learned, while various types of syntactic information are still missing.

2 Example

As a simple example, we train an English-French NMT system on 110M tokens of bilingual data (English side). We then take 10K separate English sentences and label their *voice* as active or passive. We use the learned NMT encoder to convert these sentences into 10k corresponding 1000-dimension encoding vectors. We use 9000 sentences to train a logistic regression model to predict voice using the encoding cell states, and test on the other 1000 sentences. We achieve 92.8% accuracy (Table 2), far above the majority class baseline (82.8%). This means that in reducing the source sentence to a

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1526–1534,
Austin, Texas, November 1-5, 2016.

Model	Accuracy
Majority Class	82.8
English to French (E2F)	92.8
English to English (E2E)	82.7

Table 1: Voice (active/passive) prediction accuracy using the encoding vector of an NMT system. The majority class baseline always chooses active.

fixed-length vector, the NMT system has decided to store the voice of English sentences in an easily accessible way.

When we carry out the same experiment on an English-English (auto-encoder) system, we find that English voice information is no longer easily accessed from the encoding vector. We can only predict it with 82.7% accuracy, no better than chance. Thus, in learning to reproduce input English sentences, the seq2seq model decides to use the fixed-length encoding vector for other purposes.

3 Related work

Interpreting Recurrent Neural Networks. The most popular method to visualize high-dimensional vectors, such as word embeddings, is to project them into two-dimensional space using *t-SNE* (van der Maaten and Hinton, 2008). Very few works try to interpret recurrent neural networks in NLP. Karpathy et al. (2016) use a character-level LSTM language model as a test-bed and find several activation cells that track long-distance dependencies, such as line lengths and quotes. They also conduct an error analysis of the predictions. Li et al. (2016) explore the syntactic behavior of an RNN-based sentiment analyzer, including the compositionality of negation, intensification, and concessive clauses, by plotting a 60-dimensional heat map of hidden unit values. They also introduce a first-order derivative based method to measure each unit's contribution to the final decision.

Verifying syntactic/semantic properties. Several works try to build a good distributional representation of sentences or paragraph (Socher et al., 2013; Kalchbrenner et al., 2014; Kim, 2014; Zhao et al., 2015; Le and Mikolov, 2014; Kiros et al., 2015). They implicitly verify the claimed syntactic/semantic properties of learned representations by applying them to downstream classification tasks such as sentiment analysis, sentence classification, semantic relatedness, paraphrase detection, image-sentence ranking, question-type classification, etc.

Novel contributions of our work include:

- We locate a subset of activation cells that are responsible for certain syntactic labels. We explore the concentration and layer distribution of different syntactic labels.
- We extract whole parse trees from NMT encoding vectors in order to analyze syntactic properties directly and thoroughly.
- Our methods are suitable for large scale models. The models in this work are 2-layer 1000-dimensional LSTM seq2seq models.

4 Datasets and models

We train two NMT models, English-French (E2F) and English-German (E2G). To answer whether these translation models' encoders to learn store syntactic information, and how much, we employ two benchmark models:

- An upper-bound model, in which the encoder learns quite a lot of syntactic information. For the upper bound, we train a neural parser that learns to "translate" an English sentence to its linearized constitutional tree (E2P), following Vinyals et al. (2015).
- An lower-bound model, in which the encoder learns much less syntactic information. For the lower bound, we train two sentence auto-encoders: one translates an English sentence to itself (E2E), while the other translates a permuted English sentence to itself (PE2PE). We already had an indication above (Section 2) that a copying model does not necessarily need to remember a sentence's syntactic structure.

Figure 1 shows sample inputs and outputs of the E2E, PE2PE, E2F, E2G, and E2P models.

We use English-French and English-German data from WMT2014 (Bojar et al., 2014). We take 4M English sentences from the English-German data to train E2E and PE2PE. For the neural parser (E2P), we construct the training corpus following the recipe of Vinyals et al. (2015). We collect 162K training sentences from publicly available treebanks, including Sections 0-22 of the Wall Street Journal Penn Treebank (Marcus et al., 1993), Ontonotes version 5

Model	Target Language	Input vocabulary size	Output vocabulary size	Train/Dev/Test Corpora Sizes (sentence pairs)	BLEU
E2E	English	200K	40K	4M/3000/2737	89.11
PE2PE	Permuted English	200K	40K	4M/3000/2737	88.84
E2F	French	200K	40K	4M/6003/3003	24.59
E2G	German	200K	40K	4M/3000/2737	12.60
E2P	Linearized constituency tree	200K	121	8162K/1700/2416	n/a

Table 2: Model settings and test-set BLEU-n4r1 scores (Papineni et al., 2002).

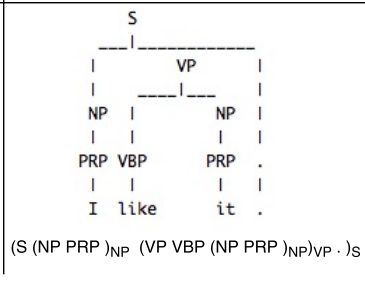
Model	Source	Target
E2E	I like it .	I like it .
PE2PE	it I . like	it I . like
E2F	I like it .	J'aime ça.
E2G	I like it .	Ich mag das.
E2P	I like it .	 (S (NP PRP)$_{NP}$ (VP VBP (NP PRP)$_{NP}$)$_{VP}$.)$_{S}$

Figure 1: Sample inputs and outputs of the E2E, PE2PE, E2F, E2G, and E2P models.

(Pradhan and Xue, 2009) and the English Web Treebank (Petrov and McDonald, 2012). In addition to these gold treebanks, we take 4M English sentences from English-German data and 4M English sentences from English-French data, and we parse these 8M sentences with the Charniak-Johnson parser[1] (Charniak and Johnson, 2005). We call these 8,162K pairs the CJ corpus. We use WSJ Section 22 as our development set and section 23 as the test set, where we obtain an F1-score of 89.6, competitive with the previously-published 90.5 (Table 4).

Model Architecture. For all experiments[2], we use a two-layer encoder-decoder with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). We use a minibatch of 128, a hidden state size of 1000, and a dropout rate of 0.2.

[1]The CJ parser is here https://github.com/BLLIP/bllip-parser and we used the pretrained model "WSJ+Gigaword-v2".

[2]We use the toolkit: https://github.com/isi-nlp/Zoph_RNN

Parser	WSJ 23 F1-score	# valid trees (out of 2416)
CJ Parser	92.1	2416
E2P	89.6	2362
(Vinyals et al., 2015)	90.5	unk

Table 3: Labeled F1-scores of different parsers on WSJ Section 23. The F1-score is calculated on valid trees only.

For auto-encoders and translation models, we train 8 epochs. The learning rate is initially set as 0.35 and starts to halve after 6 epochs. For E2P model, we train 15 epochs. The learning rate is initialized as 0.35 and starts to decay by 0.7 once the perplexity on a development set starts to increase. All parameters are re-scaled when the global norm is larger than 5. All models are non-attentional, because we want the encoding vector to summarize the whole source sentence. Table 4 shows the settings of each model and reports the BLEU scores.

5 Syntactic Label Prediction

5.1 Experimental Setup

In this section, we test whether different seq2seq systems learn to encode syntactic information about the source (English) sentence.

With 1000 hidden states, it is impractical to investigate each unit one by one or draw a heat map of the whole vector. Instead, we use the hidden states to predict syntactic labels of source sentences via logistic regression. For multi-class prediction, we use a one-vs-rest mechanism. Furthermore, to identify a subset of units responsible for certain syntactic labels, we use the recursive feature elimination (RFE) strategy: the logistic regression is first trained using

Label	Train	Test	Number of classes	Most frequent label
Voice	9000	1000	2	Active
Tense	9000	1000	2	Non-past
TSS	9000	1000	20	NP-VP
POS	87366	9317	45	NN
SPC	81292	8706	24	NP

Table 4: Corpus statistics for five syntactic labels.

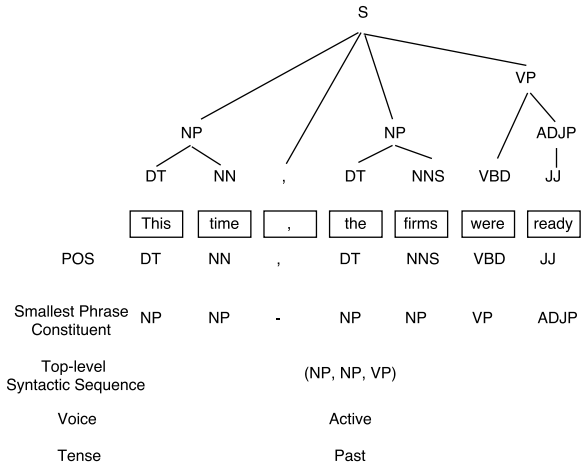


Figure 2: The five syntactic labels for sentence "This time , the firms were ready".

all 1000 hidden states, after which we recursively prune those units whose weights' absolute values are smallest.

We extract three sentence-level syntactic labels:

1. Voice: active or passive.
2. Tense: past or non-past.
3. TSS: Top level syntactic sequence of the constituent tree. We use the most frequent 19 sequences ("NP-VP", "PP-NP-VP", etc.) and label the remainder as "Other".

and two word-level syntactic labels:

1. POS: Part-of-speech tags for each word.
2. SPC: The smallest phrase constituent that above each word.

Both voice and tense labels are generated using rule-based systems based on the constituent tree of the sentence.

Figure 2 provides examples of our five syntactic labels. When predicting these syntactic labels using corresponding cell states, we split the dataset into training and test sets. Table 4 shows statistics of each labels.

For a source sentence s,

$$s = [w_1, ..., w_i, ..., w_n]$$

the two-layer encoder will generate an array of cell vectors c during encoding,

$$c = [(c_{1,0}, c_{1,1}), ..., (c_{i,0}, c_{i,1}), ..., (c_{n,0}, c_{n,1})]$$

We extract a sentence-level syntactic label L_s, and predict it using the encoding cell states that will be fed into the decoder:

$$L_s = g(c_{n,0}) \text{ or } L_s = g(c_{n,1})$$

where $g(\cdot)$ is the logistic regression.

Similarly, for extracting word-level syntactic labels:

$$L_w = [L_{w1}, ..., L_{wi}, ..., L_{wn}]$$

we predict each label L_{wi} using the cell states immediately after encoding the word w_i:

$$L_{wi} = g(c_{i,0}) \text{ or } L_{Wi} = g(c_{i,1})$$

5.2 Result Analysis

Test-set prediction accuracy is shown in Figure 3. For voice and tense, the prediction accuracy of two auto-encoders is almost same as the accuracy of majority class, indicating that their encoders do not learn to record this information. By contrast, both the neural parser and the NMT systems achieve approximately 95% accuracy. When predicting the top-level syntactic sequence (TSS) of the whole sentence, the Part-of-Speech tags (POS), and smallest phrase constituent (SPC) for each word, all five models achieve an accuracy higher than that of majority class, but there is still a large gap between the accuracy of NMT systems and auto-encoders. These observations indicate that the NMT encoder learns significant sentence-level syntactic information—it can distinguish voice and tense of the source sentence, and it knows the sentence's structure to some extent. At the word level, the NMT's encoder also tends to cluster together the words that have similar POS and SPC labels.

Different syntactic information tends to be stored at different layers in the NMT models. For word-level syntactic labels, POS and SPC, the accuracy of the lower layer's cell states ($C0$) is higher than that of the upper level ($C1$). For the sentence-level

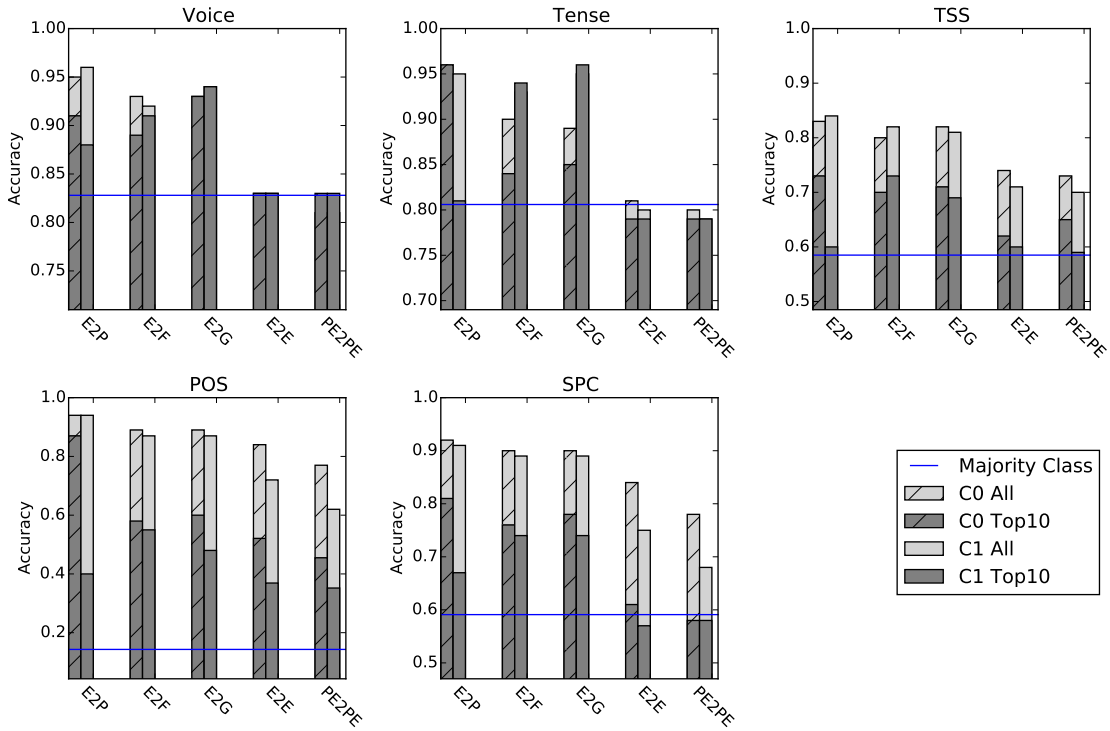


Figure 3: Prediction accuracy of five syntactic labels on test. Each syntactic label is predicted using both the lower-layer cell states (C0) and higher-layer cell states (C1). For each cell state, we predict each syntactic label using all 1000 units (All), as well as the top 10 units (Top10) selected by recursive feature elimination. The horizontal blue line is the majority class accuracy.

labels, especially tense, the accuracy of $C1$ is larger than $C0$. This suggests that the local features are somehow preserved in the lower layer whereas more global, abstract information tends to be stored in the upper layer.

For two-classes labels, such as voice and tense, the accuracy gap between all units and top-10 units is small. For other labels, where we use a one-versus-rest strategy, the gap between all units and top-10 units is large. However, when predicting POS, the gap of neural parser (E2P) on the lower layer ($C0$) is much smaller. This comparison indicates that a small subset of units explicitly takes charge of POS tags in the neural parser, whereas for NMT, the POS info is more distributed and implicit.

There are no large differences between encoders of E2F and E2G regarding syntactic information.

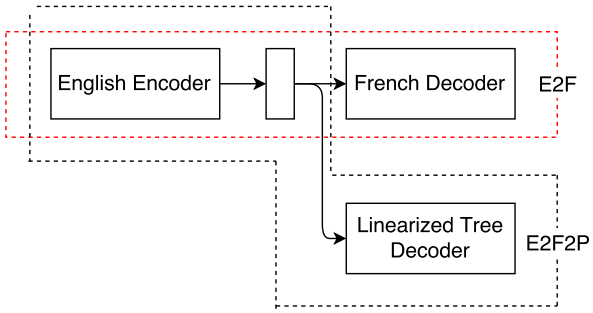


Figure 4: E2F and E2F2P share the same English encoder. When training E2F2P, we only update the parameters of linearized tree decoder, keeping the English encoder's parameters fixed.

6 Extract Syntactic Trees from Encoder

6.1 Experimental Setup

We now turn to whether NMT systems capture deeper syntactic structure as a by-product of learning to translate from English to another language. We do this by predicting full parse trees from the information stored in encoding vectors. Since this is a structured prediction problem, we can no longer use logistic regression. Instead, we extract a constituency parse tree from the encoding vector of a model E2X by using a new neural parser E2X2P with the following steps:

1. Take the E2X encoder as the encoder of the new model E2X2P.
2. Initialize the E2X2P decoder parameters with a uniform distribution.
3. Fine-tune the E2X2P decoder (while keeping its encoder parameters fixed), using the CJ corpus, the same corpus used to train $E2P$.

Figure 4 shows how we construct model E2F2P from model E2F. For fine-tuning, we use the same dropout rate and learning rate updating configuration for E2P as described in Section 4.

6.2 Evaluation

We train four new neural parsers using the encoders of the two auto-encoders and the two NMT models respectively. We use three tools to evaluate and analyze:

1. The EVALB tool[3] to calculate the labeled bracketing F1-score.
2. The zxx package[4] to calculate Tree edit distance (TED) (Zhang and Shasha, 1989).
3. The Berkeley Parser Analyser[5] (Kummerfeld et al., 2012) to analyze parsing error types.

The linearized parse trees generated by these neural parsers are not always well-formed. They can be split into the following categories:

- Malformed trees: The linearized sequence can not be converted back into a tree, due to missing or mismatched brackets.
- Well-formed trees: The sequence can be converted back into a tree. Tree edit distance can be calculated on this category.
  - Wrong length trees: The number of tree leaves does not match the number of source-sentence tokens.
  - Correct length trees: The number of tree leaves *does* match the number of source-sentence tokens.

Before we move to results, we emphasize the following points:

First, compared to the linear classifier used in Section 5, the retrained decoder for predicting a linearized parse tree is a highly non-linear method. The syntactic prediction/parsing performance will increase due to such non-linearity. Thus, we do not make conclusions based only on absolute performance values, but also on a comparison against the designed baseline models. An improvement over the lower bound models indicates that the encoder learns syntactic information, whereas a decline from the upper bound model shows that the encoder loses certain syntactic information.

Second, the NMT's encoder maps a plain English sentence into a high-dimensional vector, and our goal is to test whether the projected vectors form a more syntactically-related manifold in the high-dimensional space. In practice, one could also predict parse structure for the E2E in two steps: (1) use E2E's decoder to recover the original English sentence, and (2) parse that sentence with the CJ parser. But in this way, the manifold structure in the high-dimensional space is destroyed during the mapping.

6.2.1 Result Analysis

Table 5 reports perplexity on training and development sets, the labeled F1-score on WSJ Section 23, and the Tree Edit Distance (TED) of various systems.

Tree Edit Distance (TED) calculates the minimum-cost sequence of node edit operations (delete, insert, rename) between a gold tree and a test tree. When decoding with beam size 10, the four new neural parsers can generate well-formed trees for almost all the 2416 sentences in the WSJ section 23. This makes TED a robust metric to evaluate the overall performance of each parser. Table 5 reports the average TED per sentence. Trees extracted from E2E and PE2PE encoding vectors (via models E2E2P and PE2PE2P, respectively) get TED above 30, whereas the NMT systems get

[3] http://nlp.cs.nyu.edu/evalb/

[4] https://github.com/timtadh/zhang-shasha

[5] https://github.com/jkkummerfeld/berkeley-parser-analyser

Model	Perplexity on Train	Perplexity on WSJ 22	Labeled F1 on WSJ23	# EVALB-trees (out of 2416)	Average TED per sentence	# Well-formed trees (out of 2416)
PE2PE2P	1.83	1.92	46.64	818	34.43	2416
E2E2P	1.69	1.77	59.35	796	31.25	2416
E2G2P	1.39	1.41	80.34	974	17.11	2340
E2F2P	1.36	1.38	79.27	1093	17.77	2415
E2P	1.11	1.18	89.61	2362	11.50	2415

Table 5: Perplexity, labeled F1-score, and Tree Edit Distance (TED) of various systems. Labeled F1-scores are calculated on EVALB-trees only. Tree edit distances are calculated on the well-formed trees only. EVALB-trees are those whose number of leaves match the number of words in the source sentence, and are otherwise accepted by standard Treebank evaluation software.

approximately 17 TED.

Among the well-formed trees, around half have a mismatch between number of leaves and number of tokens in the source sentence. The labeled F1-score is reported over the rest of the sentences only. Though biased, this still reflects the overall performance: we achieve around 80 F1 with NMT encoding vectors, much higher than with the E2E and PE2PE encoding vectors (below 60).

6.2.2 Fine-grained Analysis

Besides answering whether the NMT encoders learn syntactic information, it is interesting to know what kind of syntactic information is extracted and what is not.

As Table 5 shows, different parsers generate different numbers of trees that are acceptable to Treebank evaluation software ("EVALB-trees"), having the correct number of leaves and so forth. We select the intersection set of different models' EVALB-trees. We get a total of 569 shared EVALB-trees. The average length of the corresponding sentence is 12.54 and the longest sentence has 40 tokens. The average length of all 2416 sentences in WSJ section 23 is 23.46, and the longest is 67. As we do not apply an attention model for these neural parsers, it is difficult to handle longer sentences. While the intersection set may be biased, it allows us to explore how different encoders decide to capture syntax on short sentences.

Table 6 shows the labeled F1-scores and Part-of-Speech tagging accuracy on the intersection set. The NMT encoder extraction achieves around 86 percent tagging accuracy, far beyond that of the auto-encoder based parser.

Model	Labeled F1	POS Tagging Accuracy
PE2PE2P	58.67	54.32
E2E2P	70.91	68.03
E2G2P	85.36	85.30
E2F2P	86.62	87.09
E2P	93.76	96.00

Table 6: Labeled F1-scores and POS tagging accuracy on the intersection set of EVALB-trees of different parsers. There are 569 trees in the intersection, and the average length of corresponding English sentence is 12.54.

Besides the tagging accuracy, we also utilize the Berkeley Parser Analyzer (Kummerfeld et al., 2012) to gain a more linguistic understanding of predicted parses. Like TED, the Berkeley Parser Analyzer is based on tree transformation. It repairs the parse tree via a sequence of sub-tree movements, node insertions and deletions. During this process, multiple bracket errors are fixed, and it associates this group of node errors with a linguistically meaningful error type.

The first column of Figure 5 shows the average number of bracket errors per sentence for model E2P on the intersection set. For other models, we report the ratio of each model to model E2P. Kummerfeld et al. (2013) and Kummerfeld et al. (2012) give descriptions of different error types. The NMT-based predicted parses introduce around twice the bracketing errors for the first 10 error types, whereas for "Sense Confusion", they bring more than 16 times bracket errors. "Sense confusion" is the case where the head word of a phrase receives the wrong POS,

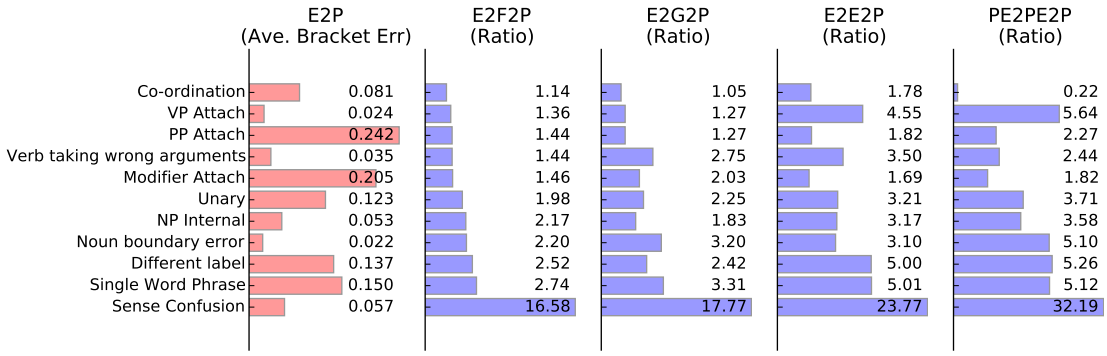


Figure 5: For model $E2P$ (the red bar), we show the average number of bracket errors per sentence due to the top 11 error types. For other models, we show the ratio of each model's average number of bracket errors to that of model $E2P$. Errors analyzed on the intersection set. The table is sorted based on the ratios of the $E2F2P$ model.

resulting in an attachment error. Figure 6 shows an example.

Even though we can predict 86 percent of parts-of-speech correctly from NMT encoding vectors, the other 14 percent introduce quite a few attachment errors. NMT sentence vectors encode a lot of syntax, but they still cannot grasp these subtle details.

7 Conclusion

We investigate whether NMT systems learn source-language syntax as a by-product of training on string pairs. We find that both local and global syntactic information about source sentences is captured by the encoder. Different types of syntax is stored in different layers, with different concentration degrees. We also carry out a fine-grained analysis of the constituency trees extracted from the encoder, highlighting what syntactic information is still missing.

Acknowledgments

This work was supported by ARL/ARO (W911NF-10-1-0533) and DARPA (HR0011-15-C-0115).

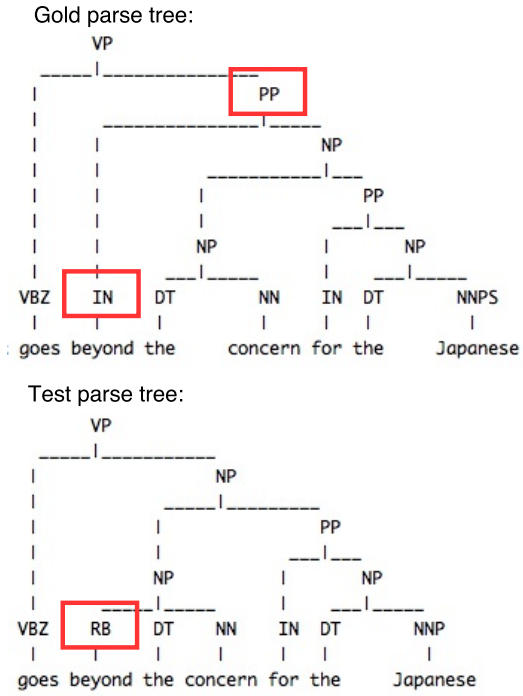


Figure 6: Example of Sense Confusion. The POS tag for word "beyond" is predicted as "RB" instead of "IN", resulting in a missing prepositional phrase.

References

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Matous Machacek, Christof Monz, Pavel Pecina, Matt Post, Herv Saint-Amand, Radu Soricut, and Lucia Specia, editors. 2014. *Proc. Ninth Workshop on Statistical Machine Translation*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. EMNLP*.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. EMNLP-CoNLL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What ' s in a translation rule ? *Information Sciences*, 2004.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL*.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. ACL*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2016. Visualizing and understanding recurrent networks. In *Proc. ICLR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. EMNLP*.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL*.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street Corral: An empirical investigation of error types in parser output. In *Proc. EMNLP-CoNLL*.

Jonathan K. Kummerfeld, Daniel Tse, James R Curran, and Dan Klein. 2013. An empirical examination of challenges in Chinese parsing. In *Proc. ACL*.

Qv Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. ICML*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proc. NAACL*.

Yang Liu, Qun Liu, and Yajuan Lü. 2011. Adjoining tree-to-string translation. In *Proc. ACL*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on Parsing the Web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Sameer S Pradhan and Nianwen Xue. 2009. Ontonotes: the 90% solution. In *Proc. NAACL*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. NIPS*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6).

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proc. IJCAI*.

Exploiting Source-side Monolingual Data in Neural Machine Translation

Jiajun Zhang[†] and **Chengqing Zong**[†‡]
[†]University of Chinese Academy of Sciences, Beijing, China
National Laboratory of Pattern Recognition, CASIA, Beijing, China
[‡]CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China
{jjzhang,cqzong}@nlpr.ia.ac.cn

Abstract

Neural Machine Translation (NMT) based on the encoder-decoder architecture has recently become a new paradigm. Researchers have proven that the target-side monolingual data can greatly enhance the decoder model of NMT. However, the source-side monolingual data is not fully explored although it should be useful to strengthen the encoder model of NMT, especially when the parallel corpus is far from sufficient. In this paper, we propose two approaches to make full use of the source-side monolingual data in NMT. The first approach employs the self-learning algorithm to generate the synthetic large-scale parallel data for NMT training. The second approach applies the multi-task learning framework using two NMTs to predict the translation and the reordered source-side monolingual sentences simultaneously. The extensive experiments demonstrate that the proposed methods obtain significant improvements over the strong attention-based NMT.

1 Introduction

Neural Machine Translation (NMT) following the encoder-decoder architecture proposed by (Kalchbrenner and Blunsom, 2013; Cho et al., 2014) has become the novel paradigm and obtained state-of-the-art translation quality for several language pairs, such as English-to-French and English-to-German (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015b; Sennrich et al., 2015). This end-to-end NMT typically consists of two recurrent neural networks. The encoder network maps the source sentence of variable length into the context vector representation; and the decoder network generates the target translation word by word starting from the context vector.

Currently, most NMT methods utilize only the sentence aligned parallel corpus for model training, which limits the capacity of the model. Recently, inspired by the successful application of target monolingual data in conventional statistical machine translation (SMT) (Koehn et al., 2007; Chiang, 2007), Gulcehre et al. (2015) and Sennrich et al. (2015) attempt to enhance the decoder network model of NMT by incorporating the target-side monolingual data so as to boost the translation fluency. They report promising improvements by using the target-side monolingual data. In contrast, the source-side monolingual data is not fully explored. Luong et al. (2015a) adopt a simple autoencoder or skip-thought method (Kiros et al., 2015) to exploit the source-side monolingual data, but no significant BLEU gains are reported. Note that, in parallel to our efforts, Cheng et al. (2016b) have explored the usage of both source and target monolingual data using a similar semi-supervised reconstruction method, in which two NMTs are employed. One translates the source-side monolingual data into target translations, and the other reconstructs the source-side monolingual data from the target translations.

In this work, we investigate the usage of the source-side large-scale monolingual data in NMT and aim at greatly enhancing its encoder network so that we can obtain high quality context vector representations. To achieve this goal, we propose two

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1535–1545,
Austin, Texas, November 1-5, 2016.

approaches. Inspired by (Ueffing et al., 2007; Wu et al., 2008) handling source-side monolingual corpus in SMT and (Sennrich et al., 2015) exploiting target-side monolingual data in NMT, the first approach adopts the self-learning algorithm to generate adequate synthetic parallel data for NMT training. In this method, we first build the baseline machine translation system with the available aligned sentence pairs, and then obtain more synthetic parallel data by translating the source-side monolingual sentences with the baseline system.

The proposed second approach applies the multi-task learning framework to predict the target translation and the reordered source-side sentences at the same time. The main idea behind is that we build two NMTs: one is trained on the aligned sentence pairs to predict the target sentence from the source sentence, while the other is trained on the source-side monolingual corpus to predict the reorderd source sentence from original source sentences[1]. It should be noted that the two NMTs share the same encoder network so that they can help each other to strengthen the encoder model.

In this paper, we make the following contributions:

- To fully investigate the source-side monolingual data in NMT, we propose and compare two methods. One attempts to enhance the encoder network of NMT by producing rich synthetic parallel corpus using a self-learning algorithm, and the other tries to perform machine translation and source sentence reordering simultaneously with a multi-task learning architecture.

- The extensive experiments on Chinese-to-English translation show that our proposed methods significantly outperform the strong NMT baseline augmented with the attention mechanism. We also find that the usage of the source-side monolingual data in NMT is more effective than that in SMT. Furthermore, we find that more monolingual data does not always improve the translation quality and only relevant monolingual data helps.

[1]We reorder all the source-side monolingual sentences so as to make them close to target language in word order.

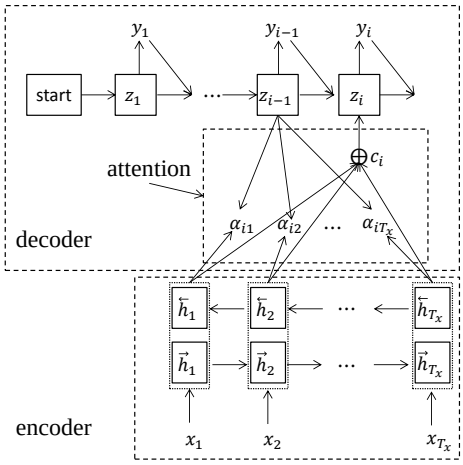


Figure 1: The encoder-decoder NMT with attention.

2 Neural Machine Translation

Our approach on using source-side monolingual corpora can be applied in any neural machine translation as long as it employs the encoder-decoder framework. Without loss of generality, we use the attention-based NMT proposed by (Bahdanau et al., 2014), which utilizes recurrent neural networks for both encoder and decoder as illustrated in Fig. 1.

The encoder-decoder NMT first encodes the source sentence $X = (x_1, x_2, \cdots, x_{T_x})$ into a sequence of context vectors $C = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{T_x})$ whose size varies with respect to the source sentence length. Then, the encoder-decoder NMT decodes from the context vectors C and generates target translation $Y = (y_1, y_2, \cdots, y_{T_y})$ one word each time by maximizing the probability of $p(y_i|y_{<i}, C)$. Note that x_j (y_i) is word embedding corresponding to the j_{th} (i_{th}) word in the source (target) sentence. Next, we briefly review the encoder introducing how to obtain C and the decoder addressing how to calculate $p(y_i|y_{<i}, C)$.

Encoder: The context vectors C are generated by the encoder using a pair of recurrent neural networks (RNN) which consists of a forward RNN and a backward RNN. The forward RNN operates left-to-right over the source sentence from the first word, resulting in the forward context vectors $C_f = (\overrightarrow{h}_1, \overrightarrow{h}_2, \cdots, \overrightarrow{h}_{T_x})$, in which

$$\overrightarrow{h}_j = RNN(\overrightarrow{h}_{j-1}, x_j) \quad (1)$$

$\overleftarrow{h}_j$ can be calculated similarly.

RNN can be a Gated Recurrent Unit (GRU) (Cho et al., 2014) or a Long Short-Term Memory Unit (LSTM) (Hochreiter and Schmidhuber, 1997). At each position j of the source sentence, the context vector $\mathbf{h}_j$ is defined as the concatenation of the forward and backward context vectors.

Decoder: The conditional probability $p(y_i|y_{<i}, C)$ is computed in different ways according to the choice of the context C at time i. In (Cho et al., 2014), the authors choose $C = \mathbf{h}_{T_x}$, while Bahdanau et al. (2014) use different context c_i at different time step and the conditional probability will become:

$$p(y_i|y_{<i}, C) = p(y_i|y_{<i}, c_i) = g(y_{i-1}, z_i, c_i) \quad (2)$$

where z_i is the i_{th} hidden state of the decoder and is calculated conditioning on the previous hidden state z_{i-1}, previous output y_{i-1} and and the source context vector c_i at time i:

$$z_i = RNN(z_{i-1}, y_{i-1}, c_i) \quad (3)$$

In attention-based NMT, c_i is computed as the weighted sum of the source-side context vectors, just as illustrated in the top half of Fig. 1.

All the parameters of the encoder-decoder NMT are optimized to maximize the following conditional log-likelihood of the ***sentence aligned bilingual data***:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{T_y} logp(y_i^{(n)}|y_{<i}^{(n)}, X^{(n)}, \theta) \quad (4)$$

3 Incorporating Source-side Monolingual Data in NMT

We can see from the above objective function that all the network parameters are only optimized on the sentence aligned parallel corpus. It is well known that more related data of high quality leads to better and more robust network models. However, bilingual data is scarce in many languages (or domains). It becomes a key issue how to improve the encoder and decoder networks using other data besides the parallel sentence pairs. Gulcehre et al. (2015) and Sennrich et al. (2015) have tried to fine-tune the decoder neural network with target-side large-scale monolingual data and they report remarkable performance improvement with the enhanced decoder. In contrast, we believe that the encoder part of NMT can also be greatly strengthened with the source-side monolingual data.

To investigate fully the source-side monolingual data in improving the encoder network of NMT, we propose two approaches: the first one employs the self-learning algorithm to provide synthetic parallel data in which the target part is obtained through automatically translating the source-side monolingual data, which we refer to as *self-learning method*. The second one applies the multi-task learning framework that consists of two NMTs sharing the same encoder network to simultaneously train one NMT model on bilingual data and the other sentence reordering NMT model[2] on source-side monolingual data, which we refer to as *sentence reordering method*.

3.1 Self-learning Method

Given the sentence aligned bitext $\mathcal{D}_b = \{(X_b^{(n)}, Y_b^{(n)})\}_{n=1}^{N}$ in which N is not big enough, we have the source-side large-scale monolingual data $\mathcal{D}_{sm} = \{X_{sm}^{m}\}_{m=1}^{M}$ which is related to the bitext and $M \gg N$.

Our goal is to generate much more bilingual data using $\mathcal{D}_b$ and $\mathcal{D}_{sm}$. From the view of machine learning, we are equipped with some labelled data $\mathcal{D}_b$ and plenty of unlabelled data $\mathcal{D}_{sm}$, and we aim to obtain more labelled data for training better models. Self-learning is a simple but effective algorithm to tackle this issue. It first establishes a baseline with labelled data and then adopts the baseline to predict the labels of the unlabelled data. Finally, the unlabelled data together with the predicted labels become new labelled data.

In our scenario, the self-learning algorithm perform the following three steps . First, a baseline machine translation (MT) system (can use any translation model, SMT or NMT) is built with the given bilingual data $\mathcal{D}_b$. Second, the baseline MT sys-

[2]NMT is essentially a sequence-to-sequence prediction model. In most cases, the input sequence is different from the output sequence. In the sentence reordering NMT, we require that output sequence to be the reordered input sentences which are close to English word order.

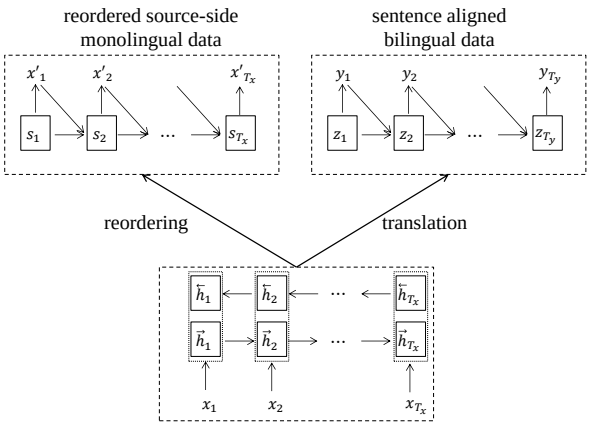


Figure 2: Multi-task learning framework to use source-side monolingual data in NMT, which includes a translation model and a sentence reordering model.

tem automatically translates the source-side monolingual sentences $\mathcal{D}_{sm}$ into target translations $\mathcal{D}_{tt} = \{(Y_{tt}^m)\}_{m=1}^M$, and further pairs $\mathcal{D}_{sm}$ with $\mathcal{D}_{tt}$ resulting in the synthetic parallel corpus $\mathcal{D}_{syn} = \{(X_{sm}^m, Y_{tt}^m)\}_{m=1}^M$. Third, the synthetic parallel corpus $\mathcal{D}_{syn}$ plus the original bitext $\mathcal{D}_b$ are combined together to train the new NMT model.

In principle, we can apply any MT system as the baseline to generate the synthetic bilingual data. In accordance with the translation model we focus on in this work, we employ NMT as the baseline MT system. Note that the synthetic target parts may negatively influence the decoder model of NMT. To address this problem, we can distinguish original bitext from the synthetic bilingual sentences during NMT training by freezing the parameters of the decoder network for the synthetic data.

It is worthy to discuss why self-learning algorithm can improve the encoder model of NMT. Even though we require $\mathcal{D}_{sm}$ to share the same source language vocabulary as $\mathcal{D}_b$ and no new word translations can be generated, the source-side monolingual data provides much more permutations of words in the vocabulary. Our RNN encoder network model will be optimized to well explain all of the word permutations. Thus, the encoder model of NMT can be enhanced for better generalization.

3.2 Sentence Reordering Method

The self-learning algorithm needs to translate first the large-scale source-side monolingual data. A natural question arises that whether can we improve the encoder model of NMT using just source-side monolingual corpora rather than the synthetic parallel data. Luong et al. (2015a) attempt to leverage source-side monolingual data in NMT using a simple autoencoder and skip-thought vectors. However, no promising results are reported. We believe that the reason lies in two aspects: 1) the large-scale monolingual data is not carefully selected; and 2) the adopted model is relatively simple. In this work, we propose to apply the multi-task learning method which designs a parameter sharing neural network framework to perform two tasks: machine translation and source sentence reordering. Fig.2 illustrates the overview of our framework for source-side monolingual data usage.

As shown in Fig. 2, our framework consists of two neural networks that shares the same encoder model but employs two different decoder models for machine translation and sentence reordering respectively. For the machine translation task trained on the sentence aligned parallel data $\mathcal{D}_b$, the network parameters are optimized to maximize the conditional probability of the target sentence $Y_b^{(n)}$ given a source sentence $X_b^{(n)}$, namely argmax$p(Y_b^{(n)}|X_b^{(n)})$.

As for the sentence reordering task trained on source-side monolingual data $\mathcal{D}_{sm}$, we regard it as a special machine translation task in which the target output is just the reordered source sentence, $Y_{sm}^{(m)} = X_{sm}^{'(m)}$. $X_{sm}^{'(m)}$ is obtained from $X_{sm}^{(m)}$ by using the pre-ordering rules proposed by (Wang et al., 2007), which can permutate the words of the source sentence so as to approximate the target language word order[3]. In this way, the sentence reordering NMT is more powerful than an autoencoder. Using the NMT paradigm, the shared encoder network is leveraged to learn the deep representation $C_{sm}^{(n)}$ of the source sentence $X_{sm}^{(n)}$, and the decoder network is employed to predict the reordered source sentence from the deep representation $C_{sm}^{(n)}$ (here $X_{sm}^{(n)} \in \mathcal{D}_{sm}$) by maximizing $p(X_{sm}^{'(n)}|X_{sm}^{(n)})$. Note that the above two

[3]The pre-ordering rules are obtained from the parsed source trees which heavily depend on the accuracy and efficiency of the parser. In fact, it takes us lots of time (even longer than synthetic parallel data generation) to parse all the source-side monolingual data. In the future, we attempt to design a more efficient pre-ordering method relying only on the bilingual training data.

tasks share the same encoder model to obtain the encoding of the source sentences. Accordingly, the overall objective function of this multi-task learning is the summation of log probabilities of machine translation and sentence reordering:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{T_y} logp(y_i^{(n)}|y_{<i}^{(n)}, X^{(n)}, \theta) + \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{T_X} logp(X_i^{'(m)}|X_{<i}^{'(m)}, X^{(m)}, \theta) \quad (5)$$

where $(\theta = \theta_{enc}, \theta_{dec_T}, \theta_{dec_R})$. θ_{enc} is the parameter collection of source language encoder network, θdec_T denotes the parameter set of the decoder network for translation, and θ_{dec_R} represents the parameters of the decoder network for sentence reordering.

Intuitively, the sentence reordering task is easier than the translation task. Furthermore, in this paper, we pay much more attention on the translation task compared to the sentence reordering task. Considering these, we distinguish these two tasks during the parameter optimization process. It is performed using an alternate iteration strategy. For each iteration, we first optimize the encoder-decoder network parameters in the reordering task for one epoch. The learnt encoder network parameters are employed to initialize the encoder model for the translation task. Then, we learn the encoder-decoder network parameters in the translation task for several epochs[4]. The new encoder parameters are then used to initialize the encoder model for the reordering task. We continue the iteration until the constraint (e.g. iteration number or no parameter change) is satisfied. The weakness is that this method is less efficient than the self-learning approach.

4 Experimental Settings

In this section we describe the data set used in our experiments, data preprocessing, the training and evaluation details, and all the translation methods we compare in experiments.

[4]We rune four epochs for the translation task in each iteration.

4.1 Dataset

We perform two tasks on Chinese-to-English translation: one for small data set and the other for large-scale data set. Our small training data includes 0.63M sentence pairs (after data cleaning) extracted from LDC corpora[5]. The large-scale data set contains about 2.1M sentence pairs including the small training data. For validation, we choose NIST 2003 (MT03) dataset. For testing, we use NIST 2004 (MT04), NIST 2005 (MT05) and NIST 2006 (MT06) datasets. As for the source-side monolingual data, we collect about 20M Chinese sentences from LDC and we retain the sentences in which more than 50% words should appear in the source-side portion of the bilingual training data, resulting in 6.5M monolingual sentences for small training data set (12M for large-scale training data set) ordered by the word hit rate.

4.2 Data Preprocessing

We apply word-level translation in experiments. The Chinese sentences are word segmented using Stanford Word Segmenter[6]. To pre-order the Chinese sentences using the syntax-based reordering method proposed by (Wang et al., 2007), we utilize the Berkeley parser (Petrov et al., 2006). The English sentences are tokenized using the tokenizer script from the Moses decoder[7]. To speed up the training procedure, we clean the training data and remove all the sentences of length over 50 words. We limit the vocabulary in both Chinese and English to the most 40K words and all the out-of-vocabulary words are replaced with *UNK*.

4.3 Training and Evaluation Details

Each NMT model is trained on GPU K40 using stochastic gradient decent algorithm AdaGrad (Duchi et al., 2011). We use mini batch size of 32. The word embedding dimension of source and target language is 500 and the size of hidden layer is set to 1024. The training time for each model ranges from 5 days to 10 days for small training data set and ranges from 8 days to 15 days for large training data

[5]LDC2000T50, LDC2002L27, LDC2002T01, LDC2002E18, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004T07.

[6]http://nlp.stanford.edu/software/segmenter.shtml

[7]http://www.statmt.org/moses/

Method	MT03	MT04	MT05	MT06
Moses	30.30	31.04	28.19	30.04
RNNSearch	28.38	30.85	26.78	29.27
RNNSearch-Mono-SL (25%)	29.65	31.92	28.65	29.86
RNNSearch-Mono-SL (50%)	**32.43**	**33.16**	**30.43**	**32.35**
RNNSearch-Mono-SL (75%)	30.24	31.18	29.33	28.82
RNNSearch-Mono-SL (100%)	29.97	30.78	26.45	28.06
RNNSearch-Mono-MTL (25%)	31.68	32.51	29.8	31.29
RNNSearch-Mono-MTL (50%)	**33.38**	**34.30**	**31.57**	**33.40**
RNNSearch-Mono-MTL (75%)	31.69	32.83	28.17	30.26
RNNSearch-Mono-MTL (100%)	30.31	30.62	27.23	28.85
RNNSearch-Mono-Autoencoder (50%)	31.55	32.07	28.19	30.85
RNNSearch-Mono-Autoencoder (100%)	27.81	30.32	25.84	27.73

Table 1: Translation results (BLEU score) for different translation methods. For our methods exploring the source-side monolingual data, we investigate the performance change as we choose different scales of monolingual data (e.g. from top 25% to 100% according to the word coverage of the monolingual sentence in source language vocabulary of bilingual training corpus).

set[8]. We use case-insensitive 4-gram BLEU score as the evaluation metric (Papineni et al., 2002).

4.4 Translation Methods

In the experiments, we compare our method with conventional SMT model and a strong NMT model. We list all the translation methods as follows:

- **Moses**: It is the state-of-the-art phrase-based SMT system (Koehn et al., 2007). We use its default configuration and train a 4-gram language model on the target portion of the bilingual training data.
- **RNNSearch**: It is an attention-based NMT system (Bahdanau et al., 2014).
- **RNNSearch-Mono-SL**: It is our NMT system which makes use of the source-side large-scale monolingual data by applying the self-learning algorithm.
- **RNNSearch-Mono-MTL**: It is our NMT system that exploits the source-side monolingual data by using our multi-task learning framework which performs machine translation and sentence reordering at the same time.
- **RNNSearch-Mono-Autoencoder**: It also applies the multi-task learning framework in which a simple autoencoder is adopted on source-side monolingual data (Luong et al., 2015a).

[8]It needs another 5 to 10 days when adding millions of monolingual data.

5 Translation Results on Small Data

For translation quality evaluation, we attempt to figure out four questions: 1) Can the source-side monolingual data improve the neural machine translation? 2) Could the improved NMT outperform the state-of-the-art phrase-based SMT? 3) Whether it is true that the more the source-side monolingual data the better the translation quality? 4) Which MT model is more suitable to incorporate source-side monolingual data: SMT or NMT?

5.1 Effects of Source-side Monolingual Data in NMT

Table 1 reports the translation quality for different methods. Comparing the first two lines in Table 1, it is obvious that the NMT method *RNNSearch* performs much worse than the SMT model *Moses* on Chinese-to-English translation. The gap is as large as approximately 2.0 BLEU points (28.38 vs. 30.30). We speculate that the encoder-decoder network models of NMT are not well optimized due to insufficient bilingual training data.

The focus of this work is to figure out whether

the encoder model of NMT can be improved using source-side monolingual data and further boost the translation quality. The four lines (3-6 in Table 1) show the BLEU scores when applying self-learning algorithm to incorporate the source-side monolingual data. Clearly, *RNNSearch-Mono-SL* outperforms *RNNSearch* in most cases. The best performance is obtained if the top 50% monolingual data is used. The biggest improvement is up to 4.05 BLEU points (32.43 vs. 28.38 on MT03) and it also significantly outperforms *Moses*.

When employing our multi-task learning framework to incorporate source-side monolingual data, the translation quality can be further improved (Lines 7-10 in Table 1). For example, *RNNSearch-Mono-MTL* using the top 50% monolingual data can remarkably outperform the baseline *RNNSearch*, with an improvement up to 5.0 BLEU points (33.38 vs. 28.38 on MT03). Moreover, it also performs significantly better than the state-of-the-art phrase-based SMT *Moses* by the largest gains of 3.38 BLEU points (31.57 vs. 28.19 on MT05). The promising results demonstrate that source-side monolingual data can improve neural machine translation and our multi-task learning is more effective.

From the last two lines in Table 1, we can see that *RNNSearch-Mono-Autoencoder* can also improve the translation quality by more than 1.0 BLEU points when using the most related monolingual data. However, it underperforms *RNNSearch-Mono-MTL* by a large gap. It indicates that sentence reordering model is better than sentence reconstruction model for exploiting the source-side monolingual data.

Note that we sort the source-side monolingual data according to the word coverage [9] in the bilingual training data. Sentences in the front have more shared words with the source-side vocabulary of bilingual training data. We can clearly see from Table 1 that monolingual data cannot always improve NMT. By adding closely related corpus (25% to 50%), the methods can achieve better and better performance. However, when adding more unrelated monolingual data (75% to 100%) which shares fewer and fewer words in common with the bilingual data, the translation quality becomes worse and worse, and even worse than the baseline *RNNSearch*. Both self-learning algorithm *RNNSearch-Mono-SL* and multi-task learning framework *RNNSearch-Mono-MTL* have the same trend. This indicates that only closely related source-side monolingual data can lead to performance improvement.

[9] In current work, the simple word coverage is applied to indicate the similarity. In the future, we plan to use phrase embedding (Zhang et al., 2014) or sentence embedding (Zhang et al., 2015; Wang et al., 2016a; Wang et al., 2016b) to select the relevant monolingual data.

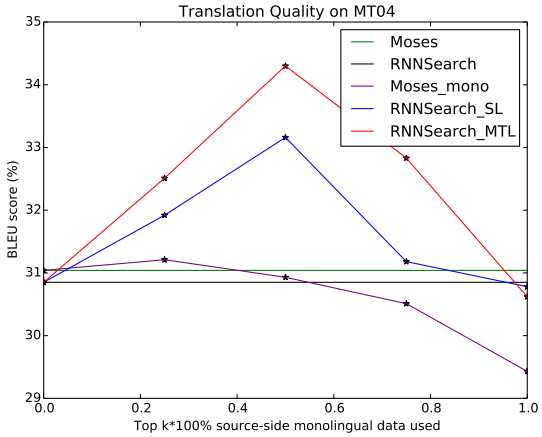


Figure 3: Effects of source-side monolingual data on MT04.

5.2 NMT vs. SMT on Using Source-side Monolingual Data

Although the proposed multi-task learning framework cannot fit SMT because of no shared deep information between the two tasks in SMT, self-learning algorithm can also be applied in SMT as done by (Ueffing et al., 2007; Wu et al., 2008). We may want to know whether NMT is more effective in using source-side monolingual data than SMT.

We apply the self-learning algorithm in SMT by incorporating top 25%, 50%, 75% and 100% synthetic sentence pairs to retrain baseline Moses. Fig. 3 shows the effect of source-side monolingual data in different methods on test set MT04. The figure reveals three similar phenomena. First, related monolingual data can boost the translation quality no matter whether NMT or SMT is used, but mixing more unrelated monolingual corpus will decrease the performance. Second, integrating closely related source-side monolingual data in NMT (*RNNSearch-SL* and *RNNSearch-MTL*) is much more effective than that in SMT (e.g. results for top 50%). It is because that SMT relies on the translation rules

Method	MT03	MT04	MT05	MT06
RNNSearch	35.18	36.20	33.21	32.86
RNNSearch-Mono-MTL (50%)	**36.32**	**37.51**	**35.08**	**34.26**
RNNSearch-Mono-MTL (100%)	35.75	36.74	34.23	33.52

Table 2: Translation results (BLEU score) for different translation methods in large-scale training data.

learnt from the bilingual training data and the synthetic parallel data is obtained by these rules, and thus the synthetic parallel data cannot generate much more information. In contrast, NMT provides a encoder-decoder mechanism and depends heavily on the source language semantic vector representations which facilitate the information sharing. Third, the translation quality changes much more dramatically in NMT methods than that in SMT. It indicates that the neural network models incline to be more affected by the quality of the training data.

6 Translation Results on Large-scale Data

A natural question arises that is the source-side monolingual data still very helpful when we have much more bilingual training data. We conduct the large-scale experiments using our proposed multi-task framework *RNNSearch-Mono-MTL*. Table 2 reports the results.

We can see from the table that closely related source-side monolingual data (the top 50%) can also boost the translation quality on all of the test sets. The performance improvement can be more than 1.0 BLEU points. Compared to the results on small training data, the gains from source-side monolingual data are much smaller. It is reasonable since large-scale training data can make the parameters of the encoder-decoder parameters much stable. We can also observe the similar phenomenon that adding more unrelated monolingual data leads to decreased translation quality.

7 Related Work

As a new paradigm for machine translation, the encoder-decoder based NMT has drawn more and more attention. Most of the existing methods mainly focus on designing better alignment mechanisms (attention model) for the decoder network (Cheng et al., 2016a; Luong et al., 2015b; Cohn et al., 2016; Feng et al., 2016; Tu et al., 2016; Mi et al., 2016a; Mi et al., 2016b), better objective functions for BLEU evaluation (Shen et al., 2016) and better strategies for handling unknown words (Luong et al., 2015c; Sennrich et al., 2015; Li et al., 2016) or large vocabularies (Jean et al., 2015; Mi et al., 2016c).

Our focus in this work is aiming to make full use of the source-side large-scale monolingual data in NMT, which is not fully explored before. The most related works lie in three aspects: 1) applying target-side monolingual data in NMT, 2) targeting knowledge sharing with multi-task NMT, and 3) using source-side monolingual data in conventional SMT and NMT.

Gulcehre et al. (2015) first investigate the target-side monolingual data in NMT. They propose shallow and deep fusion methods to enhance the decoder network by training a big language model on target-side large-scale monolingual data. Sennrich et al. (2015) further propose a new approach to use target-side monolingual data. They generate the synthetic bilingual data by translating the target monolingual sentences to source language sentences and retrain NMT with the mixture of original bilingual data and the synthetic parallel data. It is similar to our self-learning algorithm in which we concern the source-side monolingual data. Furthermore, their method requires to train an additional NMT from target language to source language, which may negatively influence the attention model in the decoder network.

Dong et al. (2015) propose a multi-task learning method for translating one source language into multiple target languages in NMT so that the encoder network can be shared when dealing with several sets of bilingual data. Zoph et al. (2016), Zoph and Knight (2016) and Firat et al. (2016) further deal with more complicated cases (e.g. multi-source languages). Note that all these methods require bilingual training corpus. Instead, we adapt the multi-task learning framework to better accommodate the source-side monolingual data.

Ueffing et al. (2007) and Wu et al. (2008) explore

the usage of source-side monolingual data in conventional SMT with a self-learning algorithm. Although we apply self-learning in this work, we use it to enhance the encoder network in NMT rather than generating more translation rules in SMT and we also adapt a multi-task learning framework to take full advantage of the source-side monolingual data. Luong et al. (2015a) also investigate the source-side monolingual data in the multi-task learning framework, in which a simple autoencoder or skip-thought vectors are employed to model the monolingual data. Our sentence reordering model is more powerful than simple autoencoder in encoder enhancement. Furthermore, they do not carefully prepare the monolingual data for which we show that only related monolingual data leads to big improvements.

In parallel to our work, Cheng et al. (2016b) propose a similar semi-supervised framework to handle both source and target language monolingual data. If source-side monolingual data is considered, a reconstruction framework including two NMTs is employed. One NMT translates the source-side monolingual data into target language translations, from which the other NMT attempts to reconstruct the original source-side monolingual data. In contrast to their approach, we propose a sentence reordering model rather than the sentence reconstruction model. Furthermore, we carefully investigate the relationship between the monolingual data quality and the translation performance improvement.

8 Conclusions and Future Work

In this paper, we propose a self-learning algorithm and a new multi-task learning framework to use source-side monolingual data so as to improve the encoder network of the encoder-decoder based NMT. The self-learning algorithm generates the synthetic parallel corpus and enlarge the bilingual training data to enhance the encoder model of NMT. The multi-task learning framework performs machine translation on bilingual data and sentence reordering on source-side monolingual data by sharing the same encoder network. The experiments show that our method can significantly outperform the strong attention-based NMT baseline, and the proposed multi-task learning framework performs better than the self-learning algorithm at the expense of low efficiency. Furthermore, the experiments also demonstrate that NMT is more effective for incorporating the source-side monolingual data than conventional SMT. We also observe that more monolingual data does not always improve the translation quality and only relevant data does help.

In the future, we would like to design smarter mechanisms to distinguish real data from synthetic data in self-learning algorithm, and attempt to propose better models for handling source-side monolingual data. We also plan to apply our methods in other languages, especially for low-resource languages.

Acknowledgments

We thank the reviewers for their valuable comments and suggestions. This research work has been partially funded by the Natural Science Foundation of China under Grant No. 91520204 and No. 61303181, and supported by the Strategic Priority Research Program of the CAS (Grant XDB02070007).

References

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

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016a. Agreement-based joint training for bidirectional attention-based neural machine translation. In *Proceedings of AAAI 2016*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016b. Semi-supervised learning for neural machine translation. In *Proceedings of ACL 2016*.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

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

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of NAACL 2016*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of ACL 2015*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder nmt model. *arXiv preprint arXiv:1601.03317*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

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

Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL 2015*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP 2013*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007*, pages 177–180.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proceedings of IJCAI 2016*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015c. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL 2015*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016a. A coverage embedding model for neural machine translation. In *Proceedings of EMNLP 2016*.

Haitao Mi, Zhiguo Wang, Niyu Ge, and Abe Ittycheriah. 2016b. Supervised attentions for neural machine translation. In *Proceedings of EMNLP 2016*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016c. Vocabulary manipulation for large vocabulary neural machine translation. In *Proceedings of ACL 2016*.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL 2006*.

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

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL 2016*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS 2014*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. In *Proceedings of ACL 2016*.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of ACL 2007*.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP 2007*.

Zhiguo Wang, Haitao Mi, and Abe Ittycheriah. 2016a. Semi-supervised clustering for short text via deep representation learning. In *Proceedings of CoNLL 2016*.

Zhiguo Wang, Haitao Mi, and Abe Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Hua Wu, Haifeng Wang, and Chengqing Zong. 2008. Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora. In *Proceedings of COLING 2008*, pages 993–1000.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of ACL 2014*.

Jiajun Zhang, Dakun Zhang, and Jie Hao. 2015. Local translation prediction with global sentence representation. In *Proceedings of IJCAI 2015*.

Barret Zoph and Keven Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201v1*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of NAACL 2016*.

Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction

Marcin Junczys-Dowmunt and **Roman Grundkiewicz**
Adam Mickiewicz University in Poznań
ul. Umultowska 87, 61-614 Poznań, Poland
{junczys,romang}@amu.edu.pl

Abstract

In this work, we study parameter tuning towards the M^2 metric, the standard metric for automatic grammar error correction (GEC) tasks. After implementing M^2 as a scorer in the Moses tuning framework, we investigate interactions of dense and sparse features, different optimizers, and tuning strategies for the CoNLL-2014 shared task. We notice erratic behavior when optimizing sparse feature weights with M^2 and offer partial solutions. We find that a bare-bones phrase-based SMT setup with task-specific parameter-tuning outperforms all previously published results for the CoNLL-2014 test set by a large margin (46.37% M^2 over previously 41.75%, by an SMT system with neural features) while being trained on the same, publicly available data. Our newly introduced dense and sparse features widen that gap, and we improve the state-of-the-art to 49.49% M^2.

1 Introduction

Statistical machine translation (SMT), especially the phrase-based variant, is well established in the field of automatic grammatical error correction (GEC) and systems that are either pure SMT or incorporate SMT as system components occupied top positions in GEC shared tasks for different languages.

With the recent paradigm shift in machine translation towards neural translation models, neural encoder-decoder models are expected to appear in the field of GEC as well, and first published results (Xie et al., 2016) already look promising. As it is the case in classical bilingual machine translation research, these models should be compared against strong SMT baselines. Similarly, system combinations of SMT with classifier-based approaches (Rozovskaya and Roth, 2016) suffer from unnecessarily weak MT base systems which make it hard to assess how large the contribution of the classifier pipelines really is. In this work we provide these baselines.

During our experiments, we find that a bare-bones phrase-based system outperforms the best published results on the CoNLL-2014 test set by a significant margin only due to a task-specific parameter tuning when being trained on the same data as previous systems. When we further investigate the influence of well-known SMT-specific features and introduce new features adapted to the problem of GEC, our final systems outperform the best reported results by 8% M^2, moving the state-of-the-art results for the CoNLL-2014 test set from 41.75% M^2 to 49.49%.

The paper is organized as follows: section 2 describes previous work, the CoNLL-2014 shared tasks on GEC and follow-up papers. Our main contributions are presented in sections 3 and 4 where we investigate the interaction of parameter tuning towards the M^2 metric with task-specific dense and sparse features. Especially tuning for sparse features is more challenging than initially expected, but we describe optimizer hyper-parameters that make sparse feature tuning with M^2 feasible. Section 5 reports on the effects of adding a web-scale n-gram language model to our models.

Scripts and models used in this paper are available from `https://github.com/grammatical/baselines-emnlp2016` to facilitate reproducibility of our results.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1546–1556,
Austin, Texas, November 1-5, 2016.

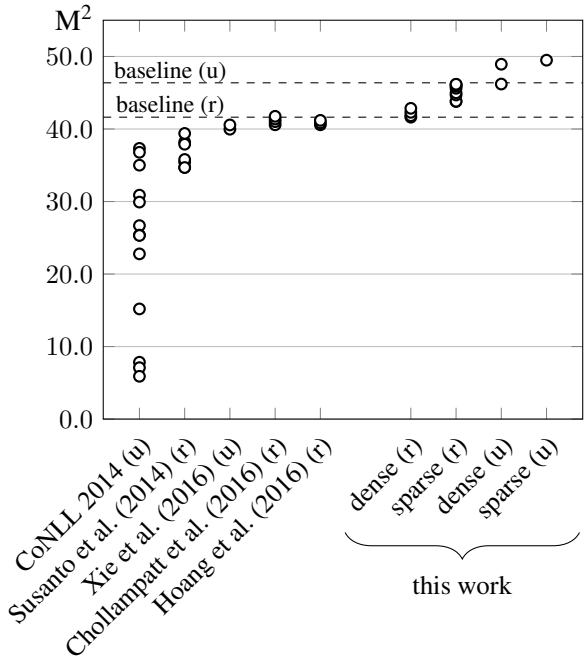


Figure 1: Comparison with previous work on the CoNLL-2014 task, trained on publicly available data. Dashed lines mark results for our baseline systems with restricted (r) and unrestricted (u) data.

2 Previous Work

While machine translation has been used for GEC in works as early as Brockett et al. (2006), we start our discussion with the CoNLL-2014 shared task (Ng et al., 2014) where for the first time an unrestricted set of errors had to be fully corrected. Previous work, most notably during the CoNLL shared-task 2013 (Ng et al., 2013), concentrated only on five selected errors types, but machine translation approaches (Yoshimoto et al., 2013; Yuan and Felice, 2013) were used as well.

The goal of the CoNLL-2014 shared task was to evaluate algorithms and systems for automatically correcting grammatical errors in essays written by second language learners of English. Grammatical errors of 28 types were targeted. Participating teams were given training data with manually annotated corrections of grammatical errors and were allowed to use additional publicly available data.

The corrected system outputs were evaluated blindly using the MaxMatch (M^2) metric (Dahlmeier and Ng, 2012). Thirteen system submissions took part in the shared task. Among the top-three positioned systems, two submissions — CAMB (Felice et al., 2014) and AMU (Junczys-Dowmunt and Grundkiewicz, 2014)[1] — were partially or fully based on SMT. The second system, CUUI (Rozovskaya et al., 2014), was a classifier-based approach, another popular paradigm in GEC.

After the shared task, Susanto et al. (2014) published work on GEC systems combinations. They combined the output from a classification-based system and a SMT-based system using MEMT (Heafield and Lavie, 2010), reporting new state-of-the-art results for the CoNLL-2014 test set.

Xie et al. (2016) presented a neural network-based approach to GEC. Their method relies on a character-level encoder-decoder recurrent neural network with an attention mechanism. They use data from the public Lang-8 corpus and combine their model with an n-gram language model trained on web-scale Common Crawl data.

More recent results are Chollampatt et al. (2016) and Hoang et al. (2016) which also rely on MT systems with new features (a feed-forward neural translation model) and n-best list re-ranking methods. However, most of the improvement over the CoNLL-2014 shared task of these works stems from using the parameter tuning tools we introduced in Junczys-Dowmunt and Grundkiewicz (2014).

In Figure 1 we give a graphical overview of the published results for the CoNLL-2014 test set in comparison to the results we will discuss in this work. Positions marked with (r) use only restricted data which corresponds to the data set used by Susanto et al. (2014). Positions with (u) make use of web-scale data, this corresponds to the resources used in Xie et al. (2016). We marked the participants of the CoNLL-2014 shared task as unrestricted as some participants made use of Common Crawl data or Google n-grams. The visible plateau for results

[1] Junczys-Dowmunt and Grundkiewicz (2014) is our own contribution and introduced many of the concepts discussed in this work, but seemingly to little effect during the task. Later analysis revealed that our submission had an incorrectly filtered language model that was missing many possible entries. Our original system without this deficiency would have achieved results around 44% M^2 already in 2014. This discovery triggered an intensive reanalysis of our shared task system with significantly new conclusions presented in this work. We apologize for supplying these results so late, as this seems to have halted progress in the field for nearly two years.

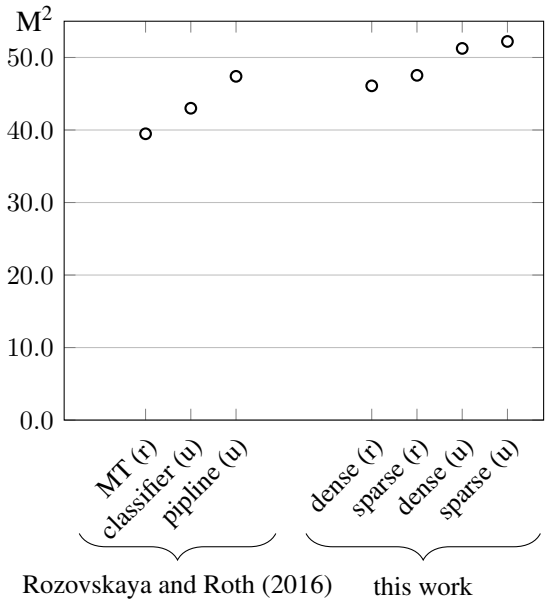


Figure 2: Comparison with Rozovskaya and Roth (2016) using the non-public Lang-8 data set. Here (r) means no web-scale monolingual resources, (u) includes Google 1T n-grams or CommonCrawl.

prior to this work seem to confirm our claims about missing strong baselines.

Rozovskaya and Roth (2016) introduce a SMT-classifier pipeline with state-of-the-art results. Unfortunately, these results are reported for a training set that is not publicly available (data crawled from the Lang-8 website)[2]. Figure 2 compares our results for this resource to Rozovskaya and Roth (2016). See Section 6 for details.

3 Dense feature optimization

Moses comes with tools that can tune parameter vectors according to different MT tuning metrics. Prior work used Moses with default settings: minimum error rate training (Och, 2003) towards BLEU (Papineni et al., 2002). BLEU was never designed for grammatical error correction; we find that directly optimizing for M^2 works far better.

[2]We shared this resource that has been crawled by us for use in Junczys-Dowmunt and Grundkiewicz (2014) privately with Rozovskaya and Roth (2016), but originally were not planning to report results for this resource in the future. We now provide a comparison to Rozovskaya and Roth (2016), but discourage any further use of this unofficial data due to reproducibility issues.

3.1 Tuning towards M^2

The M^2 metric (Dahlmeier and Ng, 2012) is an F-Score, based on the edits extracted from a Levenshtein distance matrix. For the CoNLL-2014 shared task, the β-parameter was set to 0.5, putting two times more weight on precision than on recall.

In Junczys-Dowmunt and Grundkiewicz (2014) we have shown that tuning with BLEU is counterproductive in a setting where M^2 is the evaluation metric. For inherently weak systems this can result in all correction attempts to be disabled, MERT then learns to disallow all changes since they lower the similarity to the reference as determined by BLEU. Systems with better training data, can be tuned with BLEU without suffering this "disabling" effect, but will reach non-optimal performance. However, Susanto et al. (2014) tune the feature weights of their two SMT-based systems with BLEU on the CoNLL-2013 test set and report state-of-the-art results.

Despite tuning with M^2, in Junczys-Dowmunt and Grundkiewicz (2014) we were not able to beat systems that did not tune for the task metric. We reinvestigated these ideas with radically better results, re-implemented the M^2 metric in C++ and added it as a scorer to the Moses parameter optimization framework. Due to this integration we can now tune parameter weights with MERT, PRO or Batch Mira. The inclusion of the latter two enables us to experiment with sparse features.

Based on Clark et al. (2011) concerning the effects of optimizer instability, we report results averaged over five tuning runs. Additionally, we compute parameter weight vector centroids as suggested by Cettolo et al. (2011). They showed that parameter vector centroids averaged over several tuning runs yield similar to or better than average results and reduce variance. We generally confirm this for M^2-based tuning.

3.2 Dense features

The standard features in SMT have been chosen to help guiding the translation process. In a GEC setting the most natural units seem to be minimal edit operations that can be either counted or modeled in context with varying degrees of generalization. That way, the decoder can be informed on several levels

source phrase	target phrase	LD	D	I	S
a short time .	short term only .	3	1	1	1
a situation	into a situation	1	0	1	0
a supermarket .	a supermarket .	0	0	0	0
a supermarket .	at a supermarket	2	1	1	0
able	unable	1	0	0	1

Table 1: Word-based Levenshtein distance (LD) feature and separated edit operations (D = deletions, I = insertions, S = substitutions)

of abstraction how the output differs from the input.[3] In this section we implement several features that try to capture these operation in isolation and in context.

3.2.1 Stateless features

Our stateless features are computed during translation option generation before decoding, modeling relations between source and target phrases. They are meant to extend the standard SMT-specific MLE-based phrase and word translation probabilities with meaningful phrase-level information about the correction process.

Levenshtein distance. In Junczys-Dowmunt and Grundkiewicz (2014) we use word-based Levenshtein distance between source and target as a translation model feature, Felice et al. (2014) independently experiment with a character-based version.

Edit operation counts. We further refine Levenshtein distance feature with edit operation counts. Based on the Levenshtein distance matrix, the numbers of deletions, insertions, and substitutions that transform the source phrase into the target phrase are computed, the sum of these counts is equal to the original Levenshtein distance (see Table 1).

3.2.2 Stateful features

Contrary to stateless features, stateful features can look at translation hypotheses outside their own span and take advantage of the constructed target context. The most typical stateful features are language models. In this section, we discuss LM-like features over edit operations.

[3]We believe this is important information that currently has not yet been mastered in neural encoder-decoder approaches.

Corpus	Sentences	Tokens
NUCLE	57.15 K	1.15 M
CoNLL-2013 Test Set	1.38 K	29.07 K
CoNLL-2014 Test Set	1.31 K	30.11 K
Lang-8	2.23 M	30.03 M
Lang-8 (non-public)	3.72 M	51.07 M
Wikipedia	213.08 M	3.37 G
CommonCrawl (u)	59.13 G	975.63 G

Table 2: Parallel (above line) and monolingual training data.

Operation Sequence Model. Durrani et al. (2013) introduce Operation Sequence Models in Moses. These models are Markov translation models that in our setting can be interpreted as Markov edition models. Translations between identical words are matches, translations that have different words on source and target sides are substitutions; insertions and deletions are interpreted in the same way as for SMT. Gaps, jumps, and other operations typical for OSMs do not appear as we disabled reordering.

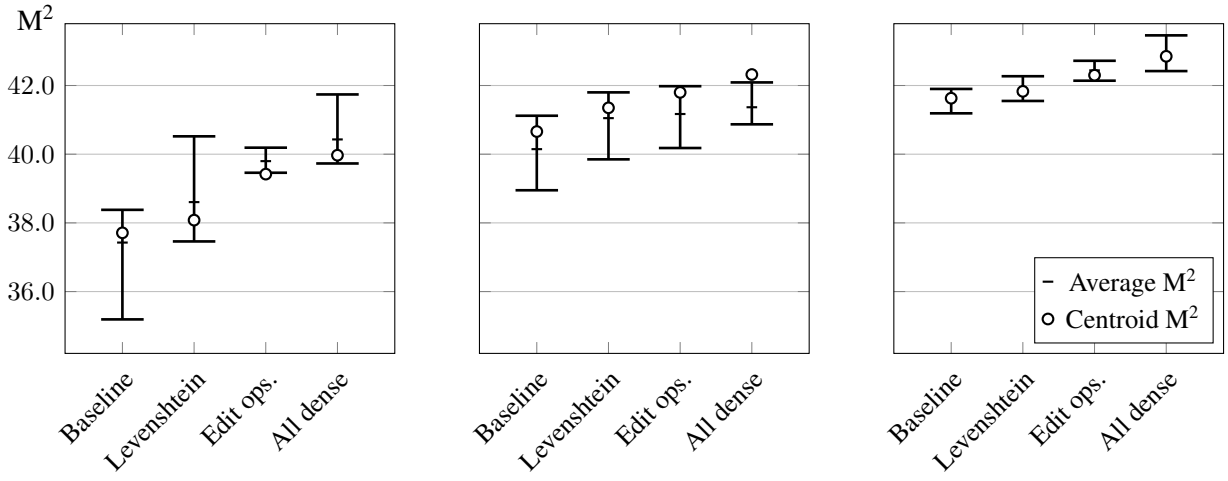
Word-class language model. The monolingual Wikipedia data has been used create a 9-gram word-class language model with 200 word-classes produced by word2vec (Mikolov et al., 2013). This features allows to capture possible long distance dependencies and semantical aspects.

3.3 Training and Test Data

The training data provided in both shared tasks is the NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013). NUCLE consists of 1,414 essays written by Singaporean students who are non-native speakers of English. The essays cover topics, such as environmental pollution, health care, etc. The grammatical errors in these essays have been hand-corrected by professional English teachers and annotated with one of the 28 predefined error type.

Another 50 essays, collected and annotated similarly as NUCLE, were used in both CoNLL GEC shared tasks as blind test data. The CoNLL-2014 test set has been annotated by two human annotators, the CoNLL-2013 by one annotator. Many participants of CoNLL-2014 shared task used the test set from 2013 as development set for their systems.

As mentioned before, we report main results us-



(a) Optimized using BLEU on the CoNLL-2013 test set

(b) Optimized using M^2 on the CoNLL-2013 test set

(c) Optimized using M^2 on 4 folds of error-rate-adapted NUCLE

Figure 3: Results on the CoNLL-2014 test set for different optimization settings (5 runs for each system) and different feature sets, the "All dense" entry includes OSM, the word class language model, and edit operations). The small circle marks results for averaged weights vectors and is chosen as the final result.

ing similar training data as Susanto et al. (2014). We refer to this setting that as the "resticted-data setting" (r). Parallel data for translation model training is adapted from the above mentioned NUCLE corpus and the publicly available Lang-8 corpus (Mizumoto et al., 2012), this corpus is distinct from the non-public web-crawled data described in Section 6. Uncorrected sentences serve as source data, corrected counterparts as target data. For language modeling, the target language sentences of both parallel resources are used, additionally we extract all text from the English Wikipedia.

Phrase-based SMT makes it ease to scale up in terms of training data, especially in the case of n-gram language models. To demonstrate the ease of data integration we propose an "unrestricted setting" (u) based on the data used in Junczys-Dowmunt and Grundkiewicz (2014), one of the shared task submissions, and later in Xie et al. (2016). We use Common Crawl data made-available by Buck et al. (2014).

3.4 Experiments

Our system is based on the phrase-based part of the statistical machine translation system Moses (Koehn et al., 2007). Only plain text data is used for language model and translation model training. External linguistic knowledge is introduced during parameter tuning as the tuning metric relies on the error annotation present in NUCLE. The translation model is built with the standard Moses training script, word-alignment models are produced with MGIZA++ (Gao and Vogel, 2008), we restrict the word alignment training to 5 iterations of Model 1 and 5 iterations of the HMM-Model. No reordering models are used, the distortion limit is set to 0, effectively prohibiting any reordering. All systems use one 5-gram language model that has been estimated from the target side of the parallel data available for translation model training. Another 5-gram language model trained on Wikipedia in the restricted setting or on Common Crawl in the unrestricted case.

Systems are retuned when new features of any type are added. We first successfully reproduce results from Susanto et al. (2014) for BLEU-based tuning on the CoNLL-2013 test set as the development set (Fig. 3a) using similar training data. Repeated tuning places the scores reported by Susanto et al. (2014) for their SMT-ML combinations (37.90 – 39.39) within the range of possible values for a purely Moses-based system without any specific features (35.19 – 38.38) or with just the Levenshtein distance features (37.46 – 40.52). Since Susanto et al. (2014) do not report results for multiple tuning steps, the extend of influence of optimizer

instability on their experiments remains unclear. Even with BLEU-based tuning, we can see significant improvements when replacing Levenshtein distance with the finer-grained edit operations, and another performance jump with additional stateful features. The value range of the different tuning runs for the last feature set includes the currently best-performing system (Xie et al. (2016) with 40.56%), but the result for the averaged centroid are inferior.

Tuning directly with M^2 (Fig. 3b) and averaging weights across five iterations, yields between 40.66% M^2 for a vanilla Moses system and 42.32% for a system with all described dense features. Results seen to be more stable. Averaging weight vectors across runs to produce the final vector seems like a fair bet. Performance with the averaged weight vectors is either similar to or better than the average number for five runs.

3.5 Larger development sets

No less important than choosing the correct tuning metric is a good choice of the development set. Among MT researches, there is a number of more or less well known truths about suitable development sets for translation-focused settings: usually they consist of between 2000 and 3000 sentences, they should be a good representation of the testing data, sparse features require more sentences or more references, etc. Until now, we followed the seemingly obvious approach from Susanto et al. (2014) to tune on the CoNLL-2013 test set. The CoNLL-2013 test set consists of 1380 sentences, which might be barely enough for a translation-task, and it is unclear how to quantify it in the context of grammar correction. Furthermore, calculating the error rate in this set reveals that only 14.97% of the tokens are part of an erroneous fragment, for the rest, input and reference data are identical. Intuitively, this seems to be very little significant data for tuning an SMT system.

We therefore decide to take advantage of the entire NUCLE data as a development set which so far has only been used as translation model training data. NUCLE consist of more than 57,000 sentences, however, the error rate is significantly lower than in the previous development set, only 6.23%. We adapt the error rate by greedily removing sentences from NUCLE until an error rate of ca. 15% is reached, 23381 sentences and most error annotations remain. We further divide the data into four folds. Each folds serves as development set for parameter tuning, while the three remaining parts are treated as translation model training data. The full Lang-8 data is concatenated with is NUCLE training set, and four models are trained. Tuning is then performed four times and the resulting four parameter weight vectors are averaged into a single weight vector across folds. We repeat this procedure again five times which results in 20 separate tuning steps. Results on the CoNLL-2014 test set are obtained using the full translation model with a parameter vector average across five runs. The CoNLL-2013 test set is not being used for tuning and can serve as a second test set.

As can be seen in Fig. 3c, this procedure significantly improves performance, also for the barebones set-up (41.63%). The lower variance between iterations is an effect of averaging across folds.

It turns out that what was meant to be a strong baseline, is actually among the strongest systems reported for this task, outperformed only by the further improvements over this baseline presented in this work.

4 Sparse Features

We saw that introducing finer-grained edit operations improved performance. The natural evolution of that idea are features that describe specific correction operations with and without context. This can be accomplished with sparse features, but tuning sparse features according to the M^2 metric poses unexpected problems.

4.1 Optimizing for M^2 with PRO and Mira

The MERT tool included in Moses cannot handle parameter tuning with sparse feature weights and one of the other optimizers available in Moses has to be used. We first experimented with both, PRO (Hopkins and May, 2011) and Batch Mira (Cherry and Foster, 2012), for the dense features only, and found PRO and Batch Mira with standard settings to either severely underperform in comparison to MERT or to suffer from instability with regard to different test sets (Table 3).

Experiments with Mira hyper-parameters allowed to counter these effects. We first change the

Optimizer	2013	2014
MERT	33.50	42.85
PRO	33.68	40.34
Mira	29.19	34.13
-model-bg	31.06	43.88
-D 0.001	33.86	42.91

Table 3: Tuning with different optimizers with dense features only, results are given for the CoNLL-2013 and CoNLL-2014 test set

background BLEU approximation method in Batch Mira to use model-best hypotheses (`--model-bg`) which seems to produce more satisfactory results. Inspecting the tuning process, however, reveals problems with this setting, too. Figure 4 documents how instable the tuning process with Mira is across iterations. The best result is reached after only three iterations. In a setting with sparse features this would result in only a small set of weighted sparse features.

After consulting with one of the authors of Batch-Mira, we set the background corpus decay rate to 0.001 (`-D 0.001`), resulting in a sentence-level approximation of M^2. Mira's behavior seems to stabilize across iterations. At this point it is not quite clear why this is required. While PRO's behavior is more sane during tuning, results on the test sets are subpar. It seems that no comparable hyper-parameter settings exist for PRO.

4.2 Sparse edit operations

Our sparse edit operations are again based on the Levenshtein distance matrix and count specific edits that are annotated with the source and target tokens that took part in the edit. For the following erroneous/corrected sentence pair

```
Err: Then a new problem comes out .
Cor: Hence , a new problem surfaces .
```

we generate sparse features that model contextless edits (matches are omitted):

```
subst(Then,Hence)=1
insert(,)=1
subst(comes, surfaces)=1
del(out)=1
```

and sparse features with one-sided left or right or two-sided context:

```
<s>_subst(Then,Hence)=1
```

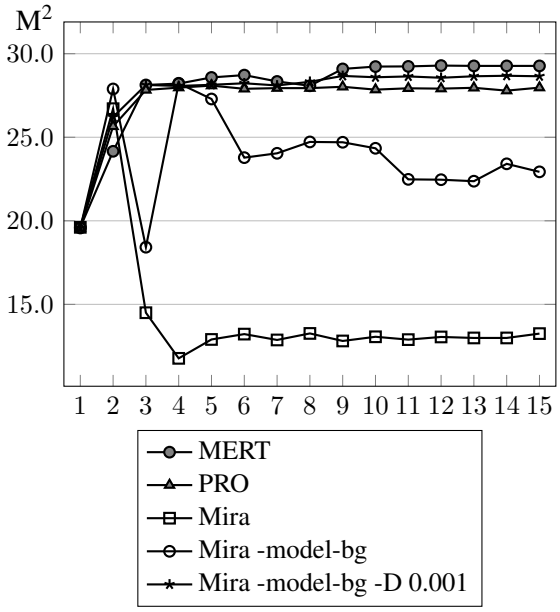
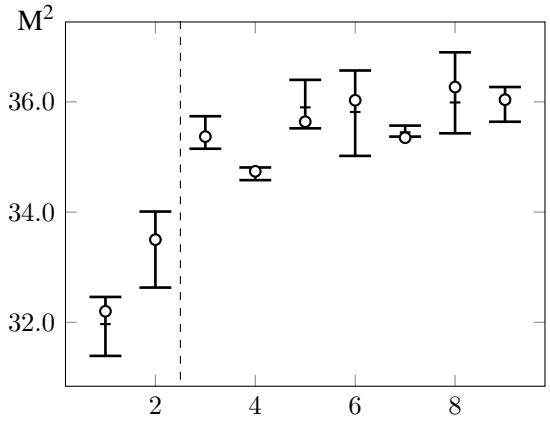


Figure 4: Results per iteration on development set (4-th NUCLE fold)

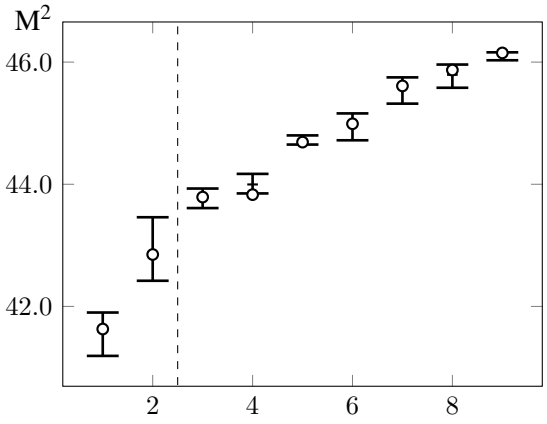
```
subst(Then,Hence)_a=1
Hence_insert(,)=1
insert(,)_a=1
problem_subst(comes, surfaces)=1
subst(comes, surfaces)_out=1
comes_del(out)=1
del(out)_.=1
<s>_subst(Then,Hence)_a=1
Hence_insert(,)_a=1
problem_subst(comes, surfaces)_out=1
comes_del(out)_.=1
```

All sparse feature types are added on-top of our best dense-features system. When using sparse features with context, the contextless features are included. The context annotation comes from the erroneous source sentence, not from the corrected target sentence. We further investigate different source factors: elements taking part in the edit operation or appearing in the context can either be word forms (factor 0) or word classes (factor 1). As before for dense features we average sparse feature weights across folds and multiple tuning runs.

Figure 5 summarizes the results for our sparse feature experiments. On both test sets we can see significant improvements when including edit-based sparse features, the performance increases even more when source context is added. The CoNLL-2013 test set contains annotations from only one annotator and is strongly biased towards high



(a) CoNLL-2014 test set



(b) CoNLL-2014 test set

Symbol	Description
E0	Edit operation on words, no context
E1	Edit operation on word classes, no context
E0C10	Edit operation on words with left/right context of maximum length 1 on words
E1C11	Edit operation on word classes with left/right context of maximum length 1 on word classes
E0C11	Edit operation on words with left/right context of maximum length 1 on word classes

Figure 5: Results on the CoNLL-2013 and CoNLL-2014 test set for different sparse features sets

precision which might explain the greater instability. It appears that sparse features with context where surface forms and word-classes are mixed allow for the best fine-tuning.

5 Adding a web-scale language model

Until now we restricted our experiments to data used by Susanto et al. (2014). However, systems from the CoNLL-2014 were free to use any publicly available data, for instance in Junczys-Dowmunt and Grundkiewicz (2014), we made use of an n-gram language model trained from Common Crawl. Xie et al. (2016) reach the best published result for the task (before this work) by integrating a similar n-gram language model with their neural approach.

We filter the English resources made available by Buck et al. (2014) with cross-entropy filtering (Moore and Lewis, 2010) using the corrected NUCLE corpus as seed data. We keep all sentence with a negative cross-entropy score and compute a 5-gram KenLM (Heafield, 2011) language model with heavy pruning. This step produces roughly 300G of compressed text and a manageable 21G binary model (available for download).

Table 4 summarizes the best results reported in this paper for the CoNLL-2014 test set (column 2014) before and after adding the Common Crawl n-gram language model. The vanilla Moses baseline with the Common Crawl model can be seen as a new simple baseline for unrestricted settings and is ahead of any previously published result. The combination of sparse features and web-scale monolingual data marks our best result, outperforming previously published results by 8% M^2 using similar training data. While our sparse features cause a respectable gain when used with the smaller language model, the web-scale language model seems to cancel out part of the effect.

Bryant and Ng (2015) extended the CoNLL-2014 test set with additional annotations from two to ten annotators. We report results for this valuable resource (column 2014-10) as well.[4] According to Bryant and Ng (2015), human annotators seem to reach on average 72.58% M^2 which can be seen as an upper-bound for the task. In this work, we made a large step towards this upper-bound.

[4]See Bryant and Ng (2015) for a re-assessment of the CoNLL-2014 systems with this extended test set.

System	2014 Prec.	2014 Recall	2014 M^2	2014-10 Prec.	2014-10 Recall	2014-10 M^2
Baseline	48.97	26.03	41.63	69.29	31.35	55.78
+CCLM	58.91	25.05	46.37	77.17	29.38	58.23
Best dense	50.94	26.21	42.85	71.21	31.70	57.00
+CCLM	59.98	28.17	48.93	79.98	32.76	62.08
Best sparse	57.99	25.11	45.95	76.61	29.74	58.25
+CCLM	61.27	27.98	49.49	80.93	32.47	62.33

Table 4: Best results in restricted setting with added unrestricted language model for original (2014) and extended (2014-10) CoNLL test set (trained with public data only).

System	Prec.	Recall	M^2
R&R (np)	60.17	25.64	47.40
Best dense (np)	53.56	29.59	46.09
+CCLM	61.74	30.51	51.25
Best sparse (np)	58.57	27.11	47.54
+CCLM	63.52	30.49	52.21

Table 5: Previous best systems trained with non-public (np) error-corrected data for comparison with Rozovskaya and Roth (2016) denoted as R&R.

6 More error-corrected data

As mentioned before, Rozovskaya and Roth (2016) trained their systems on crawled data from the Lang-8 website that has been collect by us for our submission to the CoNLL-2014 shared task. Since this data has not been made officially available, we treat it as non-public. This makes it difficult to put their results in relation with previously published work, but we can at least provide a comparison for our systems. As our strongest MT-only systems trained on public data already outperform the pipelined approaches from Rozovskaya and Roth (2016), it is unsurprising that adding more error-corrected parallel data results in an even wider gap (Table 5). We can assume that this gap would persist if only public data had been used. Although these are the highest reported results for the CoNLL-2014 shared task so far, we think of them as unofficial results and refer to Table 4 as our final results in this work.

7 Conclusions

Despite the fact that statistical machine translation approaches are among the most popular methods in automatic grammatical error correction, few papers that report results for the CoNLL-2014 test set seem to have exploited its full potential. An important aspect when training SMT systems that one needs to tune parameters towards the task evaluation metric seems to have been under-explored.

We have shown that a pure SMT system actually outperforms the best reported results for any paradigm in GEC if correct parameter tuning is performed. With this tuning mechanism available, task-specific features have been explored that bring further significant improvements, putting phrase-based SMT ahead of other approaches by a large margin. None of the explored features require complicated pipelines or re-ranking mechanisms. Instead they are a natural part of the log-linear model in phrase-based SMT. It is therefore quite easy to reproduce our results and the presented systems may serve as new baselines for automatic grammatical error correction. Our systems and scripts have been made available for better reproducibility.

Acknowledgments

The authors would like to thank Colin Cherry for his help with Batch Mira hyper-parameters and Kenneth Heafield for many helpful comments and discussions. This work was partially funded by the Polish National Science Centre (Grant No. 2014/15/N/ST6/02330) and by Facebook. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Facebook.

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Stroudsburg, USA. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 697–707. Association for Computational Linguistics.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the Common Crawl. In *Proceedings of the Language Resources and Evaluation Conference*, pages 3579–3584, Reykjavík, Iceland.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2011. Methods for smoothing the optimizer instability in SMT. In *MT Summit XIII: the Thirteenth Machine Translation Summit*, pages 32–39.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Stroudsburg, USA. Association for Computational Linguistics.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 176–181, Stroudsburg, USA. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Stroudsburg, USA. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013. Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT? In *ACL (2)*, pages 399–405. The Association for Computer Linguistics.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57. ACL.

Kenneth Heafield and Alon Lavie. 2010. Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, USA. Association for Computational Linguistics.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an smt approach to grammatical error correction.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, USA. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, pages 25–33, Baltimore, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual*

Meeting of the Association for Computational Linguistics. The Association for Computer Linguistics.

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

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012*, pages 863–872.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, , and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, pages 1–14, Baltimore, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, USA. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *CoNLL-2014*, pages 34–42.

Hendy Raymond Susanto, Peter Phandi, and Tou Hwee Ng. 2014. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 951–962. Association for Computational Linguistics.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. NAIST at 2013 CoNLL grammatical error correction shared task. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 26–33, Sofia, Bulgaria. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Incorporating Discrete Translation Lexicons into Neural Machine Translation

Philip Arthur[*], **Graham Neubig**[*†], **Satoshi Nakamura**[*]
[*] Graduate School of Information Science, Nara Institute of Science and Technology
[†] Language Technologies Institute, Carnegie Mellon University
philip.arthur.om0@is.naist.jp gneubig@cs.cmu.edu s-nakamura@is.naist.jp

Abstract

Neural machine translation (NMT) often makes mistakes in translating low-frequency content words that are essential to understanding the meaning of the sentence. We propose a method to alleviate this problem by augmenting NMT systems with discrete translation lexicons that efficiently encode translations of these low-frequency words. We describe a method to calculate the lexicon probability of the next word in the translation candidate by using the attention vector of the NMT model to select which source word lexical probabilities the model should focus on. We test two methods to combine this probability with the standard NMT probability: (1) using it as a bias, and (2) linear interpolation. Experiments on two corpora show an improvement of 2.0-2.3 BLEU and 0.13-0.44 NIST score, and faster convergence time.[1]

1 Introduction

Neural machine translation (NMT, §2; Kalchbrenner and Blunsom (2013), Sutskever et al. (2014)) is a variant of statistical machine translation (SMT; Brown et al. (1993)), using neural networks. NMT has recently gained popularity due to its ability to model the translation process end-to-end using a single probabilistic model, and for its state-of-the-art performance on several language pairs (Luong et al., 2015a; Sennrich et al., 2016).

One feature of NMT systems is that they treat each word in the vocabulary as a vector of continuous-valued numbers. This is in contrast to more traditional SMT methods such as phrase-based machine translation (PBMT; Koehn et al. (2003)), which represent translations as discrete pairs of word strings in the source and target languages. The use of continuous representations is a major advantage, allowing NMT to share statistical power between similar words (e.g. "dog" and "cat") or contexts (e.g. "this is" and "that is"). However, this property also has a drawback in that NMT systems often mistranslate into words that seem natural in the context, but do not reflect the content of the source sentence. For example, Figure 1 is a sentence from our data where the NMT system mistakenly translated "Tunisia" into the word for "Norway." This variety of error is particularly serious because the content words that are often mistranslated by NMT are also the words that play a key role in determining the whole meaning of the sentence.

Input:	I come from Tunisia.
Reference:	チュニジア の 出身です。
	Chunisia no shusshindesu.
	(I'm from Tunisia.)
System:	ノルウェー の 出身です。
	Noruue- no shusshindesu.
	(I'm from Norway.)

Figure 1: An example of a mistake made by NMT on low-frequency content words.

In contrast, PBMT and other traditional SMT methods tend to rarely make this kind of mistake. This is because they base their translations on discrete phrase mappings, which ensure that source words will be translated into a target word that has

[1] Tools to replicate our experiments can be found at http://isw3.naist.jp/~philip-a/emnlp2016/index.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1557–1567,
Austin, Texas, November 1-5, 2016.

been observed as a translation at least once in the training data. In addition, because the discrete mappings are memorized explicitly, they can be learned efficiently from as little as a single instance (barring errors in word alignments). Thus we hypothesize that if we can incorporate a similar variety of information into NMT, this has the potential to alleviate problems with the previously mentioned fatal errors on low-frequency words.

In this paper, we propose a simple, yet effective method to incorporate discrete, probabilistic lexicons as an additional information source in NMT (§3). First we demonstrate how to transform lexical translation probabilities (§3.1) into a predictive probability for the next word by utilizing attention vectors from attentional NMT models (Bahdanau et al., 2015). We then describe methods to incorporate this probability into NMT, either through linear interpolation with the NMT probabilities (§3.2.2) or as the bias to the NMT predictive distribution (§3.2.1). We construct these lexicon probabilities by using traditional word alignment methods on the training data (§4.1), other external parallel data resources such as a handmade dictionary (§4.2), or using a hybrid between the two (§4.3).

We perform experiments (§5) on two English-Japanese translation corpora to evaluate the method's utility in improving translation accuracy and reducing the time required for training.

2 Neural Machine Translation

The goal of machine translation is to translate a sequence of source words $F = f_1^{|F|}$ into a sequence of target words $E = e_1^{|E|}$. These words belong to the source vocabulary V_f, and the target vocabulary V_e respectively. NMT performs this translation by calculating the conditional probability $p_m(e_i|F, e_1^{i-1})$ of the ith target word e_i based on the source F and the preceding target words e_1^{i-1}. This is done by encoding the context $\langle F, e_1^{i-1} \rangle$ a fixed-width vector $\boldsymbol{\eta}_i$, and calculating the probability as follows:

$$p_m(e_i|F, e_1^{i-1}) = \text{softmax}(W_s \boldsymbol{\eta}_i + \boldsymbol{b}_s), \quad (1)$$

where W_s and $\boldsymbol{b}_s$ are respectively weight matrix and bias vector parameters.

The exact variety of the NMT model depends on how we calculate $\boldsymbol{\eta}_i$ used as input. While there are many methods to perform this modeling, we opt to use attentional models (Bahdanau et al., 2015), which focus on particular words in the source sentence when calculating the probability of e_i. These models represent the current state of the art in NMT, and are also convenient for use in our proposed method. Specifically, we use the method of Luong et al. (2015a), which we describe briefly here and refer readers to the original paper for details.

First, an *encoder* converts the source sentence F into a matrix R where each column represents a single word in the input sentence as a continuous vector. This representation is generated using a bidirectional encoder

$$\overrightarrow{\boldsymbol{r}}_j = \text{enc}(\text{embed}(f_j), \overrightarrow{\boldsymbol{r}}_{j-1})$$
$$\overleftarrow{\boldsymbol{r}}_j = \text{enc}(\text{embed}(f_j), \overleftarrow{\boldsymbol{r}}_{j+1})$$
$$\boldsymbol{r}_j = [\overleftarrow{\boldsymbol{r}}_j; \overrightarrow{\boldsymbol{r}}_j].$$

Here the embed$(\cdot)$ function maps the words into a representation (Bengio et al., 2003), and enc$(\cdot)$ is a stacking long short term memory (LSTM) neural network (Hochreiter and Schmidhuber, 1997; Gers et al., 2000; Sutskever et al., 2014). Finally we concatenate the two vectors $\overrightarrow{\boldsymbol{r}}_j$ and $\overleftarrow{\boldsymbol{r}}_j$ into a bidirectional representation $\boldsymbol{r}_j$. These vectors are further concatenated into the matrix R where the jth column corresponds to $\boldsymbol{r}_j$.

Next, we generate the output one word at a time while referencing this encoded input sentence and tracking progress with a *decoder* LSTM. The decoder's hidden state $\boldsymbol{h}_i$ is a fixed-length continuous vector representing the previous target words e_1^{i-1}, initialized as $\boldsymbol{h}_0 = \boldsymbol{0}$. Based on this $\boldsymbol{h}_i$, we calculate a similarity vector $\boldsymbol{\alpha}_i$, with each element equal to

$$\alpha_{i,j} = \text{sim}(\boldsymbol{h}_i, \boldsymbol{r}_j). \quad (2)$$

sim$(\cdot)$ can be an arbitrary similarity function, which we set to the dot product, following Luong et al. (2015a). We then normalize this into an *attention* vector, which weights the amount of focus that we put on each word in the source sentence

$$\boldsymbol{a}_i = \text{softmax}(\boldsymbol{\alpha}_i). \quad (3)$$

This attention vector is then used to weight the encoded representation R to create a context vector $\boldsymbol{c}_i$ for the current time step

$$\boldsymbol{c} = R\boldsymbol{a}.$$

Finally, we create $\boldsymbol{\eta}_i$ by concatenating the previous hidden state $\boldsymbol{h}_{i-1}$ with the context vector, and performing an affine transform

$$\boldsymbol{\eta}_i = W_\eta[\boldsymbol{h}_{i-1}; \boldsymbol{c}_i] + b_\eta,$$

Once we have this representation of the current state, we can calculate $p_m(e_i|F, e_1^{i-1})$ according to Equation (1). The next word e_i is chosen according to this probability, and we update the hidden state by inputting the chosen word into the decoder LSTM

$$\boldsymbol{h}_i = \text{enc}(\text{embed}(e_i), \boldsymbol{h}_{i-1}). \quad (4)$$

If we define all the parameters in this model as θ, we can then train the model by minimizing the negative log-likelihood of the training data

$$\hat{\theta} = \operatorname*{argmin}_\theta \sum_{\langle F,\, E\rangle} \sum_i -\log(p_m(e_i|F, e_1^{i-1}; \theta)).$$

3 Integrating Lexicons into NMT

In §2 we described how traditional NMT models calculate the probability of the next target word $p_m(e_i|e_1^{i-1}, F)$. Our goal in this paper is to improve the accuracy of this probability estimate by incorporating information from discrete probabilistic lexicons. We assume that we have a lexicon that, given a source word f, assigns a probability $p_l(e|f)$ to target word e. For a source word f, this probability will generally be non-zero for a small number of translation candidates, and zero for the majority of words in V_E. In this section, we first describe how we incorporate these probabilities into NMT, and explain how we actually obtain the $p_l(e|f)$ probabilities in §4.

3.1 Converting Lexicon Probabilities into Conditioned Predictive Proabilities

First, we need to convert lexical probabilities $p_l(e|f)$ for the individual words in the source sentence F to a form that can be used together with $p_m(e_i|e_1^{i-1}, F)$. Given input sentence F, we can construct a matrix in which each column corresponds to a word in the input sentence, each row corresponds to a word in the V_E, and the entry corresponds to the appropriate lexical probability:

$$L_F = \begin{bmatrix} p_l(e=1|f_1) & \cdots & p_l(e=1|f_{|F|}) \\ \vdots & \ddots & \vdots \\ p_l(e=|V_e||f_1) & \cdots & p_l(e=|V_e||f_{|F|}) \end{bmatrix}.$$

This matrix can be precomputed during the encoding stage because it only requires information about the source sentence F.

Next we convert this matrix into a predictive probability over the next word: $p_l(e_i|F, e_1^{i-1})$. To do so we use the alignment probability $\boldsymbol{a}$ from Equation (3) to weight each column of the L_F matrix:

$$p_l(e_i|F, e_1^{i-1}) = L_F \boldsymbol{a}_i = \begin{bmatrix} p_l(e=1|f_1) & \cdots & p_{lex}(e=1|f_{|F|}) \\ \vdots & \ddots & \vdots \\ p_l(e=V_e|f_1) & \cdots & p_{lex}(e=V_e|f_{|F|}) \end{bmatrix} \begin{bmatrix} a_{i,1} \\ \vdots \\ a_{i,|F|} \end{bmatrix}.$$

This calculation is similar to the way how attentional models calculate the context vector $\boldsymbol{c}_i$, but over a vector representing the probabilities of the target vocabulary, instead of the distributed representations of the source words. The process of involving $\boldsymbol{a}_i$ is important because at every time step i, the lexical probability $p_l(e_i|e_1^{i-1}, F)$ will be influenced by different source words.

3.2 Combining Predictive Probabilities

After calculating the lexicon predictive probability $p_l(e_i|e_1^{i-1}, F)$, next we need to integrate this probability with the NMT model probability $p_m(e_i|e_1^{i-1}, F)$. To do so, we examine two methods: (1) adding it as a bias, and (2) linear interpolation.

3.2.1 Model Bias

In our first `bias` method, we use $p_l(\cdot)$ to bias the probability distribution calculated by the vanilla NMT model. Specifically, we add a small constant ϵ to $p_l(\cdot)$, take the logarithm, and add this adjusted log probability to the input of the softmax as follows:

$$p_b(e_i|F, e_1^{i-1}) = \text{softmax}(W_s\boldsymbol{\eta}_i + b_s + \log(p_l(e_i|F, e_1^{i-1}) + \epsilon)).$$

We take the logarithm of $p_l(\cdot)$ so that the values will still be in the probability domain after the softmax is calculated, and add the hyper-parameter ϵ to prevent zero probabilities from becoming $-\infty$ after taking the log. When ϵ is small, the model will be more heavily biased towards using the lexicon, and when ϵ is larger the lexicon probabilities will be given less weight. We use $\epsilon = 0.001$ for this paper.

3.2.2 Linear Interpolation

We also attempt to incorporate the two probabilities through linear interpolation between the standard NMT probability model probability $p_m(\cdot)$ and the lexicon probability $p_l(\cdot)$. We will call this the `linear` method, and define it as follows:

$$p_o(e_i|F,e_1^{i-1}) = \begin{bmatrix} p_l(e_i=1|F,e_1^{i-1}) & p_m(e=1|F,e_1^{i-1}) \\ \vdots & \vdots \\ p_l(e_i=|V_e||F,e_1^{i-1}) & p_m(e=|V_e||F,e_1^{i-1}) \end{bmatrix} \begin{bmatrix} \lambda \\ 1-\lambda \end{bmatrix},$$

where λ is an interpolation coefficient that is the result of the sigmoid function $\lambda = \text{sig}(x) = \frac{1}{1+e^{-x}}$. x is a learnable parameter, and the sigmoid function ensures that the final interpolation level falls between 0 and 1. We choose $x = 0$ ($\lambda = 0.5$) at the beginning of training.

This notation is partly inspired by Allamanis et al. (2016) and Gu et al. (2016) who use linear interpolation to merge a standard attentional model with a "copy" operator that copies a source word as-is into the target sentence. The main difference is that they use this to copy words into the output while our method uses it to influence the probabilities of all target words.

4 Constructing Lexicon Probabilities

In the previous section, we have defined some ways to use predictive probabilities $p_l(e_i|F,e_1^{i-1})$ based on word-to-word lexical probabilities $p_l(e|f)$. Next, we define three ways to construct these lexical probabilities using automatically learned lexicons, handmade lexicons, or a combination of both.

4.1 Automatically Learned Lexicons

In traditional SMT systems, lexical translation probabilities are generally learned directly from parallel data in an unsupervised fashion using a model such as the IBM models (Brown et al., 1993; Och and Ney, 2003). These models can be used to estimate the alignments and lexical translation probabilities $p_l(e|f)$ between the tokens of the two languages using the expectation maximization (EM) algorithm.

First in the expectation step, the algorithm estimates the expected count $c(e|f)$. In the maximization step, lexical probabilities are calculated by dividing the expected count by all possible counts:

$$p_{l,a}(e|f) = \frac{c(f,e)}{\sum_{\tilde{e}} c(f,\tilde{e})},$$

The IBM models vary in level of refinement, with Model 1 relying solely on these lexical probabilities, and latter IBM models (Models 2, 3, 4, 5) introducing more sophisticated models of fertility and relative alignment. Even though IBM models also occasionally have problems when dealing with the rare words (e.g. "garbage collecting" effects (Liang et al., 2006)), traditional SMT systems generally achieve better translation accuracies of low-frequency words than NMT systems (Sutskever et al., 2014), indicating that these problems are less prominent than they are in NMT.

Note that in many cases, NMT limits the target vocabulary (Jean et al., 2015) for training speed or memory constraints, resulting in rare words not being covered by the NMT vocabulary V_E. Accordingly, we allocate the remaining probability assigned by the lexicon to the unknown word symbol $\langle\text{unk}\rangle$:

$$p_{l,a}(e=\langle\text{unk}\rangle|f) = 1 - \sum_{i\in V_e} p_{l,a}(e=i|f). \quad (5)$$

4.2 Manual Lexicons

In addition, for many language pairs, broad-coverage handmade dictionaries exist, and it is desirable that we be able to use the information included in them as well. Unlike automatically learned lexicons, however, handmade dictionaries generally do not contain translation probabilities. To construct the probability $p_l(e|f)$, we define the set of translations K_f existing in the dictionary for particular source word f, and assume a uniform distribution over these words:

$$p_{l,m}(e|f) = \begin{cases} \frac{1}{|K_f|} & \text{if } e \in K_f \\ 0 & \text{otherwise} \end{cases}.$$

Following Equation (5), unknown source words will assign their probability mass to the $\langle\text{unk}\rangle$ tag.

4.3 Hybrid Lexicons

Handmade lexicons have broad coverage of words but their probabilities might not be as accurate as the

Data	Corpus	Sentence	Tokens	
			En	Ja
Train	BTEC	464K	3.60M	4.97M
	KFTT	377K	7.77M	8.04M
Dev	BTEC	510	3.8K	5.3K
	KFTT	1160	24.3K	26.8K
Test	BTEC	508	3.8K	5.5K
	KFTT	1169	26.0K	28.4K

Table 1: Corpus details.

learned ones, particularly if the automatic lexicon is constructed on in-domain data. Thus, we also test a `hybrid` method where we use the handmade lexicons to complement the automatically learned lexicon.[2] [3] Specifically, inspired by phrase table fill-up used in PBMT systems (Bisazza et al., 2011), we use the probability of the automatically learned lexicons $p_{l,a}$ by default, and fall back to the handmade lexicons $p_{l,m}$ only for uncovered words:

$$p_{l,h}(e|f) = \begin{cases} p_{l,a}(e|f) & \text{if } f \text{ is covered} \\ p_{l,m}(e|f) & \text{otherwise} \end{cases} \quad (6)$$

5 Experiment & Result

In this section, we describe experiments we use to evaluate our proposed methods.

5.1 Settings

Dataset: We perform experiments on two widely-used tasks for the English-to-Japanese language pair: KFTT (Neubig, 2011) and BTEC (Kikui et al., 2003). KFTT is a collection of Wikipedia article about city of Kyoto and BTEC is a travel conversation corpus. BTEC is an easier translation task than KFTT, because KFTT covers a broader domain, has a larger vocabulary of rare words, and has relatively long sentences. The details of each corpus are depicted in Table 1.

We tokenize English according to the Penn Treebank standard (Marcus et al., 1993) and lowercase, and tokenize Japanese using KyTea (Neubig et al., 2011). We limit training sentence length up to 50 in both experiments and keep the test data at the original length. We replace words of frequency less than a threshold u in both languages with the $\langle$unk$\rangle$ symbol and exclude them from our vocabulary. We choose $u = 1$ for BTEC and $u = 3$ for KFTT, resulting in $|V_f| = 17.8$k, $|V_e| = 21.8$k for BTEC and $|V_f| = 48.2$k, $|V_e| = 49.1$k for KFTT.

NMT Systems: We build the described models using the Chainer[4] toolkit. The depth of the stacking LSTM is $d = 4$ and hidden node size $h = 800$. We concatenate the forward and backward encodings (resulting in a 1600 dimension vector) and then perform a linear transformation to 800 dimensions.

We train the system using the Adam (Kingma and Ba, 2014) optimization method with the default settings: $\alpha = 1\text{e}{-3}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1\text{e}{-8}$. Additionally, we add dropout (Srivastava et al., 2014) with drop rate $r = 0.2$ at the last layer of each stacking LSTM unit to prevent overfitting. We use a batch size of $B = 64$ and we run a total of $N = 14$ iterations for all data sets. All of the experiments are conducted on a single GeForce GTX TITAN X GPU with a 12 GB memory cache.

At test time, we use beam search with beam size $b = 5$. We follow Luong et al. (2015b) in replacing every unknown token at position i with the target token that maximizes the probability $p_{l,a}(e_i|f_j)$. We choose source word f_j according to the highest alignment score in Equation (3). This unknown word replacement is applied to both baseline and proposed systems. Finally, because NMT models tend to give higher probabilities to shorter sentences (Cho et al., 2014), we discount the probability of $\langle$EOS$\rangle$ token by 10% to correct for this bias.

Traditional SMT Systems: We also prepare two traditional SMT systems for comparison: a PBMT system (Koehn et al., 2003) using Moses[5] (Koehn et al., 2007), and a hierarchical phrase-based MT system (Chiang, 2007) using Travatar[6] (Neubig, 2013), Systems are built using the default settings, with models trained on the training data, and weights tuned on the development data.

Lexicons: We use a total of 3 lexicons for the

[2] Alternatively, we could imagine a method where we combined the training data and dictionary before training the word alignments to create the lexicon. We attempted this, and results were comparable to or worse than the fill-up method, so we use the fill-up method for the remainder of the paper.

[3] While most words in the V_f will be covered by the learned lexicon, many words (13% in experiments) are still left uncovered due to alignment failures or other factors.

[4] `http://chainer.org/index.html`

[5] `http://www.statmt.org/moses/`

[6] `http://www.phontron.com/travatar/`

System	BTEC			KFTT		
	BLEU	NIST	RECALL	BLEU	NIST	RECALL
pbmt	48.18	6.05	27.03	22.62	5.79	13.88
hiero	52.27	6.34	24.32	22.54	5.82	12.83
attn	48.31	5.98	17.39	20.86	5.15	17.68
auto-bias	**49.74***	**6.11***	**50.00**	**23.20**†	**5.59**†	**19.32**
hyb-bias	**50.34**†	**6.10***	**41.67**	**22.80**†	**5.55**†	16.67

Table 2: Accuracies for the baseline attentional NMT (attn) and the proposed bias-based method using the automatic (auto-bias) or hybrid (hyb-bias) dictionaries. Bold indicates a gain over the attn baseline, † indicates a significant increase at $p < 0.05$, and $*$ indicates $p < 0.10$. Traditional phrase-based (pbmt) and hierarchical phrase based (hiero) systems are shown for reference.

proposed method, and apply bias and linear method for all of them, totaling 6 experiments. The first lexicon (auto) is built on the training data using the automatically learned lexicon method of §4.1 separately for both the BTEC and KFTT experiments. Automatic alignment is performed using GIZA++ (Och and Ney, 2003). The second lexicon (man) is built using the popular English-Japanese dictionary Eijiro[7] with the manual lexicon method of §4.2. Eijiro contains 104K distinct word-to-word translation entries. The third lexicon (hyb) is built by combining the first and second lexicon with the hybrid method of §4.3.

Evaluation: We use standard single reference BLEU-4 (Papineni et al., 2002) to evaluate the translation performance. Additionally, we also use NIST (Doddington, 2002), which is a measure that puts a particular focus on low-frequency word strings, and thus is sensitive to the low-frequency words we are focusing on in this paper. We measure the statistical significant differences between systems using paired bootstrap resampling (Koehn, 2004) with 10,000 iterations and measure statistical significance at the $p < 0.05$ and $p < 0.10$ levels.

Additionally, we also calculate the recall of rare words from the references. We define "rare words" as words that appear less than eight times in the target training corpus or references, and measure the percentage of time they are recovered by each translation system.

5.2 Effect of Integrating Lexicons

In this section, we first a detailed examination of the utility of the proposed bias method when used with the auto or hyb lexicons, which empirically gave the best results, and perform a comparison among the other lexicon integration methods in the following section. Table 2 shows the results of these methods, along with the corresponding baselines.

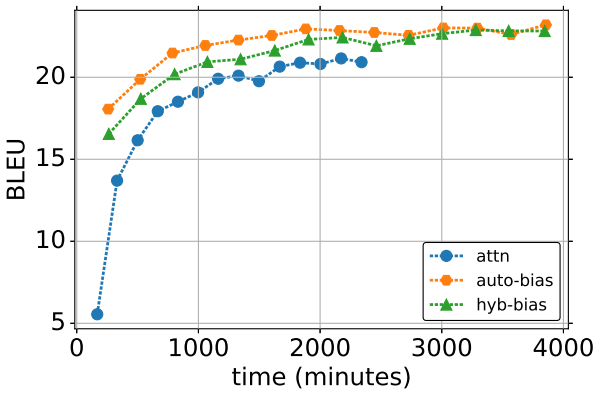


Figure 2: Training curves for the baseline attn and the proposed bias method.

First, compared to the baseline attn, our bias method achieved consistently higher scores on both test sets. In particular, the gains on the more difficult KFTT set are large, up to 2.3 BLEU, 0.44 NIST, and 30% Recall, demonstrating the utility of the proposed method in the face of more diverse content and fewer high-frequency words.

Compared to the traditional pbmt systems hiero, particularly on KFTT we can see that the proposed method allows the NMT system to exceed the traditional SMT methods in BLEU. This is despite the fact that we are not performing ensembling, which has proven to be essential to exceed traditional systems in several previous works (Sutskever

[7] http://eijiro.jp

Input	Do you have an opinion regarding extramarital affairs?
Reference	不倫 に 関して 意見 が あります か。 Furin ni kanshite iken ga arimasu ka.
attn	サッカー に 関する 意見 は あります か。 Sakkā ni kansuru iken wa arimasu ka. *(Do you have an opinion about soccer?)*
auto-bias	不倫 に 関して 意見 が あります か。 Furin ni kanshite iken ga arimasu ka. *(Do you have an opinion about affairs?)*
Input	Could you put these fragile things in a safe place?
Reference	この 壊れ物 を 安全な 場所 に 置いて もらえません か 。 Kono kowaremono o anzen'na basho ni oite moraemasen ka.
attn	貴重品 を 安全 に 出したい の ですが 。 Kichō-hin o anzen ni dashitai nodesuga. *(I'd like to safely put out these valuables.)*
auto-bias	この 壊れ物 を 安全な 場所 に 置いて もらえません か 。 Kono kowaremono o anzen'na basho ni oite moraemasen ka. *(Could you put these fragile things in a safe place?)*

Table 3: Examples where the proposed auto-bias improved over the baseline system attn. Underlines indicate words were mistaken in the baseline output but correct in the proposed model's output.

et al., 2014; Luong et al., 2015a; Sennrich et al., 2016). Interestingly, despite gains in BLEU, the NMT methods still fall behind in NIST score on the KFTT data set, demonstrating that traditional SMT systems still tend to have a small advantage in translating lower-frequency words, despite the gains made by the proposed method.

In Table 3, we show some illustrative examples where the proposed method (auto-bias) was able to obtain a correct translation while the normal attentional model was not. The first example is a mistake in translating "extramarital affairs" into the Japanese equivalent of "soccer," entirely changing the main topic of the sentence. This is typical of the errors that we have observed NMT systems make (the mistake from Figure 1 is also from attn, and was fixed by our proposed method). The second example demonstrates how these mistakes can then affect the process of choosing the remaining words, propagating the error through the whole sentence.

Next, we examine the effect of the proposed method on the training time for each neural MT method, drawing training curves for the KFTT data in Figure 2. Here we can see that the proposed bias training methods achieve reasonable BLEU scores in the upper 10s even after the first iteration. In contrast, the baseline attn method has a BLEU score of around 5 after the first iteration, and takes significantly longer to approach values close to its maximal

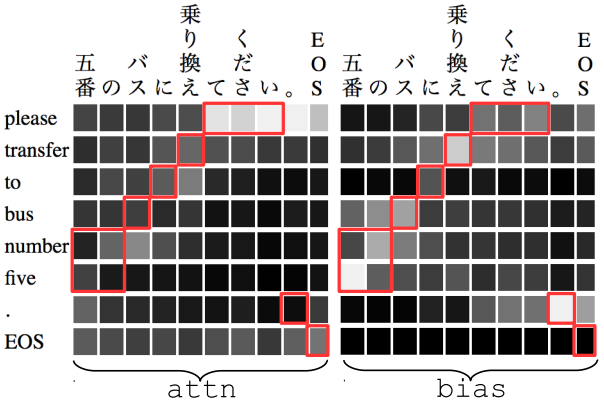


Figure 3: Attention matrices for baseline attn and proposed bias methods. Lighter colors indicate stronger attention between the words, and boxes surrounding words indicate the correct alignments.

accuracy. This shows that by incorporating lexical probabilities, we can effectively bootstrap the learning of the NMT system, allowing it to approach an appropriate answer in a more timely fashion.[8]

It is also interesting to examine the alignment vectors produced by the baseline and proposed meth-

[8]Note that these gains are despite the fact that one iteration of the proposed method takes a longer (167 minutes for attn vs. 275 minutes for auto-bias) due to the necessity to calculate and use the lexical probability matrix for each sentence. It also takes an additional 297 minutes to train the lexicon with GIZA++, but this can be greatly reduced with more efficient training methods (Dyer et al., 2013).

(a) BTEC

Lexicon	BLEU		NIST	
	bias	linear	bias	linear
-	48.31		5.98	
auto	**49.74***	47.97	**6.11**	5.90
man	**49.08**	**51.04**†	**6.03***	**6.14**†
hyb	**50.34**†	**49.27**	**6.10***	5.94

(b) KFTT

Lexicon	BLEU		NIST	
	bias	linear	bias	linear
-	20.86		5.15	
auto	**23.20**†	18.19	**5.59**†	4.61
man	20.78	**20.88**	5.12	5.11
hyb	**22.80**†	20.33	**5.55**†	5.03

Table 4: A comparison of the bias and linear lexicon integration methods on the automatic, manual, and hybrid lexicons. The first line without lexicon is the traditional attentional NMT.

ods, a visualization of which we show in Figure 3. For this sentence, the outputs of both methods were both identical and correct, but we can see that the proposed method (right) placed sharper attention on the actual source word corresponding to content words in the target sentence. This trend of peakier attention distributions in the proposed method held throughout the corpus, with the per-word entropy of the attention vectors being 3.23 bits for auto-bias, compared with 3.81 bits for attn, indicating that the auto-bias method places more certainty in its attention decisions.

5.3 Comparison of Integration Methods

Finally, we perform a full comparison between the various methods for integrating lexicons into the translation process, with results shown in Table 4. In general the bias method improves accuracy for the auto and hyb lexicon, but is less effective for the man lexicon. This is likely due to the fact that the manual lexicon, despite having broad coverage, did not sufficiently cover target-domain words (coverage of unique words in the source vocabulary was 35.3% and 9.7% for BTEC and KFTT respectively).

Interestingly, the trend is reversed for the linear method, with it improving man systems, but causing decreases when using the auto and hyb lexicons. This indicates that the linear method is more suited for cases where the lexicon does not closely match the target domain, and plays a more complementary role. Compared to the log-linear modeling of bias, which strictly enforces constraints imposed by the lexicon distribution (Klakow, 1998), linear interpolation is intuitively more appropriate for integrating this type of complimentary information.

On the other hand, the performance of linear interpolation was generally lower than that of the bias method. One potential reason for this is the fact that we use a constant interpolation coefficient that was set fixed in every context. Gu et al. (2016) have recently developed methods to use the context information from the decoder to calculate the different interpolation coefficients for every decoding step, and it is possible that introducing these methods would improve our results.

6 Additional Experiments

To test whether the proposed method is useful on larger data sets, we also performed follow-up experiments on the larger Japanese-English ASPEC dataset (Nakazawa et al., 2016) that consist of 2 million training examples, 63 million tokens, and 81,000 vocabulary size. We gained an improvement in BLEU score from 20.82 using the attn baseline to 22.66 using the auto-bias proposed method. This experiment shows that our method scales to larger datasets.

7 Related Work

From the beginning of work on NMT, unknown words that do not exist in the system vocabulary have been focused on as a weakness of these systems. Early methods to handle these unknown words replaced them with appropriate words in the target vocabulary (Jean et al., 2015; Luong et al., 2015b) according to a lexicon similar to the one used in this work. In contrast to our work, these only handle unknown words and do not incorporate information from the lexicon in the learning procedure.

There have also been other approaches that incorporate models that learn when to copy words as-is into the target language (Allamanis et al., 2016; Gu et al., 2016; Gülçehre et al., 2016). These models

are similar to the `linear` approach of §3.2.2, but are only applicable to words that can be copied as-is into the target language. In fact, these models can be thought of as a subclass of the proposed approach that use a lexicon that assigns a all its probability to target words that are the same as the source. On the other hand, while we are simply using a static interpolation coefficient λ, these works generally have a more sophisticated method for choosing the interpolation between the standard and "copy" models. Incorporating these into our `linear` method is a promising avenue for future work.

In addition Mi et al. (2016) have also recently proposed a similar approach by limiting the number of vocabulary being predicted by each batch or sentence. This vocabulary is made by considering the original HMM alignments gathered from the training corpus. Basically, this method is a specific version of our bias method that gives some of the vocabulary a bias of negative infinity and all other vocabulary a uniform distribution. Our method improves over this by considering actual translation probabilities, and also considering the attention vector when deciding how to combine these probabilities.

Finally, there have been a number of recent works that improve accuracy of low-frequency words using character-based translation models (Ling et al., 2015; Costa-Jussà and Fonollosa, 2016; Chung et al., 2016). However, Luong and Manning (2016) have found that even when using character-based models, incorporating information about words allows for gains in translation accuracy, and it is likely that our lexicon-based method could result in improvements in these hybrid systems as well.

8 Conclusion & Future Work

In this paper, we have proposed a method to incorporate discrete probabilistic lexicons into NMT systems to solve the difficulties that NMT systems have demonstrated with low-frequency words. As a result, we achieved substantial increases in BLEU (2.0-2.3) and NIST (0.13-0.44) scores, and observed qualitative improvements in the translations of content words.

For future work, we are interested in conducting the experiments on larger-scale translation tasks. We also plan to do subjective evaluation, as we expect that improvements in content word translation are critical to subjective impressions of translation results. Finally, we are also interested in improvements to the `linear` method where λ is calculated based on the context, instead of using a fixed value.

Acknowledgment

We thank Makoto Morishita and Yusuke Oda for their help in this project. We also thank the faculty members of AHC lab for their supports and suggestions.

This work was supported by grants from the Ministry of Education, Culture, Sport, Science, and Technology of Japan and in part by JSPS KAKENHI Grant Number 16H05873.

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of the 33th International Conference on Machine Learning (ICML)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the 2011 International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 103–111.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1693–1703.

Marta R. Costa-Jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 357–361.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation*, pages 2451–2471.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631–1640.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 140–149.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, pages 1735–1780.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th Internationali Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1–10.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709.

Gen-ichiro Kikui, Eiichiro Sumita, Toshiyuki Takezawa, and Seiichi Yamamoto. 2003. Creating corpora for speech-to-speech translation. In *8th European Conference on Speech Communication and Technology, EUROSPEECH 2003 - INTERSPEECH 2003, Geneva, Switzerland, September 1-4, 2003*, pages 381–384.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Dietrich Klakow. 1998. Log-linear interpolation of language models. In *Proceedings of the 5th International Conference on Speech and Language Processing (ICSLP)*.

Phillip Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 104–111.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based neural machine translation. *CoRR*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1054–1063.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th Internationali Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 11–19.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, pages 313–330.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 124–129.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 529–533.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–96.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, pages 19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Transfer Learning for Low-Resource Neural Machine Translation

Barret Zoph*
Information Sciences Institute
University of Southern California
barretzoph@gmail.com

Deniz Yuret
Computer Engineering
Koç University
dyuret@ku.edu.tr

Jonathan May and Kevin Knight
Information Sciences Institute
Computer Science Department
University of Southern California
{jonmay,knight}@isi.edu

Abstract

The encoder-decoder framework for neural machine translation (NMT) has been shown effective in large data scenarios, but is much less effective for low-resource languages. We present a transfer learning method that significantly improves BLEU scores across a range of low-resource languages. Our key idea is to first train a high-resource language pair (the *parent model*), then transfer some of the learned parameters to the low-resource pair (the *child model*) to initialize and constrain training. Using our transfer learning method we improve baseline NMT models by an average of 5.6 BLEU on four low-resource language pairs. Ensembling and unknown word replacement add another 2 BLEU which brings the NMT performance on low-resource machine translation close to a strong syntax based machine translation (SBMT) system, exceeding its performance on one language pair. Additionally, using the transfer learning model for re-scoring, we can improve the SBMT system by an average of 1.3 BLEU, improving the state-of-the-art on low-resource machine translation.

1 Introduction

Neural machine translation (NMT) (Sutskever et al., 2014) is a promising paradigm for extracting translation knowledge from parallel text. NMT systems have achieved competitive accuracy rates under large-data training conditions for language pairs such as English–French. However, neural methods are data-hungry and learn poorly from low-count events. This behavior makes vanilla NMT a poor choice for low-resource languages, where parallel data is scarce. Table 1 shows that for 4 low-resource languages, a standard string-to-tree statistical MT system (SBMT) (Galley et al., 2004; Galley et al., 2006) strongly outperforms NMT, even when NMT uses the state-of-the-art local attention plus feed-input techniques from Luong et al. (2015a).

Language	Train Size	Test Size	SBMT BLEU	NMT BLEU
Hausa	1.0m	11.3K	23.7	16.8
Turkish	1.4m	11.6K	20.4	11.4
Uzbek	1.8m	11.5K	17.9	10.7
Urdu	0.2m	11.4K	17.9	5.2

Table 1: NMT models with attention are outperformed by standard string-to-tree statistical MT (SBMT) when translating low-resource languages into English. Train/test bitext corpus sizes are English token counts. Single-reference, case-insensitive BLEU scores are given for held-out test corpora.

In this paper, we describe a method for substantially improving NMT results on these languages. Our key idea is to first train a high-resource language pair, then use the resulting trained network (the *parent model*) to initialize and constrain training for our low-resource language pair (the *child model*). We find that we can optimize our results by fixing certain parameters of the parent model and letting the rest be fine-tuned by the child model. We report NMT improvements from transfer learning of 5.6 BLEU on average, and we provide an analysis of why the method works. The final NMT system

This work was carried out while all authors were at USC's Information Sciences Institute.

*This author is currently at Google Brain.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1568–1575,
Austin, Texas, November 1-5, 2016.

approaches strong SBMT baselines in all four language pairs, and exceeds SBMT performance in one of them. Furthermore, we show that NMT is an exceptional re-scorer of 'traditional' MT output; even NMT that on its own is worse than SBMT is consistently able to improve upon SBMT system output when incorporated as a re-scoring model.

We provide a brief description of our NMT model in Section 2. Section 3 gives some background on transfer learning and explains how we use it to improve machine translation performance. Our main experiments translating Hausa, Turkish, Uzbek, and Urdu into English with the help of a French–English parent model are presented in Section 4. Section 5 explores alternatives to our model to enhance understanding. We find that the choice of parent language pair affects performance, and provide an empirical upper bound on transfer performance using an artificial language. We experiment with English-only language models, copy models, and word-sorting models to show that what we transfer goes beyond monolingual information and that using a translation model trained on bilingual corpora as a parent is essential. We show the effects of freezing, fine-tuning, and smarter initialization of different components of the attention-based NMT system during transfer. We compare the learning curves of transfer and no-transfer models, showing that transfer solves an overfitting problem, not a search problem. We summarize our contributions in Section 6.

2 NMT Background

In the neural encoder-decoder framework for MT (Neco and Forcada, 1997; Castaño and Casacuberta, 1997; Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a), we use a recurrent neural network (encoder) to convert a source sentence into a dense, fixed-length vector. We then use another recurrent network (decoder) to convert that vector to a target sentence. In this paper, we use a two-layer encoder-decoder system (Figure 1) with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The models were trained to optimize maximum likelihood (via a softmax layer) with back-propagation through time (Werbos, 1990). Additionally, we use an attention mechanism that allows the target decoder to look back at

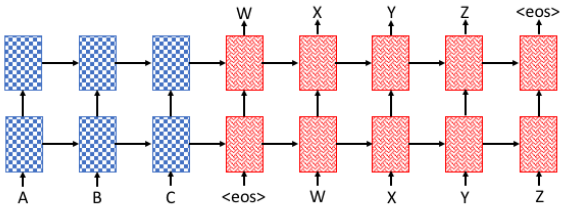


Figure 1: The encoder-decoder framework for neural machine translation (NMT) (Sutskever et al., 2014). Here, a source sentence C B A (presented in reverse order as A B C) is translated into a target sentence W X Y Z. At each step, an evolving real-valued vector summarizes the state of the encoder (blue, checkerboard) and decoder (red, lattice). Not shown here are the attention connections present in our model used by the decoder to access encoder states.

the source encoder, specifically the local attention model from Luong et al. (2015a). In our model we also use the feed-input input connection from Luong et al. (2015a) where at each timestep on the decoder we feed in the top layer's hidden state into the lowest layer of the next timestep.

3 Transfer Learning

Transfer learning uses knowledge from a learned task to improve the performance on a related task, typically reducing the amount of required training data (Torrey and Shavlik, 2009; Pan and Yang, 2010). In natural language processing, transfer learning methods have been successfully applied to speech recognition, document classification and sentiment analysis (Wang and Zheng, 2015). Deep learning models discover multiple levels of representation, some of which may be useful across tasks, which makes them particularly suited to transfer learning (Bengio, 2012). For example, Cireşan et al. (2012) use a convolutional neural network to recognize handwritten characters and show positive effects of transfer between models for Latin and Chinese characters. Ours is the first study to apply transfer learning to neural machine translation.

There has also been work on using data from multiple language pairs in NMT to improve performance. Recently, Dong et al. (2015) showed that sharing a source encoder for one language helps performance when using different target decoders

Decoder	Hausa	Turkish	Uzbek	Urdu
NMT	16.8	11.4	10.7	5.2
Xfer	21.3	17.0	14.4	13.8
Final	24.0	18.7	16.8	14.5
SBMT	23.7	20.4	17.9	17.9

Table 2: Our method significantly improves NMT results for the translation of low-resource languages into English. Results show test-set BLEU scores. The 'NMT' row shows results without transfer, and the 'Xfer' row shows results with transfer. The 'Final' row shows BLEU after we ensemble 8 models and use unknown word replacement.

Re-scorer	SBMT Decoder			
	Hausa	Turkish	Uzbek	Urdu
None	23.7	20.4	17.9	17.9
NMT	24.5	21.4	19.5	18.2
Xfer	**24.8**	**21.8**	**19.5**	**19.1**
LM	23.6	21.1	17.9	18.2

Table 3: Our transfer method applied to re-scoring output n-best lists from the SBMT system. The first row shows the SBMT performance with no re-scoring and the other 3 rows show the performance after re-scoring with the selected model. Note: the 'LM' row shows the results when an RNN LM trained on the large English corpus was used to re-score.

for different languages. In that paper the authors showed that using this framework improves performance for low-resource languages by incorporating a mix of low-resource and high-resource languages. Firat et al. (2016) used a similar approach, employing a separate encoder for each source language, a separate decoder for each target language, and a shared attention mechanism across all languages. They then trained these components jointly across multiple different language pairs to show improvements in a lower-resource setting.

There are a few key differences between our work and theirs. One is that we are working with truly small amounts of training data. Dong et al. (2015) used a training corpus of about 8m English words for the low-resource experiments, and Firat et al. (2016) used from 2m to 4m words, while we have at most 1.8m words, and as few as 0.2m. Additionally, the aforementioned previous work used the same domain for both low-resource and high-resource languages, while in our case the datasets come from vastly different domains, which makes the task much harder and more realistic. Our approach only requires using one additional high-resource language, while the other papers used many. Our approach also allows for easy training of new low-resource languages, while Dong et al. (2015) and Firat et al. (2016) do not specify how a new language should be added to their pipeline once the models are trained. Finally, Dong et al. (2015) observe an average BLEU gain on their low-resource experiments of +1.16, and Firat et al. (2016) obtain BLEU gains of +1.8, while we see a +5.6 BLEU gain.

The transfer learning approach we use is simple and effective. We first train an NMT model on a large corpus of parallel data (e.g., French–English). We call this the *parent model*. Next, we initialize an NMT model with the already-trained parent model. This new model is then trained on a very small parallel corpus (e.g., Uzbek–English). We call this the *child model*. Rather than starting from a random position, the child model is initialized with the weights from the parent model.

A justification for this approach is that in scenarios where we have limited training data, we need a strong prior distribution over models. The parent model trained on a large amount of bilingual data can be considered an anchor point, the peak of our prior distribution in model space. When we train the child model initialized with the parent model, we fix parameters likely to be useful across tasks so that they will not be changed during child model training. In the French–English to Uzbek–English example, as a result of the initialization, the English word embeddings from the parent model are copied, but the Uzbek words are initially mapped to random French embeddings. The parameters of the English embeddings are then frozen, while the Uzbek embeddings' parameters are allowed to be modified, i.e. *fine-tuned*, during training of the child model. Freezing certain transferred parameters and fine tuning others can be considered a hard approximation to a tight prior or strong regularization applied to some of the parameter space. We also experiment with ordinary L2 regularization, but find it does not significantly improve over the parameter freezing described above.

Our method results in large BLEU increases for a variety of low resource languages. In one of the

Language Pair	Role	Train Size	Dev Size	Test Size
Spanish–English	child	2.5m	58k	59k
French–English	parent	53m	58k	59k
German–English	parent	53m	58k	59k

Table 4: Data used for a low-resource Spanish–English task. Sizes are English-side token counts.

Parent	BLEU ↑	PPL ↓
none	16.4	15.9
French–English	**31.0**	**5.8**
German–English	29.8	6.2

Table 5: For a low-resource Spanish–English task, we experiment with several choices of parent model: none, French–English, and German–English. We hypothesize that French–English is best because French and Spanish are similar.

four language pairs our NMT system using transfer beats a strong SBMT baseline. Not only do these transfer models do well on their own, they also give large gains when used for re-scoring n-best lists ($n = 1000$) from the SBMT system. Section 4 details these results.

4 Experiments

To evaluate how well our transfer method works we apply it to a variety of low-resource languages, both stand-alone and for re-scoring a strong SBMT baseline. We report large BLEU increases across the board with our transfer method.

For all of our experiments with low-resource languages we use French as the parent source language and for child source languages we use Hausa, Turkish, Uzbek, and Urdu. The target language is always English. Table 1 shows parallel training data set sizes for the child languages, where the language with the most data has only 1.8m English tokens. For comparison, our parent French–English model uses a training set with 300 million English tokens and achieves 26 BLEU on the development set. Table 1 also shows the SBMT system scores along with the NMT baselines that do not use transfer. There is a large gap between the SBMT and NMT systems when our transfer method is not used.

The SBMT system used in this paper is a string-to-tree statistical machine translation system (Galley et al., 2006; Galley et al., 2004). In this system there are two count-based 5-gram language models. One is trained on the English side of the WMT 2015 English–French dataset and the other is trained on the English side of the low-resource bitext. Additionally, the SBMT models use thousands of sparsely-occurring, lexicalized syntactic features (Chiang et al., 2009).

For our NMT system, we use development sets for Hausa, Turkish, Uzbek, and Urdu to tune the learning rate, parameter initialization range, dropout rate, and hidden state size for all the experiments. For training we use a minibatch size of 128, hidden state size of 1000, a target vocabulary size of 15K, and a source vocabulary size of 30K. The child models are trained with a dropout probability of 0.5, as in Zaremba et al. (2014). The common parent model is trained with a dropout probability of 0.2. The learning rate used for both child and parent models is 0.5 with a decay rate of 0.9 when the development perplexity does not improve. The child models are all trained for 100 epochs. We re-scale the gradient when the gradient norm of all parameters is greater than 5. The initial parameter range is [-0.08, +0.08]. We also initialize our forget-gate biases to 1 as specified by Józefowicz et al. (2015) and Gers et al. (2000). For decoding we use a beam search of width 12.

4.1 Transfer Results

The results for our transfer learning method applied to the four languages above are in Table 2. The parent models were trained on the WMT 2015 (Bojar et al., 2015) French–English corpus for 5 epochs. Our baseline NMT systems ('NMT' row) all receive a large BLEU improvement when using the transfer method (the 'Xfer' row) with an average BLEU improvement of 5.6. Additionally, when we use unknown word replacement from Luong et al. (2015b) and ensemble together 8 models (the 'Final' row) we further improve upon our BLEU scores, bringing the average BLEU improvement to 7.5. Overall our method allows the NMT system to reach competitive scores and outperform the SBMT system in one of the four language pairs.

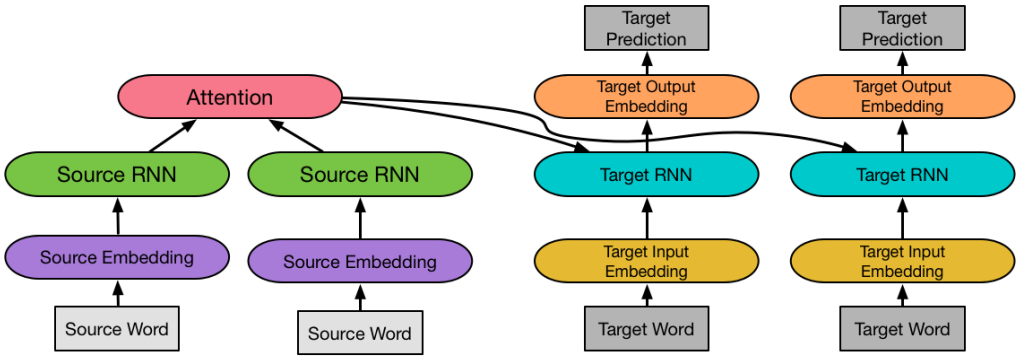


Figure 2: Our NMT model architecture, showing six blocks of parameters, in addition to source/target words and predictions. During transfer learning, we expect the source-language related blocks to change more than the target-language related blocks.

Language Pair	Parent	Train Size	BLEU ↑	PPL ↓
Uzbek–English	None	1.8m	10.7	22.4
	French–English	1.8m	**15.0 (+4.3)**	**13.9**
French′–English	None	1.8m	13.3	28.2
	French–English	1.8m	**20.0 (+6.7)**	**10.9**

Table 6: A better match between parent and child languages should improve transfer results. We devised a child language called French′, identical to French except for word spellings. We observe that French transfer learning helps French′ (13.3→20.0) more than it helps Uzbek (10.7→15.0).

4.2 Re-scoring Results

We also use the NMT model with transfer learning as a feature when re-scoring output n-best lists ($n = 1000$) from the SBMT system. Table 3 shows the results of re-scoring. We compare re-scoring with transfer NMT to re-scoring with baseline (i.e. non-transfer) NMT and to re-scoring with a neural language model. The neural language model is an LSTM RNN with 2 layers and 1000 hidden states. It has a target vocabulary of 100K and is trained using noise-contrastive estimation (Mnih and Teh, 2012; Vaswani et al., 2013; Baltescu and Blunsom, 2015; Williams et al., 2015). Additionally, it is trained using dropout with a dropout probability of 0.2 as suggested by Zaremba et al. (2014). Re-scoring with the transfer NMT model yields an improvement of 1.1–1.6 BLEU points above the strong SBMT system; we find that transfer NMT is a better re-scoring feature than baseline NMT or neural language models.

In the next section, we describe a number of additional experiments designed to help us understand the contribution of the various components of our transfer model.

5 Analysis

We analyze the effects of using different parent models, regularizing different parts of the child model, and trying different regularization techniques.

5.1 Different Parent Languages

In the above experiments we use French–English as the parent language pair. Here, we experiment with different parent languages. In this set of experiments we use Spanish–English as the child language pair. A description of the data used in this section is presented in Table 4.

Our experimental results are shown in Table 5, where we use French and German as parent languages. If we just train a model with no transfer on a small Spanish–English training set we get a BLEU score of 16.4. When using our transfer method we get Spanish–English BLEU scores of 31.0 and 29.8 via French and German parent languages, respectively. As expected, French is a better parent than German for Spanish, which could be the result of the parent language being more similar to the child language. We suspect using closely-related parent language pairs would improve overall quality.

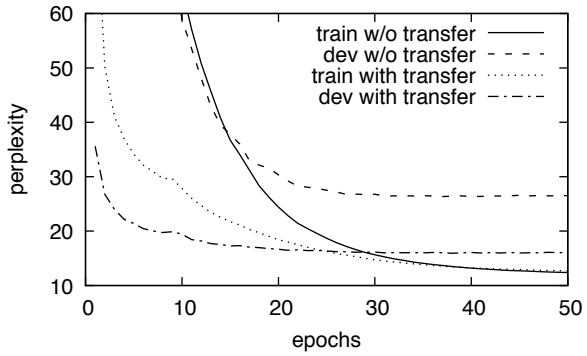


Figure 3: Uzbek–English learning curves for the NMT attention model with and without transfer learning. The training perplexity converges to a similar value in both cases. However, the development perplexity for the transfer model is significantly better.

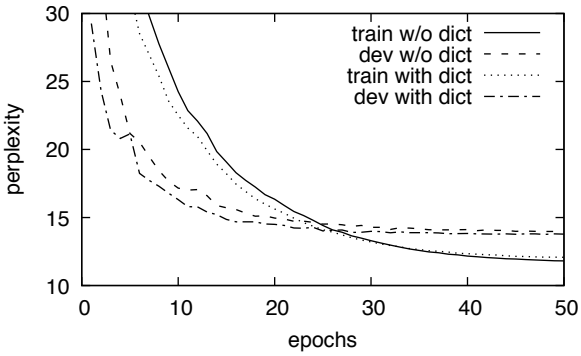


Figure 4: Uzbek–English learning curves for the transfer model with and without dictionary-based assignment of Uzbek word types to French word embeddings (from the parent model). Dictionary-based assignment enables faster improvement in early epochs. The model variants converge, showing that the unaided model is able to untangle the initial random Uzbek/French word-type mapping without help.

5.2 Effects of having Similar Parent Language

Next, we look at a best-case scenario in which the parent language is as similar as possible to the child language.

Here we devise a synthetic child language (called French′) which is exactly like French, except its vocabulary is shuffled randomly. (e.g., "internationale" is now "pomme," etc). This language, which looks unintelligible to human eyes, nevertheless has the same distributional and relational properties as actual French, i.e. the word that, prior to vocabulary reassignment, was 'roi' (king) is likely to share distributional characteristics, and hence embedding similarity, to the word that, prior to reassignment, was 'reine' (queen). French should be the ideal parent model for French′.

The results of this experiment are shown in Table 6. We get a 4.3 BLEU improvement with an unrelated parent (i.e. French–parent and Uzbek–child), but we get a 6.7 BLEU improvement with a 'closely related' parent (i.e. French–parent and French′–child). We conclude that the choice of parent model can have a strong impact on transfer models, and choosing better parents for our low-resource languages (if data for such parents can be obtained) could improve the final results.

5.3 Ablation Analysis

In all the above experiments, only the target input and output embeddings are fixed during training. In this section we analyze what happens when different parts of the model are fixed, in order to determine the scenario that yields optimal performance. Figure 2 shows a diagram of the components of a sequence-to-sequence model. Table 7 shows the effects of allowing various components of the child NMT model to be trained. We find that the optimal setting for transferring from French–English to Uzbek–English in terms of BLEU performance is to allow all of the components of the child model to be trained except for the input and output target embeddings.

Even though we use this setting for our main experiments, the optimum setting is likely to be language- and corpus-dependent. For Turkish, experiments show that freezing attention parameters as well gives slightly better results. For parent-child models with closely related languages we expect freezing, or strongly regularizing, more components of the model to give better results.

5.4 Learning Curve

In Figure 3 we plot learning curves for both a transfer and a non-transfer model on training and development sets. We see that the final training set perplexities for both the transfer and non-transfer model are very similar, but the development set perplexity for the transfer model is much better.

The fact that the two models start from and converge to very different points, yet have similar training set performances, indicates that our architecture

Source Embeddings	Source RNN	Target RNN	Attention	Target Input Embeddings	Target Output Embeddings	Dev BLEU ↑	Dev PPL ↓
🔒	🔒	🔒	🔒	🔒	🔒	0.0	112.6
🔓	🔒	🔒	🔒	🔒	🔒	7.7	24.7
🔓	🔓	🔒	🔒	🔒	🔒	11.8	17.0
🔓	🔓	🔓	🔒	🔒	🔒	14.2	14.5
🔓	🔓	🔓	🔓	🔒	🔒	**15.0**	13.9
🔓	🔓	🔓	🔓	🔓	🔒	14.7	**13.8**
🔓	🔓	🔓	🔓	🔓	🔓	13.7	14.4

Table 7: Starting with the parent French–English model (BLEU =24.4, PPL=6.2), we randomly assign Uzbek word types to French word embeddings, freeze various parameters of the neural network model (🔒), and allow Uzbek–English (child model) training to modify other parts (🔓). The table shows how Uzbek–English BLEU and perplexity vary as we allow more parameters to be re-trained.

and training algorithm are able to reach a good minimum of the training objective regardless of the initialization. However, the training objective seems to have a large basin of models with similar performance and not all of them generalize well to the development set. The transfer model, starting with and staying close to a point known to perform well on a related task, is guided to a final point in the weight space that generalizes to the development set much better.

5.5 Dictionary Initialization

Using the transfer method, we always initialize input language embeddings for the child model with randomly-assigned embeddings from the parent (which has a different input language). A smarter method might be to initialize child embeddings with similar parent embeddings, where similarity is measured by word-to-word t-table probabilities. To get these probabilities we compose Uzbek–English and English–French t-tables obtained from the Berkeley Aligner (Liang et al., 2006). We see from Figure 4 that this dictionary-based assignment results in faster improvement in the early part of the training. However the final performance is similar to our standard model, indicating that the training is able to untangle the dictionary permutation introduced by randomly-assigned embeddings.

5.6 Different Parent Models

In the above experiments, we use a parent model trained on a large French–English corpus. One might hypothesize that our gains come from exploiting the English half of the corpus as an additional language model resource. Therefore, we explore transfer learning for the child model with parent models that only use the English side of the French–English corpus. We consider the following parent models in our ablative transfer learning scenarios:

- A true translation model (French–English Parent)
- A word-for-word English copying model (English–English Parent)
- A model that unpermutes scrambled English (EngPerm–English Parent)
- (The parameters of) an RNN language model (LM Xfer)

Transfer Model	BLEU ↑	PPL ↓
None	10.7	22.4
French–English Parent	**14.4**	**14.3**
English–English Parent	5.3	55.8
EngPerm–English Parent	10.8	20.4
LM Xfer	12.9	16.3

Table 8: Transfer for Uzbek–English NMT with parent models trained only on English data. The English–English parent learns to copy English sentences, and the EngPerm–English learns to un-permute scrambled English sentences. The LM is a 2-layer LSTM RNN language model.

The results, in Table 8, show that transfer learning does not simply import an English language model, but makes use of translation parameters learned from the parent's large bilingual text.

6 Conclusion

Overall, our transfer method improves NMT scores on low-resource languages by a large margin and allows our transfer NMT system to come close to the performance of a very strong SBMT system, even exceeding its performance on Hausa–English. In addition, we consistently and significantly improve state-of-the-art SBMT systems on low-resource languages when the transfer NMT system is used for rescoring. Our experiments suggest that there is still room for improvement in selecting parent languages that are more similar to child languages, provided data for such parents can be found.

Acknowledgments

This work was supported by ARL/ARO (W911NF-10-1-0533), DARPA (HR0011-15-C-0115), and the Scientific and Technological Research Council of Turkey (TÜBİTAK) (grants 114E628 and 215E201).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

P. Baltescu and P. Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *Proc. HLT-NAACL*.

Y. Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. *JMLR*, 27.

O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, M. Huck, C. Hokamp, P. Koehn, V. Logacheva, C. Monz, M. Negri, M. Post, C. Scarton, L. Specia, and M. Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proc. WMT*.

M. A. Castaño and F. Casacuberta. 1997. A connectionist approach to machine translation. In *Proc. Eurospeech*.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *Proc. HLT-NAACL*.

D. C. Cireşan, U. Meier, and J. Schmidhuber. 2012. Transfer learning for Latin and Chinese characters with deep neural networks. In *Proc. IJCNN*.

D. Dong, H. Wu, W. He, D. Yu, and H. Wang. 2015. Multi-task learning for multiple language translation. In *Proc. ACL-IJCNLP*.

O. Firat, K. Cho, and Y. Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proc. NAACL-HLT*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL-COLING*.

F. A. Gers, J. Schmidhuber, and F. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10).

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8).

R. Józefowicz, W. Zaremba, and I. Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. ICML*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. HLT-NAACL*.

M. Luong, H. Pham, and C. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. ACL*.

A. Mnih and Y. W. Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. ICML*.

R. Neco and M. Forcada. 1997. Asynchronous translations with recurrent neural nets. In *Proc. International Conference on Neural Networks*.

S. J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10).

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*.

L. Torrey and J. Shavlik. 2009. Transfer learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, editors, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global.

A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proc. EMNLP*.

D. Wang and T. Fang Zheng. 2015. Transfer learning for speech and language processing. *CoRR*, abs/1511.06066.

P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10).

W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson. 2015. Scaling recurrent neural network language models. In *Proc. ICASSP*.

W. Zaremba, I. Sutskever, and O. Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

MixKMeans: Clustering Question-Answer Archives

Deepak P
Centre for Data Sciences and Scalable Computing
Queen's University Belfast, UK
deepaksp@acm.org

Abstract

Community-driven Question Answering (CQA) systems that crowdsource experiential information in the form of questions and answers and have accumulated valuable reusable knowledge. Clustering of QA datasets from CQA systems provides a means of organizing the content to ease tasks such as manual curation and tagging. In this paper, we present a clustering method that exploits the two-part question-answer structure in QA datasets to improve clustering quality. Our method, *MixKMeans*, composes question and answer space similarities in a way that the space on which the match is higher is allowed to dominate. This construction is motivated by our observation that semantic similarity between question-answer data (QAs) could get localized in either space. We empirically evaluate our method on a variety of real-world labeled datasets. Our results indicate that our method significantly outperforms state-of-the-art clustering methods for the task of clustering question-answer archives.

1 Introduction

Community-based Question Answering (CQA) systems such as Yahoo! Answers[1], StackOverflow[2] and Baidu Zhidao[3] have become dependable sources of knowledge to solve common user problems. Unlike factoid question answering[4], CQA systems focus on crowdsourcing *how* and *why* questions and their answers. As is the case with any system where content is generated by web users, the generated content would be of varying quality, reliability, readability and abstraction. Thus, manual curation of such datasets is inevitable to weed out low quality and duplicate content to ensure user satisfaction. A natural way to aid manual curation of such broad-based CQA archives is to employ clustering so that semantically related QAs are grouped together; this would help organize the corpus in a way that experts engaged in manual curation be assigned specific clusters relating to areas of their expertise. Clustering also provides a platform to enable tagging the QA dataset; cluster topics could be used as tags, or other QAs in the same cluster could be tagged as being *related* to a QA. The fundamental difference between CQA archives and general text document collections is the existence of a two-part structure in QAs and the difference in lexical "character" between the question and answer parts. This *lexical chasm (i.e., gap)* (Berger et al., 2000) between question and answer parts has been a subject of much study, especially, in the context of improving QA retrieval. In this paper, we consider using the two-part structure in QAs for clustering CQA datasets.

Motivating Example: Table 1 lists four example QAs from the context of a CQA system focused on addressing myriad technical issues. These QAs have been tagged in the table with a manually identified root-cause to aid understanding; the root-cause is not part of the CQA data per se. $QA1$ and $QA2$ are seen to address related issues pertaining to routers, whereas $QA3$ and $QA4$ are focused on the same nar-

[1] http://answers.yahoo.com
[2] http://www.stackoverflow.com
[3] http://en.wikipedia.org/en/Baidu_Knows
[4] e.g., http://trec.nist.gov/data/qa.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1576–1585,
Austin, Texas, November 1-5, 2016.

row issue dealing with java libraries. Since $QA1$ and $QA2$ address different problems, they may not be expected to be part of the same cluster in fine-grained clusterings. On the other hand, the solutions suggested in $QA3$ and $QA4$ are distinct and different legitimate solutions to the *same* problem cause. Thus, from a semantics perspective, it is intuitive that $QA3$ and $QA4$ should be part of the same cluster in any clustering of the CQA dataset to aid actioning on them together; a human expert might decide to merge the question parts and tag one of the answers as an *alternative* answer. Let us now examine the lexical relatedness between the pairs as illustrated in Table 2. State-of-the-art text similarity measures that quantify word overlaps are likely to judge $QA1$ and $QA2$ to be having a *medium* similarity when either the question-part or the answer-part are considered. For the pair $(QA3, QA4)$, the question-part similarity would be judged to be *high* and the answer-part similarity as *low*. Thus, the high similarity between the root-causes of $QA3$ and $QA4$ manifest primarily in their question-parts. Analogously, we observed that some QAs involving the same root-cause lead to high answer-part similarity despite poor question-part similarity. This is especially true in cases involving suggestion of the same sequence of solution steps despite the question-part being divergent due to focusing on different symptoms of the same complex problem. From these observations, we posit that high similarities on *either* the question-space or answer-space is indicative of semantic relatedness. Any clustering method that uses a sum, average or weighted sum aggregation function to arrive at pair-wise similarities, such as a K-Means clustering that treats the collated QA as a single document, would intuitively be unable to heed to such differential manifestation of semantic similarities across the two parts.

Our Contributions: We address the problem of harnessing the two-part structure in QA pairs to improve clustering of CQA data. Based on our observations on CQA data such as those illustrated in the example, we propose a clustering approach, *MixK-Means*, that composes similarities (dissimilarities) in the question and answer spaces using a max (min) operator style aggregation. Through abundant empirical analysis on real-world CQA data, we illustrate that our method outperforms the state-of-the-art approaches for the task of CQA clustering.

2 Related Work

To enable position our work in the context of existing literature, we now summarize prior work along three related directions, viz., (1) processing of CQA datasets, (2) multi-modal data clustering, and (3) K-Means extensions.

Processing CQA Datasets: Most work on processing CQA data has been in the realm of retrieval, where the task addressed is to leverage CQA datasets to aid answering new questions. These start with a new question and find one of (i) related questions (Zhou et al., 2015), (ii) potentially usable answers (Shtok et al., 2012), or (iii) related QAs (Xue et al., 2008). Different methods differ in the technique employed to overcome the *lexical chasm*, with statistical translation models (Brown et al., 1993) that model word-level correlations between questions and answers being the most popular tool for the same. Usage of topic models (e.g., (Cai et al., 2011)) and combining evidence from topic and translation models (Zhou et al., 2015) have also met with success. The usage of deep-learning methods such as deep belief networks (Wang et al., 2011) and auto-encoders (Zhou et al., 2016) have also been explored for QA retrieval. While the problem of estimating the relevance of a QA to address a new question is related to the problem of estimating similarities between QAs to aid clustering, the latter problem is different in that both question and answer parts are available at either side. In fact, our problem, CQA clustering, has been largely unexplored among literature in CQA data processing. In the interest of benchmarking our work against techniques from the CQA processing community, we consider the correlated latent representation learnt by the recent auto-encoder based neural network (AENN) method (Zhou et al., 2016) as input to K-Means, and empirically validate our technique against the AENN+K-Means combination (referred to as AENN, for short) in our experimental study. Outside the task of retrieval, there has been work on getting QAs from experience reports (Deepak et al., 2012) and discussion forums (P and Visweswariah, 2014). Conversational transcripts from contact centres, as outlined in (Kummamuru et al., 2009), form

#	QA	Cause
QA1	**Q:** My internet connection is not working, my router shows the "Internet" led blinking in red.	Router Authentication Issue
	A: Please go to the router login page and re-login with broadband credentials; click "connect" and you should be on the internet.	
QA2	**Q:** My internet connection is not working, only the power led is lit in the router.	Router Loose Connection
	A: Check whether the router login page is loading. Else, the broadband cable may not be connected properly.	
QA3	**Q:** My Java app is picking up the old dojo 0.4.4 libraries though I have a newer version.	Multiple Libraries in Classpath
	A: Search for dojo 0.4.4 in Windows, and delete off the folder, and it should automatically start using the newer version.	
QA4	**Q:** My java application is not picking up the new dojo 1.11.1 libraries that I just installed.	Multiple Libraries in Classpath
	A: Update the java classpath variable to exclude the path to the earlier version, and add the path to the new version.	

Table 1: Example CQA Data

QA Pair		Part	Lexical Similarity
QA1	QA2	Question	Medium
QA1	QA2	Answer	Medium
QA3	QA4	Question	High
QA3	QA4	Answer	Low

Table 2: Similarity Analysis of QAs from Table 1

another rich source of QA data, but need careful segmentation due to interleaving of question and answer parts.

Multi-modal Data Clustering: The problem of clustering CQA data is an instance of the general problem of clustering multi-modal (aka multi-relational, multi-view or heterogeneous) data when the question and answer parts are seen as instantiations of the same root cause, but in question and answer 'modalities'. Clustering multi-modal data has been explored well in the context of multi-media data clustering where each data element comes in multi-modal form such as *[image, caption]* pairs or *[audio, text]* pairs. The pioneering work in this field adapted markov random fields (Bekkerman and Jeon, 2007) to generate separate clusterings for each modality. Later approaches are closer to our task of generating a unified clustering across modalities; they work by learning a unified latent space embedding of the dataset, followed by usage of K-Means clustering (MacQueen and others, 1967). Eigen-decomposition (Petkos et al., 2012), spectral methods (Blaschko and Lampert, 2008) and canonical correlation analysis (Jin et al., 2015) have been exploited for learning the latent space prior to the clustering step. A recent work (Meng et al., 2014) proposes a single-pass leader-clustering[5] style formulation called GHF-ART to progressively assign data objects to clusters. Unlike most other methods that assume that vector representations are obtained from general multimedia data, the authors of GHF-ART lay out how text data be pre-processed for usage in GHF-ART, making it an appropriate method for usage in our setting. Accordingly, we will use GHF-ART as a baseline method for our experimental study.

K-Means Extensions: The method that we propose in this paper, *MixKMeans*, draws generous inspiration from the classical K-Means clustering formulation (MacQueen and others, 1967). There have been numerous extensions to the basic K-Means formulation over the last many decades; many such extensions have been covered in focused surveys (Steinley, 2006; Jain, 2010). Of particular interest in our scenario are those relating to usage of varying (dis)similarity measures. (Patel and Mehta, 2012) discuss the usage of various popular distance measures within the K-Means framework. The point-symmetry distance, where the distance between an object and the cluster prototype is determined using other objects' information, has been explored (Su and Chou, 2001) for usage within K-Means for face recognition applications. Another work (Visalakshi and Suguna, 2009) suggests the computation of the aggregate distance as a fraction of the distance

[5] https://cran.r-project.org/web/packages/leaderCluster/index.html

along the closest attribute to that along the farthest attribute. Despite the plethora of work around extending K-Means to work with a variety of methods to aggregate distances across attributes, we have not come across previous work composing distances at the level of attribute sets (or modalities) like we will do in this work.

3 Problem Definition

Let $\mathcal{D} = \{(q_1, a_1), \ldots, (q_n, a_n)\}$ be a dataset of QAs from a CQA archive where each answer a_i was posted in response to the corresponding question q_i. The CQA clustering problem is the task of partitioning $\mathcal{D}$ into k clusters $\mathcal{C} = \{C_1, \ldots, C_k\}$ where $\cup_i C_i = \mathcal{D}$ and $\forall(i,j), i \neq j \Rightarrow C_i \cap C_j = \phi$ (disjointedness) hold such that similar QAs are grouped into the same cluster and dissimilar QAs are assigned to different clusters. The key aspect that differentiates the CQA clustering problem from general clustering of relational data is the opportunity to leverage the specifics of the CQA data, such as the two-part structure, to model the similarity measure that would drive the clustering.

Evaluation: The quality of a clustering method may be quantified by assessing how well the clustering it produces, i.e., $\mathcal{C}$, reflects the semantic similarities between QAs in $\mathcal{D}$. Given a QA $(q_i, a_i) \in \mathcal{D}$, the other QAs that share the same cluster may be thought of as the *result* set, i.e., the set of *related* QAs according to $\mathcal{C}$. In a labeled dataset such as CQADupStack (Hoogeveen et al., 2015) where related QA pairs have been manually identified for each (q_i, a_i), the quality of the results set may be assessed by contrasting against the labeled set using a standard metric such as F-score[6]. These QA-specific F-scores are then aggregated across the QAs in $\mathcal{D}$ to arrive at a single quality measure for the clustering. We will use such aggregated dataset-level F-scores as our primary evaluation measure. It may be noted that the *related* labellings may not be "clustering-friendly"; for example, there may not exist any k-clustering with no *related* labels going across clusters. Additionally, we observed that not all related QAs were labeled to be related in the CQADupStack dataset. The dataset owners confirm the problem of missing labelings in a very recent study (Hoogeveen et al., 2016). It is conceivable that only a few potential results were manually inspected to inject labellings. Thus, while the relative trends on F-score offer insights, the absolute F-scores may only be treated as a loose lower bounds.

[6] https://en.wikipedia.org/wiki/F1_score

4 *MixKMeans*: Our Method

We now describe the key details of our proposed technique, *MixKMeans*. The name is motivated by the flexibility that is built into the method to *mix* (dis)similarities across question and answer spaces in a formulation that derives inspiration from the classical K-Means algorithm (MacQueen and others, 1967). Throughout this formulation, we represent question and answer parts of QAs by their respective tf-idf vectors. We start with our objective function and move on to the iterative optimization.

4.1 Objective Function

Guided by our observation from Section 1 that the space in which a pair of QAs are more similar should hold sway in determining their overall match, we outline a penalty function for a clustering $\mathcal{C}$:

$$\mathcal{O}^* = \sum_{C \in \mathcal{C}} \sum_{(q,a) \in C} min\{w_q \, d(q, C.\mu.q), w_a \, d(a, C.\mu.a)\} \quad (1)$$

where $C.\mu = (C.\mu.q, C.\mu.a)$ is a prototypical QA vector for cluster C and the parameter pair $[w_q, w_a]$ control the relative weighting between question and answer parts. $d(.,.)$ is a dissimilarity function modeled as a simple sum of squares of element-wise differences between vector entries, i.e., $d(x,y) = \sum_i (x[i] - y[i])^2$.

Intuitively, $\mathcal{O}^*$ sums up the distance between each QA in $\mathcal{D}$ and the prototypical QA vector of the cluster to which it is assigned to, making it a penalty function. Since we use dissimilarities that are inversely related to similarities, the min function captures the idea that the aggregate (dis)similarity be estimated according to the measure in the best matching space. For optimization convenience, we replace the min construction by a differentiable approximation to get a modified objective function:

$$\mathcal{O} = \sum_{C \in \mathcal{C}} \sum_{(q,a) \in C} \Bigg(\Big(w_q d(q, C.\mu.q) \Big)^x + \Big(w_a d(a, C.\mu.a) \Big)^x \Bigg)^{\frac{1}{x}} \quad (2)$$

where x is a reasonably high negative value or $x \to -\infty$. This is used since $(a^x + b^x)^{1/x}$ approximates $min\{a, b\}$ for high negative values of x. It is worth noting that the opposite effect (i.e., max approximation) is achieved when $x \to \infty$ for usage in scenarios where a max combination is desirable. The remainder of the steps are applicable for positive values of x too.

4.2 Optimization Approach

There are two sets of variables in Equation 2, viz., cluster assignments of QAs in $\mathcal{D}$ and the cluster prototypes ($C.\mu$s). We optimize for each set of variables alternatively, much like in the EM-steps used in the classical K-Means algorithm.

4.2.1 Estimating Cluster Memberships

The cluster membership estimation directly falls out from the objective function and the current estimates of cluster prototypes since $\mathcal{O}$ (Equation 2) involves an instance-specific term for each QA. We will simply assign each QA to the cluster such that the respective instance-specific term is minimized:

$$Cluster((q,a)) = \arg\min_{C \in \mathcal{C}} \Big(d^x_{Q+A}((q,a), C.\mu) \Big)^{\frac{1}{x}} \quad (3)$$

$d^x_{Q+A}(.,.)$ is a short-hand for composite distance, composed of two terms (which we will denote as $d^x_Q(.,.)$ and $d^x_A(.,.)$ respectively):

$$d^x_{Q+A}((q,a), C.\mu^\circ) = \big(w_q \times d(q, C.\mu^\circ.q) \big)^x + \big(w_a \times d(a, C.\mu^\circ.a) \big)^x \quad (4)$$

4.2.2 Estimating Cluster Prototypes

We now estimate the cluster prototype in element-wise fashion. Consider a particular element in the $C.\mu.q$ vector, $C.\mu.q[i]$; computing the partial derivative and simplifying:

$$\frac{\partial \mathcal{O}}{\partial C.\mu.q[i]} = \sum_{(q,a) \in C} \Bigg[-2 \Big(d^x_{Q+A}((q,a), C.\mu) \Big)^{\frac{1-x}{x}} d^{x-1}_Q((q,a), C.\mu)\ w_q\ (q[i] - C.\mu.q[i]) \Bigg] \quad (5)$$

Equating the first derivative to zero and solving for $C.\mu.q[i]$ gets us to the following form:

$$C.\mu.q[i] = \frac{\sum_{(q,a) \in C} q[i] \Bigg[\Big(d^x_{Q+A}((q,a), C.\mu^\circ) \Big)^{\frac{1-x}{x}} d^{x-1}_Q((q,a), C.\mu^\circ) \Bigg]}{\sum_{(q,a) \in C} \Bigg[\Big(d^x_{Q+A}((q,a), C.\mu^\circ) \Big)^{\frac{1-x}{x}} d^{x-1}_Q((q,a), C.\mu^\circ) \Bigg]} \quad (6)$$

where $C.\mu^\circ$ is used to indicate the estimate of $C.\mu$ from the previous iteration. The corresponding estimation for $C.\mu.a[i]$ is:

$$C.\mu.a[i] = \frac{\sum_{(q,a) \in C} a[i] \Bigg[\Big(d^x_{Q+A}((q,a), C.\mu^\circ) \Big)^{\frac{1-x}{x}} d^{x-1}_A((q,a), C.\mu^\circ) \Bigg]}{\sum_{(q,a) \in C} \Bigg[\Big(d^x_{Q+A}((q,a), C.\mu^\circ) \Big)^{\frac{1-x}{x}} d^{x-1}_A((q,a), C.\mu^\circ) \Bigg]} \quad (7)$$

Equations 6 and 7 form the cluster prototype estimation steps of our method. It may be noted that for the choice of parameters ($x = 1, w_q = w_a$), either equations reduce it to the usual centroid estimation process for K-Means (since the terms within $[\ldots]$ reduce to 1.0), as intuitively expected. Thus, the modified formulation generalizes K-Means by allowing to weigh each element differently, the weight being modeled as a product two components:

- First component involves $d^x_{Q+A}(.,.)$ and is a function of the composite distance of (q,a) to the cluster prototype.
- Second component involves one of $d^{x-1}_Q(.,.)$ or $d^{x-1}_A(.,.)$ and is a function of the respective space (Q or A) to which the specific vector element belongs.

Alg. 1 ***MixKMeans***

Input. Dataset $\mathcal{D}$, number of clusters k
Hyper-parameters: x, w_q, w_a
Output. Clustering $\mathcal{C}$

1: *Initialize* $C.\mu$s *using data points from* $\mathcal{D}$
2: **while** *not yet converged* **do**
3: $\quad \forall (q,a) \in \mathcal{D}$, *assign cluster using Eq. 3*
4: $\quad \forall C \in \mathcal{C}$, *estimate* $C.\mu$ *using Eq. 6 & 7*
5: **end while**
6: *Return current clustering assignments as* $\mathcal{C}$

4.3 *MixKMeans*: The Algorithm

Having outlined the various steps, we are now ready to present the overall $MixKMeans$ algorithm in Algorithm 1. As the pseudo-code indicates, the cluster assignment and prototype estimation steps are run in a loop until the clustering converges. Additionally, we terminate after a threshold number of iterations even if the clustering does not converge by then; we set the threshold to 10.

Initialization: In the initialization step, we initialize the first cluster prototype using a random QA from $\mathcal{D}$. Each of the next cluster prototypes are initialized using the QA that has the highest sum of distances to all pre-chosen cluster prototypes, distance computed using $(d^x_{Q+A}(.,.))^{1/x}$. This is inspired by previous work on spreading out the cluster centroids (Arthur and Vassilvitskii, 2007) in K-Means initialization.

Hyperparameters: The algorithm has three hyper-parameters, viz., the exponentiation parameter x and the weight parameters w_q and w_a. As outlined in Sec. 4.1, x should be a negative value; we observed that any value beyond -3.0 does not make any significant differences to the final clustering (while higher absolute values for the exponent pose an underflow risk) and thus use $x = -3.0$ consistently. For the weights, we set $w_q = 0.2$ and $w_a = 0.8$. Due to the min-formulation in the objective function, a lower weight increases the influence of the respective space. Thus, we let our composed similarities be influenced more by the question-space similarities as in previous work (Xue et al., 2008).

4.4 Generalizing *MixKMeans*

Since the question and answer spaces are neatly segregated into different terms in the parameter update equations, ***MixKMeans*** is easily generalizable to work with more than two spaces. Consider the set of spaces to be $\mathcal{M} = \{\ldots, M, \ldots\}$ and that each object, $X \in \mathcal{D}$ be represented by an $|\mathcal{M}|$ tuple; now, the modified update equations are as follows:

$$Cluster(X) = \arg\min_{C \in \mathcal{C}} \left(d_{\sum_{M \in \mathcal{M}} M}(X, C.\mu) \right)^{\frac{1}{x}} \tag{8}$$

$$C.\mu.M[i] = \frac{\sum_{X \in C} X.M[i] \left[\left(d^x_{\sum_{M \in \mathcal{M}} M}(X, C.\mu^\circ) \right)^{\frac{1-x}{x}} d^{x-1}_M(X, C.\mu^\circ) \right]}{\sum_{X \in C} \left[\left(d^x_{\sum_{M \in \mathcal{M}} M}(X, C.\mu^\circ) \right)^{\frac{1-x}{x}} d^{x-1}_M(X, C.\mu^\circ) \right]} \tag{9}$$

where the somewhat awkward notation $d^x_{\sum_{M \in \mathcal{M}} M}(.,.)$ denotes the direct generalization of $d^x_{Q+A}(.,.)$ to cover all spaces in $\mathcal{M}$.

A simple modeling extension to use the generalized *MixKMeans* in the CQA setting is to consider the question title and question description as two separate spaces instead of using a single question space, increasing the total number of spaces to three; such a split of the question-part was used in (Qiu et al., 2013). In certain cases, one might want to use spaces that are of questionable quality due to reasons such as sparsity (e.g., set of tags associated with a question) and reliability (e.g., comments attached to a QA that could be noisy). The best way to leverage such spaces would be to include it in $\mathcal{M}$ for the modeling, but use a high weight for w_M; due to the min-style construction in the objective function, that setting will ensure that that space is called into play only when (a) signals from other spaces are not very strong, and (b) the signal from the space in question is very strong.

5 Experimental Evaluation

5.1 Datasets, Baselines and Setup

Datasets: We use the recently released data col-

lection, CQADupStack (Hoogeveen et al., 2015), for our experimental evaluation. Unlike most other datasets, this has each QA labeled with a set of *related* QAs, as alluded to in Section 3; this makes automated evaluation feasible in lieu of a laborious user study. We use the *android*, *gis*, *stats* and *physics* datasets from the CQADupStack collection, with our choice of datasets motivated by dataset size. These datasets comprise 2193, 3726, 4004 and 5044 QAs respectively.

Baselines: We use two baselines from literature in our study, (i) AENN (Zhou et al., 2016), (ii) GHF-ART (Meng et al., 2014). AENN, as alluded to in Section 2, refers to the K-Means clustering in the latent space learnt by correlated auto-encoders across the Q-A subspaces. AENN requires triplets of the form [question, answer, other answer] in the training phase; we populate the *other answer* part by the answer to a *related* question from the dataset (it may be noted that this is advantageous to AENN since it gets to *'see'* some *related* labelings in the training, whereas other methods can't). GHF-ART is the state-of-the-art multi-modal clustering approach that is targeted towards scenarios that involve a text modality. Unlike typical clustering algorithms that can generate a pre-specified (k) number of clusters, the number of clusters in the GHF-ART output is controlled by a *vigilance parameter*, ρ. Lower values of ρ result in smaller number of clusters and vice versa. A third intuitive baseline is the degenerate $x = 1$ instantiation of *MixKMeans*, which we will denote as $X1$. We are interested in evaluating the improvement achieved by *MixKMeans* over the best possible instantiation of $X1$; towards that, for every setting denoted by the combination $[dataset, k]$, we do a search over possible positive values of w_q and w_a within the locus of the line $w_q + w_a = 1$. It may be noted that this search space includes simple QA clustering using K-Means, being the case where $w_q = w_a = 0.5$. We collect the best result of $X1$ from across the grid-search for each setting. This approach, which we will denote as $X1^*$, while impractical in real scenarios due to usage of labeled data, gives an empirical upper bound of the accuracy of $X1$.

Experimental Setup: We use a latent space dimensionality of 2000 for AENN since we observed an accuracy peak around that value, and set GHF-ART parameters to their recommended values from the paper. For *MixKMeans*, we use tf-idf representation and set $(x = -3.0, w_q = 0.2, w_a = 0.8)$ as discussed earlier (Section 4.3). We use the F-score[7] measure to experimentally compare the approaches. The F-score is computed using the *related*

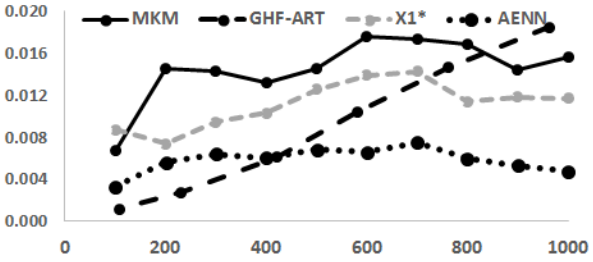


Figure 1: Android: F-Score (Y-Axis) vs. k

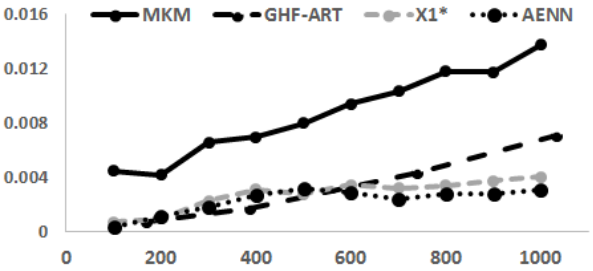


Figure 2: GIS: F-Score (Y-Axis) vs. k

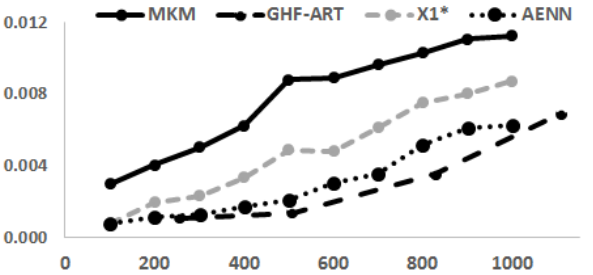


Figure 3: Stats: F-Score (Y-Axis) vs. k

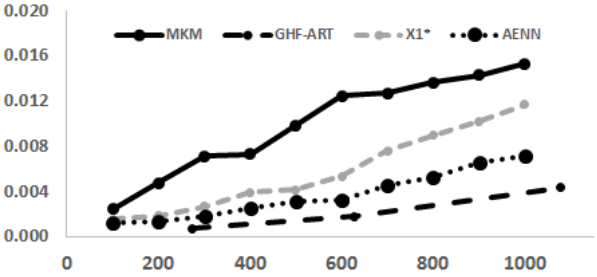


Figure 4: Physics: F-Score (Y-Axis) vs. k

[7] https://en.wikipedia.org/wiki/F1_score

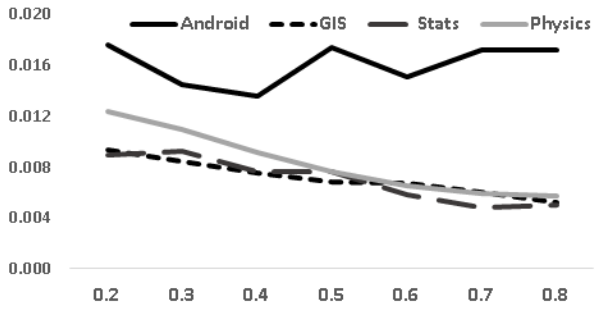


Figure 5: *MixKMeans*: F-Score (Y-Axis) vs. w_q at $k = 600$

labellings in the CQADupStack data, in a manner as described in Section 3. As pointed out therein, due to the sparse labellings, the F-score may only be regarded as a loose lower bound of their real values on a fully-labeled dataset.

5.2 Comparative Analysis

The results of the comparative analysis benchmarking our approach *MixKMeans* (MKM) against baselines $X1^*$, AENN and GHF-ART for the various datasets appear in Fig 1 (Android), Fig 2 (GIS), Fig 3 (Stats) and Fig 4 (Physics). Each of the trend-lines plot the F-Score against varying number of clusters in the output (k) in the range $\{100, 1000\}$. Since the number of clusters cannot be pre-specified for GHF-ART directly, we varied its ρ parameter to generate varying number of clusters to generate a trend-line that can be compared against *MixKMeans*, AENN and $X1^*$ directly. It may be noted that F-score is generally seen to increase when the clustering is more fine-grained (i.e., high k); this is an artifact of the sparse labeling that causes large clusters to have very low precision, causing precision and recall to diverge at low k, thus reducing the F-score. In most cases, *MixKMeans* is seen to outperform the other methods by scoring significantly higher in the F-Score, illustrating the superiority of our method. A notable exception appears in the higher values of k in the android dataset where GHF-ART quickly catches up and surpasses the others; however, it may be noted that $k \approx 1000$ is an extremely fine-grained clustering for the android dataset with 2193 QAs, and is thus not a very useful setting in practice. On the average, *MixKMeans* achieves an F-score improvement of between $30 - 100\%$ over the other methods.

5.3 *MixKMeans* Parameter Analysis

We now analyze the F-score trends of *MixKMeans* against varying values of the weight parameters. Since the relative weighting between w_q and w_a is what matters (simply scaling them both up by the same multiplier does not make any difference due to the construction of the objective), we set $w_a = (1.0 - w_q)$ and do the analysis for varying values of w_q keeping $k = 600$. As may be observed from the results in Figure 5, *MixKMeans* was seen to peak around $w_q = 0.2$-0.5 while degrading gracefully towards higher values of w_q. The android dataset, perhaps due to its relatively small size, records a different behavior as compared to the other trend-lines. Similar trends were observed for other values of k, indicating *MixKMeans* is not highly sensitive to the parameter and degrades gracefully outside the peak.

6 Conclusions and Future Work

We considered the problem of clustering question-answer archives from CQA systems. Clustering, we observed, helps in organizing CQA archival data for purposes such as manual curation and tagging. We motivated, by way of examples, as to why similarities along question and answer spaces be better composed using methods other than simple sum or average type aggregation. In particular, we noted that there are potentially different ways to answer questions pertaining to the same root-cause, mitigating the manifestation of the inherent root-cause similarity in the answer-space. Analogously, a sophisticated root-cause could be narrated differently by different people in the question part, while eliciting very similar answers. In short, we observe that legitimate reasons cause manifestation of semantic similarity between QAs to be localized on to one of the spaces. Accordingly, we propose a clustering method for QA archives, *MixKMeans*, that can heed to high similarities in either spaces to drive the clustering. *MixKMeans* works by iteratively optimizing the two sets of parameters, cluster assignments and cluster prototype learning, in an approach inspired by the classical K-Means algorithm. We empirically benchmark our method against current methods on multiple real-world datasets. Our experimental study illustrates that our method is able to significantly outperform other methods, establishing

MixKMeans as the preferred method for the task of clustering CQA datasets.

Future Work: As discussed in Section 4.4, *MixKMeans* is eminently generalizable to beyond two spaces. Considering the usage of other kinds of data (e.g., tags, comments) as additional "spaces" to extend the CQA clustering problem is an interesting direction for future work. The applicability of *MixKMeans* and it's *max* variant (i.e., with $x > 0$) for other kinds of multi-modal clustering problems from domains such as multimedia processing is worth exploring. The extension of the formulation to include a weight learning step may be appropriate for scenarios where prior information on the relative importance of the different spaces is not available. It is easy to observe that *MixKMeans* is prone to local optima issues; this makes devising better initialization strategies another potential direction. Yet another direction of interest is to make MixKMeans clusters interpretable, potentially by augmenting each cluster with word-level rules as used in earlier work on partitional document clustering (Balachandran et al., 2012).

References

David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

Vipin Balachandran, Deepak P, and Deepak Khemani. 2012. Interpretable and reconfigurable clustering of document datasets by deriving word-based rules. *Knowl. Inf. Syst.*, 32(3):475–503.

Ron Bekkerman and Jiwoon Jeon. 2007. Multi-modal clustering for multimedia collections. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–199. ACM.

Matthew B Blaschko and Christoph H Lampert. 2008. Correlational spectral clustering. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

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

Li Cai, Guangyou Zhou, Kang Liu, and Jun Zhao. 2011. Learning the latent topics for question retrieval in community qa. In *IJCNLP*, volume 11, pages 273–281.

P. Deepak, Karthik Visweswariah, Nirmalie Wiratunga, and Sadiq Sani. 2012. Two-part segmentation of text documents. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 793–802.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*, page 3. ACM.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2016. Cqadupstack: Gold or silver?

Anil K Jain. 2010. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

Cheng Jin, Wenhui Mao, Ruiqi Zhang, Yuejie Zhang, and Xiangyang Xue. 2015. Cross-modal image clustering via canonical correlation analysis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Krishna Kummamuru, Deepak Padmanabhan, Shourya Roy, and L Venkata Subramaniam. 2009. Unsupervised segmentation of conversational transcripts. *Statistical Analysis and Data Mining*, 2(4):231–245.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Lei Meng, Ah-Hwee Tan, and Dong Xu. 2014. Semi-supervised heterogeneous fusion for multimedia data co-clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 26(9):2293–2306.

Deepak P and Karthik Visweswariah. 2014. Unsupervised solution post identification from discussion forums. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 155–164.

Vaishali R. Patel and Rupa G. Mehta, 2012. *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011: Volume 2*, chapter Data Clustering: Integrating Different Distance Measures with Modified k-Means Algorithm, pages 691–700. Springer India, India.

Georgios Petkos, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2012. Social event detection using multimodal clustering and integrating supervisory signals.

In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 23. ACM.

Xipeng Qiu, Le Tian, and Xuanjing Huang. 2013. Latent semantic tensor indexing for community-based question answering. In *ACL (2)*, pages 434–439.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the past: answering new questions with past answers. In *Proceedings of the 21st international conference on World Wide Web*, pages 759–768. ACM.

Douglas. Steinley. 2006. K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34.

Mu-Chun Su and Chien-Hsing Chou. 2001. A modified version of the k-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):674–680.

N Karthikeyani Visalakshi and J Suguna. 2009. K-means clustering using max-min distance measure. In *Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American*, pages 1–6. IEEE.

Baoxun Wang, Bingquan Liu, Xiaolong Wang, Chengjie Sun, and Deyuan Zhang. 2011. Deep learning approaches to semantic relevance modeling for chinese question-answer pairs. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(4):21.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 475–482. ACM.

Tom Chao Zhou, Michael Rung-Tsong Lyu, Irwin King, and Jie Lou. 2015. Learning to suggest questions in social media. *Knowledge and Information Systems*, 43(2):389–416.

Guangyou Zhou, Yin Zhou, Tingting He, and Wensheng Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems*, 93:75–83.

It Takes Three to Tango: Triangulation Approach to Answer Ranking in Community Question Answering

Preslav Nakov, **Lluís Màrquez** and **Francisco Guzmán**
Arabic Language Technologies Research Group
Qatar Computing Research Institute, HBKU
{pnakov,lmarquez,fguzman}@qf.org.qa

Abstract

We address the problem of answering new questions in community forums, by selecting suitable answers to already asked questions. We approach the task as an answer ranking problem, adopting a pairwise neural network architecture that selects which of two competing answers is better. We focus on the utility of the three types of similarities occurring in the triangle formed by the original question, the related question, and an answer to the related comment, which we call *relevance, relatedness*, and *appropriateness*. Our proposed neural network models the interactions among all input components using syntactic and semantic embeddings, lexical matching, and domain-specific features. It achieves state-of-the-art results, showing that the three similarities are important and need to be modeled together. Our experiments demonstrate that all feature types are relevant, but the most important ones are the lexical similarity features, the domain-specific features, and the syntactic and semantic embeddings.

1 Introduction

In recent years, community Question Answering (cQA) forums, such as StackOverflow, Quora, Qatar Living, etc., have gained a lot of popularity as a source of knowledge and information. These forums typically organize their content in the form of multiple topic-oriented *question–comment threads*, where a question posed by a user is followed by a list of other users' comments, which intend to answer the question.

Many of such on-line forums are not moderated, which often results in (a) *noisy* and (b) *redundant* content, as users tend to deviate from the question and start asking new questions or engage in conversations, fights, etc.

Web forums try to solve problem (a) in various ways, most often by allowing users to up/down-vote answers according to their perceived usefulness, which makes it easier to retrieve useful answers in the future. Unfortunately, this negatively penalizes recent comments, which might be the most relevant and updated ones. This is due to the time it takes for a comment to accumulate votes. Moreover, voting is prone to abuse by forum trolls (Mihaylov et al., 2015; Mihaylov and Nakov, 2016a).

Problem (b) is harder to solve, as it requires that users verify that their question has not been asked before, possibly in a slightly different way. This search can be hard, especially for less experienced users as most sites only offer basic search, e.g., a site search by Google. Yet, solving problem (b) automatically is important both for site owners, as they want to prevent question duplication as much as possible, and for users, as finding an answer to their questions without posting means immediate satisfaction of their information needs.

In this paper, we address the general problem of finding good answers to a given *new question* (referred to as *original question*) in one such community-created forum. More specifically, we use a pairwise deep neural network to rank comments retrieved from different question-comment threads according to their *relevance* as answers to the original question being asked.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1586–1597,
Austin, Texas, November 1-5, 2016.

A key feature of our approach is that we investigate the contribution of the edges in the triangle formed by the pairwise interactions between the *original question*, the *related question*, and the *related comments* to rank comments in a unified fashion. Additionally, we use three different sets of features that capture such similarity: lexical, distributed (semantics/syntax), and domain-specific knowledge.

The experimental results show that addressing the answer ranking task directly, i.e., modelling only the similarity between the original question and the answer-candidate comments, yields very low results. The other two edges of the triangle are needed to obtain good results, i.e., the similarity between the original question and the related question and the similarity between the related question and the related comments. Both aspects add significant and cumulative improvements to the overall performance. Finally, we show that the full network, including the three pairs of similarities, outperforms the state-of-the-art on a benchmark dataset.

The rest of the paper is organized as follows: Section 2 discusses the similarity triangle in answer ranking for cQA, Section 3 presents our pairwise neural network model for answering new questions in community forums, which integrates multiple levels of interaction, Section 4 describes the features we used, Section 5 presents our evaluation setup, the experiments and the results, Section 6 discusses some related work, and Section 7 wraps up the paper with a brief summary of the contributions and some possible directions for future work.

2 The Similarity Triangle in cQA

Figure 1 presents an example illustrating the similarity triangle that we use when solving the answer ranking problem in cQA. In the figure, q stands for the new question, q' is an existing related question, and c is a comment within the thread of question q'.

The edge $\overline{qc}$ relates to the main cQA task addressed in this paper, i.e., deciding whether a comment for a potentially related question is a good answer to the original question. We will say that the relation captures the *relevance* of c for q.

The edge $\overline{qq'}$ represents the similarity between the original and the related questions. We will call this relation *relatedness*.

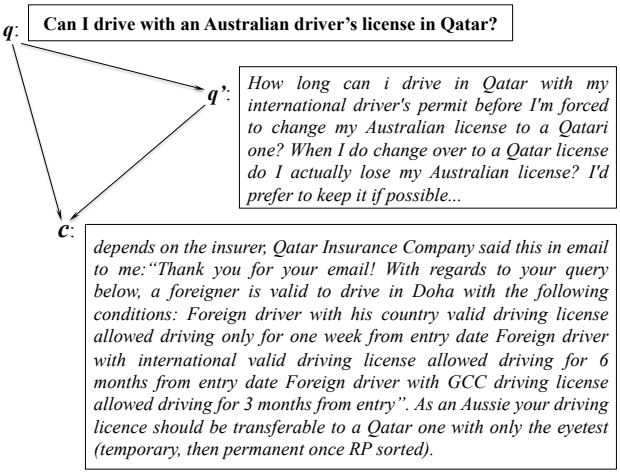


Figure 1: The similarity triangle in cQA.

Finally, the edge $\overline{q'c}$ represents the decision of whether c is a good answer for the question from its thread, q'. We will call this relation *appropriateness*.

In this particular example, q and q' are indeed related, and c is a good answer for both q' and q.[1]

In the past, the approaches to cQA were focused on using information from the new question q, an existing related question q', and a comment c within the thread of q', to solve different cQA sub-tasks. For example, *answer selection*, which selects the most appropriate comment c within the thread q', was addressed in SemEval-2015 Task 3 (Nakov et al., 2015). Similarly, *question–question similarity*, which looks for the most related questions to a given question, was addressed by many authors (Jeon et al., 2005; Duan et al., 2008; Li and Manandhar, 2011; Zhou et al., 2015; dos Santos et al., 2015).

In this paper, we solve the cQA task problem[2] in a novel way by using the three types of similarities jointly. Our main hypothesis is that *relevance*, *appropriateness*, and *relatedness* are essential to finding the best answer in a community Question Answering setting. Below we present experimental results that support this hypothesis.

[1]The essence of this triangle is also described in SemEval 2016 Task 3 to motivate a three-subtask setting for cQA (Nakov et al., 2016). In that evaluation exercise, $\overline{q'c}$ and $\overline{qq'}$ are presented as subtask A and subtask B, respectively. In this paper, we mainly use them as similarity relations to be modeled in the learning architecture to solve the answer ranking task.

[2]We use the task setup and the datasets from SemEval-2016 Task 3, focusing on subtask C (Nakov et al., 2016).

3 Neural Model for Answer Ranking

As explained above, we tackle answer ranking as a three-way similarity problem, exploring similarity features that capture lexical, distributed (semantics and syntax), and domain-specific knowledge. To achieve this, we propose a pairwise neural network (NN) approach for the cQA task, which is inspired by our NN framework for machine translation evaluation (Guzmán et al., 2015).[3] The input of the NN consists of the original question q, two competing comments, c_1 and c_2, and the questions from the threads of the two comments, q'_1 and q'_2. The output of the network is a decision about which of the two comments is a better answer to q.

The main properties of our NN approach can be summarized as follows: (*i*) it works in a pairwise fashion, which is appropriate for the ranking nature of the cQA problem; (*ii*) it allows for an easy incorporation of rich syntactic and semantic embedded representations of the input texts; (*iii*) it models non-linear relationships between all input elements (q, c_1, c_2, q'_1 and q'_2), which allows us to study the interactions and the impact of the three types of similarity (relevance, relatedness and appropriateness) when solving the answer ranking task.

3.1 Architecture

Our full NN model for pairwise answer ranking is depicted in Figure 2. We have a binary classification task with input $x = (q, q'_1, c_1, q'_2, c_2)$, which should output 1 if c_1 is a better answer to the original question q than c_2, and 0 otherwise.[4] In this setting, q'_1 and q'_2 are questions related to q, whose threads contain the comments c_1 and c_2, respectively. They provide useful information to link the two comments to the original question. On the one hand, they allow to predict whether the comments are good answers within their respective threads. On the other hand, they allow to infer whether the questions for which the comments were produced are closely related to the original question. The pair of comments can belong to the same thread (i.e., $q'_1 \equiv q'_2$) or they can come from different threads.

[3] Also, we previously used a similar framework for finding good answers in a question-comment thread (Guzmán et al., 2016a; Guzmán et al., 2016b).

[4] In this work, we do not learn to predict ties, and ties are excluded from our training data.

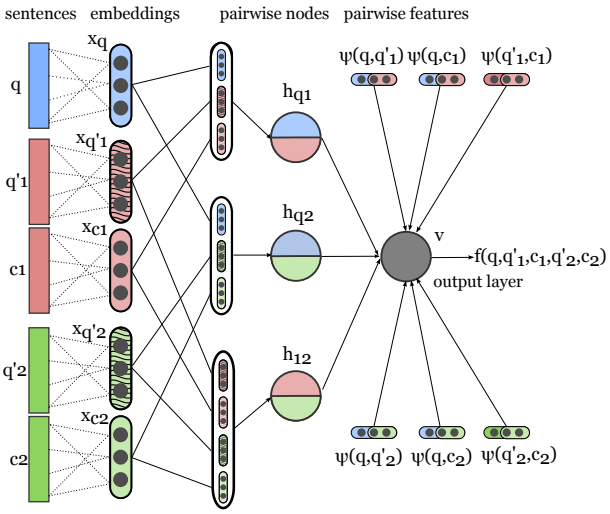


Figure 2: The overall architecture of our neural network model for pairwise answer ranking in community question answering.

The feed-forward neural network computes a sigmoid function $f(q, q'_1, c_1, q'_2, c_2) = \text{sig}(\mathbf{w_v^T}\phi(q, q'_1, c_1, q'_2, c_2) + b_v)$, where $\phi(.)$ transforms the input through the hidden layer, $\mathbf{w_v}$ are the weights from the hidden layer to the output layer, and b_v is a bias term. The function $\phi(.)$ is actually a concatenation of three subfunctions: $\phi(q, q'_1, c_1, q'_2, c_2) = [\phi_1(q, q'_1, c_1), \phi_2(q, q'_2, c_2), \phi_{1,2}(q'_1, c_1, q'_2, c_2)]$.

We first map the question and the comments to a fixed-length vector $[\mathbf{x}_q, \mathbf{x}_{q'_1}, \mathbf{x}_{c_1}, \mathbf{x}_{q'_2}, \mathbf{x}_{c_2}]$ using syntactic and semantic embeddings. Then, we feed this vector as input to the neural network, which models several types of interactions, using different groups of nodes in the hidden layer. Overall, we make use of three different groups of nodes in the hidden layer.

The first two groups include the *relevance* nodes h_{q1} and h_{q2}. These groups of hidden nodes model how relevant comment c_j is to the original question q given that it belongs to the thread of the related question q'_j. In these hidden nodes, we model complex non-linear interactions between the distributed representations of q, q'_j and c_j. Intuitively, these nodes are designed to learn to distinguish a relevant comment by extracting features from the distributed representations of a comment and of the question it is supposed to answer.

The last group of nodes in the hidden layer is the *similarity* node h_{12}. It measures the similarity between c_1 and c_2 and their respective questions q'_1 and q'_2. This node is designed to compute the non-linear interactions between the syntactic and semantic representations of comment-comment, comment-question and question-question pairs. Intuitively, this can help disambiguate when comments are very similar or were generated from the same or from very similar questions.

The model further allows to incorporate external sources of information in the form of *skip arcs* that go directly from the input to the output layer, skipping the hidden layer. These arcs represent pairwise *similarity* feature vectors inspired by the edges of the triangle in Figure 1. In Figure 2, we indicate these pairwise external feature sets as: $\psi(q, q'_1)$, $\psi(q, q'_2)$ for *relatedness*; $\psi(q'_1, c_1)$, $\psi(q'_2, c_2)$ for *appropriateness*; and $\psi(q, c_1)$, $\psi(q, c_2)$ for *relevance*. When including the *skip-arc* features, the activation at the output is $f(q, q'_1, c_1, q'_2, c_2) = \text{sig}(\mathbf{w_v^T}[\phi(q, q'_1, c_1, q'_2, c_2), \psi(q, q'_1), \psi(q, q'_2), \psi(q'_1, c_1), \psi(q'_2, c_2), \psi(q, c_1), \psi(q, c_2)] + b_v)$.

We use these feature vectors to encode machine translation evaluation measures, components thereof, cQA task-specific features, etc. The next section gives more detail about these features.

4 Features

We experiment with three kinds of features: (*i*) lexical features that measure similarity at a word, word n-gram, and paraphrase level, (*ii*) distributed representations that measure similarity at a syntactic and semantic level, (*iii*) domain-specific knowledge features, which capture similarity using thread-level information and other features that have proven valuable to solve similar tasks (Nicosia et al., 2015).

4.1 Lexical similarity features

These types of features measure similarity at a surface level between the following pairs: (q,q'_1), (q,q'_2), (q'_1, c_1), (q'_2, c_2), (q_1, c_1), and (q_2, c_2). They are inspired by our previous work on Machine Translation Evaluation (MTE) (Guzmán et al., 2015), and we previously found them useful for finding good answers in a question-comment thread (Guzmán et al., 2016a; Guzmán et al., 2016b).

MTFEATS We use (as pairwise features) the following six machine translation evaluation features: (*i*) BLEU: This is the most commonly used measure for machine translation evaluation, which is based on n-gram overlap and length ratios (Papineni et al., 2002). (*ii*) NIST: This measure is similar to BLEU, and is used at evaluation campaigns run by NIST (Doddington, 2002). (*iii*) TER: Translation error rate; it is based on the edit distance between a translation hypothesis and the reference (Snover et al., 2006). (*iv*) METEOR: A complex measure, which matches the hypothesis and the reference using synonyms and paraphrases (Lavie and Denkowski, 2009). (*v*) Unigram PRECISION and RECALL.

BLEUCOMP Following (Guzmán et al., 2015), we further use as features various components that are involved in the computation of BLEU: n-gram precisions, n-gram matches, total number of n-grams (n=1,2,3,4), lengths of the hypotheses and of the reference, length ratio between them, and BLEU's brevity penalty. Again, these are computed over the same six pairs of vectors as before.

4.2 Distributed representations

We use the following vector-based embeddings of all input components: q, c_1, c_2, q'_1, and q'_2.

GOOGLE_VEC We use the pre-trained, 300-dimensional embedding vectors from WORD2VEC (Mikolov et al., 2013). We compute a vector representation of the text by simply averaging over the embeddings of all words in the text.

QL_VEC We train in-domain word embeddings using WORD2VEC on all available QatarLiving data. Again, we use these embeddings to compute 100-dimensional vector representations for all input components by averaging over all words in the texts.

SYNTAX_VEC We parse the entire question/comment using the Stanford neural parser (Socher et al., 2013), and we use the final 25-dimensional vector that is produced internally as a by-product of parsing.

Moreover, we use the above vectors to calculate pairwise similarity features, i.e., the cosine between the following six vector pairs: (q, c_1), (q, c_2), (q'_1, c_1), (q'_2, c_2), (q, q'_1) and (q, q'_2).

4.3 Domain-specific features

We extract various domain-specific features that use thread-level and other useful information known to capture *relatedness* and *appropriateness*.

SAME_AUTHOR We have a thread-level meta-feature, which we apply to the pairs (q'_1, c_1), (q'_2, c_2). It checks whether the person answering the question is also the one who asked it, i.e., do the related question and the comment have the same author. The idea is that the person asking a question is unlikely to answer his/her own question, but s/he could ask a clarification question or thank another person who has provided a useful answer earlier in the thread.

CQ′RANK_FEAT We further have two thread-level meta-features related to the rank of the comment in the thread, which we apply to the pairs (q'_1, c_1) and (q'_2, c_2): (*i*) reciprocal rank of the comment in the thread, i.e., $1/\rho$, where ρ is the rank of the comment; (*ii*) percentile of the number of comments in the thread, calculated as follows: the first comment gets the score of 1.0, the second one gets 0.9, and so on. Note that in our dataset, there are exactly ten comments per thread.

QQ′RANK_FEAT We also have three features modeling the rank of the related question in the list of related questions for the original question, which we apply to the pairs (q, q'_1) and (q, q'_2).

In total, use the following six features: (*i*) the reciprocal rank of q'_1 or q'_2 in the list of related questions for q; (*ii*) the reciprocal ordinal rank[5] of q'_1 or q'_2 in the list of related questions for q; (*iii*) the percentile of the q'_1 or q'_2 in the list of related questions for q, calculated as for the comments.

CQRANK_FEAT. Finally, we have features for the rank of the comment in the list of 100 comments for the original question, which we apply to the pairs (q, c_1) and (q, c_2): (*i*) reciprocal rank of the comment in the list; (*ii*) percentile of the comment in the list.

TASK_FEAT. We further have features that have been proven useful in the answer selection task from SemEval 2015 Task 3 (Nakov et al., 2015). This includes some comment-specific features, which refer to c_1 and c_2 only, but which we apply twice, to generate features for the pairs (q'_1, c_1), (q'_2, c_2), (q_1, c_1), and (q_2, c_2): number of URLs/images/emails/phone numbers; number of occurrences of the string *thank*;[6] number of tokens/sentences; average number of tokens; number of nouns/verbs/adjectives/adverbs/pronouns; number of positive/negative smileys; number of single/double/triple exclamation/ interrogation symbols; number of interrogative sentences (based on parsing); number of words that are not in word2vec's Google News vocabulary.[7]

And also some question-comment pair features, which we apply to the pairs (q'_1, c_1), (q'_2, c_2), (q_1, c_1), and (q_2, c_2): (*i*) question to comment count ratio in terms of sentences/tokens/nouns/verbs/adjectives/adverbs/pronouns; (*ii*) question to comment count ratio of words that are not in word2vec's Google News vocabulary.

5 Experiments and Results

We experimented with the data from SemEval-2016 Task 3 on "Community Question Answering". More precisely, the problem addressed is subtask C (*Question–External Comment Similarity*), which is the primary cQA task. For a given new question (referred to as the *original question*), the task provides the set of the first ten related questions (retrieved by a search engine), each associated with the first ten comments appearing in the question-comment thread. The goal then is to rank the total of 100 comments according to their appropriateness with respect to the original question.

In this framework, the retrieval part of the task is done as a pre-processing step, and the challenge is to learn to rank all *good* comments above all *bad* ones. All the data comes from the QatarLiving forum, and the related questions are obtained using Google search with the original question's text limited to the `www.qatarliving.com` domain.

[5]The related questions are obtained using a query to a search engine (using words from the original question), with results limited to QatarLiving. However, some of the returned results pointed to the wrong (non-forum) sections of the website or to questions with less than ten comments, and these were skipped. Suppose that the surviving top ten related questions were at ranks 3, 7, 18, ... in the original list. Now, we can use these ranks ρ, or we can use instead the ordinal ranks r: 1, 2, 3, ...

[6]When an author thanks somebody, this post is typically a bad answer to the original question.

[7]Can detect slang, foreign language, etc., which would indicate a bad answer.

The task offers a higher quality training dataset TRAIN-PART1, which includes 200 original questions, 1,999 related questions and 19,990 comments, and a lower-quality TRAIN-PART2, which we did not use. Additionally, it provides a development set (DEV, with 50 original questions, 500 related questions and 5,000 related comments) and a TEST set (70 original questions, 700 related questions and 7,000 related comments). Apart from the class labels for subtask C, the datasets also offer class labels for subtask A (i.e., whether a comment is a good answer to the question in the thread) and subtask B (i.e., whether the related questions is relevant for the original question).

5.1 Setting

we use Theano (Bergstra et al., 2010) to train our model on TRAIN-PART1 with hidden layers of size 3 for 100 epochs with minibatches of size 30, regularization of 0.05, and a learning rate of 0.01, using stochastic gradient descent with adagrad (Duchi et al., 2011). We normalize the input feature values to the $[-1; 1]$ interval using minmax, and we initialize the NN weights by sampling from a uniform distribution as in (Bengio and Glorot, 2010).

We evaluate the model on DEV after each epoch, and ultimately we keep the model that achieves the highest accuracy;[8] in case of a tie, we prefer the parameters from a later epoch. We selected the above parameter values on the DEV dataset using the full model, and we use them for all experiments in Section 5.3, where we evaluate on the TEST dataset.

Note that, we train the NN using all pairs of (Good, Bad) comments, in both orders, ignoring ties. At test time, we compute the full ranking of comments by scoring all possible pairs, and by then accumulating the scores at the comment level.

5.2 Evaluation and baselines

The results are calculated with the official scorer from the SemEval-2016 Task 3. We report three ranking-based measures that are commonly accepted in the IR community: Mean average precision (MAP), which is the official evaluation measure of the task, average recall (AvgRec), and mean reciprocal rank (MRR).

[8] We tried Kendall's Tau (τ), but it performed slightly worse.

For comparison purposes, we report the results for two baselines. One corresponds to a random ordering of the comments, assuming zero knowledge of the task. The second one is a more realistic baseline, which keeps the question ranking from the search engine (Google search) and the chronological order of the comments within the thread of teh related question. Although this may be considered a very naïve baseline, it is actually notably informed. The question ranking from Google search takes into account the relevance of the entire thread (question and comments) to the original question. Moreover, there is a natural concentration of the best answers in the first comments of the threads.

5.3 Main results

Table 1 shows the evaluation results on the TEST dataset for several variants of our pairwise neural network architecture. Regarding our network configurations, we present the results from simpler to more complex.

Relevance The "Relevance only" network contains only the *relevance* relations and features corresponding to q, c_1 and c_2. The rest of the components are deactivated in the network. This corresponds to solving the task without any information about the related questions and the appropriateness of the comments in their threads, i.e., just by comparing the texts of the comments and of the original question. In some sense, this setup is largely less informed than the IR baseline. The results are very low, being only $\sim$7 MAP points higher than the random baseline.

Relevance + appropriateness Adding the *appropriateness* interactions between c_1 and q'_1, and between c_2 and q'_2 improves MAP by $\sim$9 points. Although more informed, as some information from the related questions is taken indirectly, the results of this system are still below the IR baseline.

Relevance + relatedness Adding the *relatedness* interactions and features between q and q'_1, and q and q'_2, turns out to be crucial. When added to the "Relevance only" basic system, the MAP score jumps to 52.43, significantly above the IR baseline. This shows that *question–question* similarity plays an important role in solving the cQA task.

System	MAP	AvgRec	MRR
Relevance relations only	21.78	20.66	22.59
+ Appropriateness	30.94	29.86	35.02
+ Relatedness	52.43	57.05	60.14
Full Network	**54.51**	**60.93**	**62.94**
Baseline 1 (random)	15.01	11.44	15.19
Baseline 2 (IR+chron.)	40.36	45.97	45.83

Table 1: Results on the answer ranking task of our full NN vs. variants using partial information.

Full Network Adding both *appropriateness* and *relatedness* interactions yields an improvement of another two MAP points absolute (to 54.51), which shows that *appropriateness* features encode information that is complementary to the information modeled by *relevance* and *relatedness*. Note that the results with the other evaluation metrics (AvgRec and MRR) follow exactly the same pattern. In summary, we can conclude that in order to solve the community question answering problem, we need to (*i*) find the best related questions, and (*ii*) judge the relevance of individual comments with respect to the new question.

5.4 Features in perspective

Table 2 shows the results of an ablation study when removing some groups of features.[9] More specifically, we drop lexical similarities, domain-specific features, and the complex semantic-syntactic interactions modeled in the hidden layer between the embeddings and the domain-specific features.

We can see that the lexical similarity features (which we modeled by MT evaluation metrics), have a large impact: excluding them from the network yields a decrease of over eight MAP points. This can be explained as the strong dependence that *relatedness* has over strict word matching. Since questions are relatively short, a better related question will be one that matches better the original question.

[9]Note that here we only show the impact of *groups* of features, e.g., we do not consider experiments with different embeddings such as GOOGLE_VEC, QL_VEC, and SYNTAX_VEC, which all belong to the lexical similarity group of features. This is because in previous work (which was limited to subtask A), our ablation study has shown that all features in a group clearly contribute to the overall performance (Guzmán et al., 2016a; Guzmán et al., 2016b).

System	MAP	AvgRec	MRR	Δ_{MAP}
Full Network	**54.51**	**60.93**	**62.94**	
– Lexical similarity	45.89	51.54	53.29	-8.62
– Domain-specific	48.48	50.46	53.78	-6.03
– Distributed rep.	51.17	56.63	56.91	-3.34
No hidden layer	52.19	58.23	59.95	-2.32

Table 2: Results of the ablation study.

As expected, eliminating the domain-specific features also hurts the performance greatly: by six MAP points absolute. Eliminating the use of distributed representation has a lesser impact: 3.3 MAP points absolute. This is in line with our previous findings (Guzmán et al., 2015; Guzmán et al., 2016a; Guzmán et al., 2016b) that semantic and syntactic embeddings are useful to make a fine-grained distinction between comments (*relevance*, *appropriateness*), which are usually longer.

We have also found that there is an interaction between features and similarity relations. For example, for *relatedness*, lexical similarity is 2.6 MAP points more informative[10] than distributed representations. In contrast, for *relevance*, distributed representations are 0.7 MAP points more informative than lexical similarities.

5.5 Impact of the hidden layer

Table 2 also presents the results of a system that has the full set of features, but eliminates the hidden layer from the neural network. This is equivalent to training a Maximum Entropy classifier with the complete set of features. This simplified system performs consistently worse than the full NN model (-2.32 MAP, -2.7 AvgRec, and -2.99 MRR points), which shows that using the hidden layer to model the non-linear interactions between information sources has a decent overall contribution.

5.6 Making appropriateness more useful

Since the SemEval-2016 Task 3 datasets also provide labeled examples for the so called "subtask A" ($\overline{q'c}$; appropriateness) and "subtask B" ($\overline{qq'}$; relatedness), one could use this supervision to help train the neural network for the primary cQA task. We observed that *relatedness* has proven quite informative. However, the improvements observed from using *appropriateness* were more modest.

[10]As measured by the relative drop in MAP performance.

System	MAP	AvgRec	MRR
Full Network	54.51	60.93	62.94
Full + *appr.* preds.	**55.82**	**61.63**	**62.39**

Table 3: Using *appropriateness* predictions.

We present here a stacked experiment in which an additional neural network trained to predict *appropriateness* is used to inform the full network model. More concretely, we train a feed-forward pairwise neural network for subtask A, which is a simplification of the architecture from Figure 2. The input is reduced to three elements (q', c_1, c_2), where q' is the thread question and c_1 and c_2 are a pair of comments in the thread. The output consists of deciding whether c_1 is a better answer to q' than c_2. All the pairwise interactions between input components are included in the hidden layer, and we use the same features to train the network as the ones described in Section 4 (obviously, this time the input and the features are reduced to those involving q', c_1 and c_2). We used this exact setting in previous work for solving subtask A (Guzmán et al., 2016a; Guzmán et al., 2016b).

We used the network to classify all subtask A examples in TRAIN-PART1, DEV and TEST, and we used the resulting scores at the comment level as skip-arc features for the full NN model: (a) alone, included in $\psi(q'_1, c_1)$ and $\psi(q'_2, c_2)$, and (b) multiplied by each of the QQ′Rank_feat features, included in $\psi(q, c_1)$ and $\psi(q, c_2)$.

In Table 3, we observe that using the pre-trained network to incorporate subtask A predictions as features yields another sizable improvement to a final MAP of 55.82 (the increase is smaller for AvgRec, and MRR is slightly hurt), which suggests that pre-training parts of the NN with labeled examples to perform a dedicated task, is a promising direction for future work.

5.7 Results in perspective

Next, in order to put our results in perspective, we compare them to the state of the art for this problem, represented by the systems that participated in SemEval-2016 Task 3, subtask C. The comparison is shown in Table 4, where we list the top-3 systems, as well as the average and the worst scores for the official runs of all participating teams.

System	MAP	AvgRec	MRR
Full Network + subtask A preds.	**55.82**	**61.63**	62.39
* 1st (Mihaylova et al., 2016)	55.41	60.66	61.48
Full Network	54.51	60.93	**62.94**
* 2nd (Filice et al., 2016)	52.95	59.27	59.23
* 3rd (Mihaylov and Nakov, 2016b)	51.68	53.43	55.96
...	...	...	...
SemEval Average	49.30	53.74	54.39
...	...	...	...
SemEval Worst	43.20	47.96	47.79
Baseline 2 (IR+chron.)	40.36	45.97	45.83

Table 4: Comparative results with the state of the art, i.e., the top-3 systems that participated in SemEval-2016 Task 3, subtask C.

We can see that all systems in the competition performed over the IR baseline with MAP scores ranging from 43.20 to 55.41. We can further see that our full network with subtask A predictions achieves the best results with 55.82 MAP. The margin over the best SemEval system is small in terms of MAP but more noticeable in terms of AvgRec and MRR. Note that, even without the Subtask A predictions, our pairwise neural network still produces results that are on par with the state of the art (with improvements slightly over one point in both cases).

6 Related Work

Recently, a variety of neural network models have been applied to community question answering tasks such as *question-question similarity* (Zhou et al., 2015; dos Santos et al., 2015; Lei et al., 2015) and *answer selection* (Severyn and Moschitti, 2015; Wang and Nyberg, 2015; Feng et al., 2015; Tan et al., 2015; Filice et al., 2016; Barrón-Cedeño et al., 2016; Mohtarami et al., 2016). Most of these papers concentrate on constructing advanced neural network architectures in order to model the problem at hand better.

For instance, dos Santos et al. (2015) propose a neural network approach combining a convolutional neural network and a bag-of-words representation for modeling question-question similarity. Similarly, Tan et al. (2015) adopt a neural attention mechanism over bidirectional long short-term memory (LSTM) neural network to generate better answer representations given the questions.

Similarly, Lei et al. (2015) use a combination of recurrent and convolutional neural models to map questions to semantic representations. The models are pre-trained within an encoder-decoder framework (from body to title) in order to de-noise the long question body from irrelevant text.

The main objective of our work here is different: we focus on studying the impact of the different input components in a novel cQA setting of ranking answers for new questions, and we use a more standard neural network.

The setting of cQA as a triangle of three interrelated subtasks, which we use here, has been recently proposed in SemEval-2016 Task 3 on *Community Question Answering* (Nakov et al., 2016). Above, we empirically compared our results to those of the best participating systems. Unfortunately, most of the systems that took part in the competition, including the winning system of the SUper team (Mihaylova et al., 2016), approached the task indirectly by solving subtask A at the thread level and then using these predictions together with the reciprocal rank of the related questions to produce a final ranking for subtask C.

One exception is the *Kelp* system (Filice et al., 2016), which was ranked second in the competition. Their approach is most similar to ours, as it also tries to combine information from different subtasks and from all input components. It does so in a modular kernel function, including stacking from independent subtask A and B classifiers, and it applies SVMs to train a Good vs. Bad classifier (Filice et al., 2016). In contrast, our approach here proceeds in a pairwise setting, it is lighter in terms of features engineering, and presents a direct way to combine the relations between the different subtasks in an integrated neural network model.

Finally, our model uses lexical features derived from machine translation evaluation. Some previous work also used MT model(s) as a feature(s) (Berger et al., 2000; Echihabi and Marcu, 2003; Jeon et al., 2005; Soricut and Brill, 2006; Riezler et al., 2007; Li and Manandhar, 2011; Surdeanu et al., 2011; Tran et al., 2015; Hoogeveen et al., 2016; Wu and Zhang, 2016), e.g., a variation of IBM model 1 (Brown et al., 1993), to compute the probability that the question is a "translation" of the candidate answer.

7 Conclusion

We presented a neural-based approach to a novel problem in cQA, where given a new question, the task is to rank comments from related question-threads according to their relevance as answers to the original question. We explored the utility of three types of similarities between the original question, the related question, and the related comment.

We adopted a pairwise feed-forward neural network architecture, which takes as input the original question and two comments together with their corresponding related questions. This allowed us to study the impact and the interaction effects of the question-question *relatedness* and comment-to-related question *appropriateness* relations when solving the primary cQA *relevance* task. The large performance gains obtained from using *relatedness* features show that question-question similarity plays a crucial role in finding relevant comments (+30 MAP points). Yet, including *appropriateness* relations is needed to achieve state-of-the-art results (+3.3 MAP) on benchmark datasets.

We also studied the impact of several types of features, especially domain-specific features, but also lexical features and syntactic embeddings. We observed that lexical similarity MTE features prove the most important, followed by domain-specific features, and syntactic and semantic embeddings. Overall, they all showed to be necessary to achieve state-of-the-art results.

In future work, we plan to use the labels for subtasks A and B, which are provided in the datasets in order to pre-train the corresponding components of the full network for answer ranking. We further want to apply a similar network to other semantic similarity problems, such as textual entailment.

Acknowledgments

This research was performed by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI), HBKU, part of Qatar Foundation. It is part of the Interactive sYstems for Answer Search (Iyas) project, which is developed in collaboration with MIT-CSAIL.

Last but not least, we would also like to thank the anonymous reviewers for their constructive comments, which have helped us improve the paper.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A. Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 896–903, San Diego, CA.

Yoshua Bengio and Xavier Glorot. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of Artificial Intelligence and Statistics*, AISTATS '10, pages 249–256, Chia Laguna Resort, Sardinia, Italy.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 192–199, Athens, Greece.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*, SciPy '10, Austin, TX.

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

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Diego, CA.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 694–699, Beijing, China.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '08, pages 156–164, Columbus, OH.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, ACL '03, pages 16–23, Sapporo, Japan.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, ASRU '15, pages 813–820, Scottsdale, AZ.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1116–1123, San Diego, CA.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 805–814, Beijing, China.

Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2016a. Machine translation evaluation meets community question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 460–466, Berlin, Germany.

Francisco Guzmán, Preslav Nakov, and Lluís Màrquez. 2016b. MTE-NN at SemEval-2016 Task 3: Can machine translation evaluation help community question answering? In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 887–895, San Diego, CA.

Doris Hoogeveen, Yitong Li, Huizhi Liang, Bahar Salehi, Timothy Baldwin, and Long Duong. 2016. UniMelb at SemEval-2016 Task 3: Identifying similar questions by combining a CNN with string similarity measures. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 851–856, San Diego, CA.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, Bremen, Germany.

Alon Lavie and Michael Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2–3):105–115.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi S. Jaakkola, Kateryna Tymoshenko, Alessandro Mos-

chitti, and Lluís Màrquez. 2015. Denoising bodies to titles: Retrieving similar questions with recurrent convolutional models. *arXiv preprint arXiv:1512.05726*.

Shuguang Li and Suresh Manandhar. 2011. Improving question recommendation by exploiting information need. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL '11, pages 1425–1434, Portland, OR.

Todor Mihaylov and Preslav Nakov. 2016a. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 399–405, Berlin, Germany.

Todor Mihaylov and Preslav Nakov. 2016b. SemanticZ at SemEval-2016 Task 3: Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 879–886, San Diego, CA.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, CoNLL '15, pages 310–314, Beijing, China.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. SUper Team at SemEval-2016 Task 3: Building a feature-rich system for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 836–843, San Diego, CA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 746–751, Atlanta, GA.

Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, Scott Cyphers, and Jim Glass. 2016. SLS at SemEval-2016 Task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 828–835, San Diego, CA.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 Task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 269–281, Denver, CO.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 525–545, San Diego, CA.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeño, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, Lluís Màrquez, Shafiq Joty, and Walid Magdy. 2015. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '2015, pages 203–209, Denver, CO.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, PA.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 464–471, Prague, Czech Republic.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, Santiago, Chile.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, pages 223–231, Cambridge, MA.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 455–465, Sofia, Bulgaria.

Radu Soricut and Eric Brill. 2006. Automatic question answering using the web: Beyond the factoid. *Inf. Retr.*, 9(2):191–206.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Comput. Linguist.*, 37(2):351–383.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. JAIST: Combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 215–219, Denver, CO.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 707–712, Beijing, China.

Yunfang Wu and Minghua Zhang. 2016. ICL00 at SemEval-2016 Task 3: Translation-based method for CQA system. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 857–860, San Diego, CA.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 250–259, Beijing, China.

Character-Level Question Answering with Attention

David Golub
University of Washington
golubd@cs.washington.edu

Xiaodong He
Microsoft Research
xiaohe@microsoft.com

Abstract

We show that a character-level encoder-decoder framework can be successfully applied to question answering with a structured knowledge base. We use our model for single-relation question answering and demonstrate the effectiveness of our approach on the SimpleQuestions dataset (Bordes et al., 2015), where we improve state-of-the-art accuracy from 63.9% to 70.9%, without use of ensembles. Importantly, our character-level model has 16x fewer parameters than an equivalent word-level model, can be learned with significantly less data compared to previous work, which relies on data augmentation, and is robust to new entities in testing. [1]

1 Introduction

Single-relation factoid questions are the most common form of questions found in search query logs and community question answering websites (Yih et al., 2014; Fader et al., 2013). A knowledge-base (KB) such as Freebase, DBpedia, or Wikidata can help answer such questions after users reformulate them as queries. For instance, the question "Where was Barack Obama born?" can be answered by issuing the following KB query:

$$\lambda(x).place\_of\_birth(Barack\_Obama, x)$$

However, automatically mapping a natural language question such as "Where was Barack Obama born?" to its corresponding KB query remains a challenging task.

There are three key issues that make learning this mapping non-trivial. First, there are many paraphrases of the same question. Second, many of the KB entries are unseen during training time; however, we still need to correctly predict them at test time. Third, a KB such as Freebase typically contains millions of entities and thousands of predicates, making it difficult for a system to predict these entities at scale (Yih et al., 2014; Fader et al., 2014; Bordes et al., 2015). In this paper, we address all three of these issues with a character-level encoder-decoder framework that significantly improves performance over state-of-the-art word-level neural models, while also providing a much more compact model that can be learned from less data.

First, we use a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder to embed the question. Second, to make our model robust to unseen KB entries, we extract embeddings for questions, predicates and entities purely from their character-level representations. Character-level modeling has been previously shown to generalize well to new words not seen during training (Ljubešić et al., 2014; Chung et al., 2016), which makes it ideal for this task. Third, to scale our model to handle the millions of entities and thousands of predicates in the KB, instead of using a large output layer in the decoder to directly predict the entity and predicate, we use a general interaction function between the question embeddings and KB embeddings that measures their semantic relevance to determine the output. The combined use of character-

[1] Our code is publicly available at https://github.com/davidgolub/simpleqa

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1598–1607,
Austin, Texas, November 1-5, 2016.

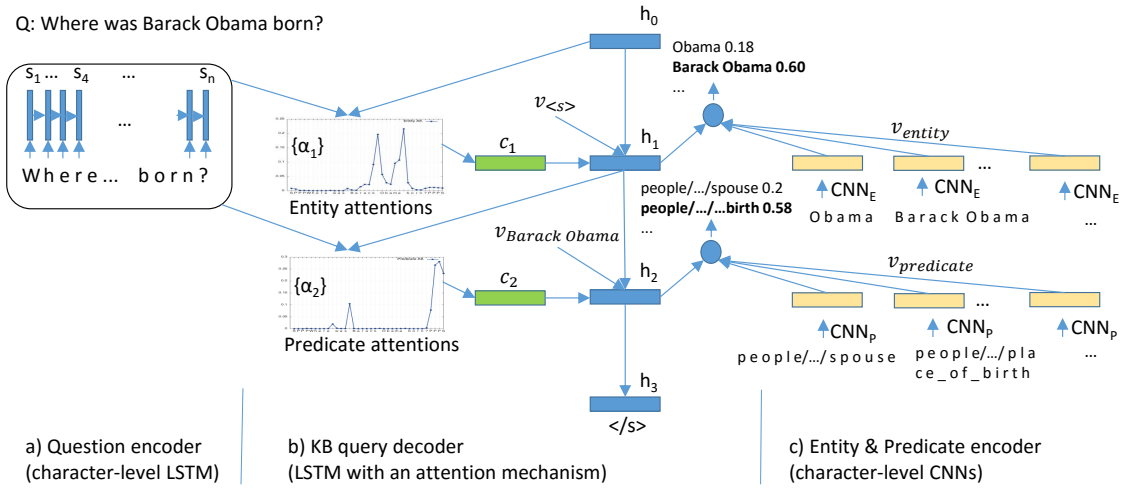


Figure 1: Our encoder-decoder architecture that generates a query against a structured knowledge base. We encode our question via a long short-term memory (LSTM) network and an attention mechanism to produce our context vector. During decoding, at each time step, we feed the current context vector and an embedding of the English alias of the previously generated knowledge base entry into an attention-based decoding LSTM to generate the new candidate entity or predicate.

level modeling and a semantic relevance function allows us to successfully produce likelihood scores for the KB entries that are not present in our vocabulary, a challenging task for standard encoder-decoder frameworks.

Our novel, character-level encoder-decoder model is compact, requires significantly less data to train than previous work, and is able to generalize well to unseen entities in test time. In particular, without use of ensembles, we achieve 70.9% accuracy in the Freebase2M setting and 70.3% accuracy in the Freebase5M setting on the SimpleQuestions dataset, outperforming the previous state-of-arts of 62.7% and 63.9% (Bordes et al., 2015) by 8.2% and 6.4% respectively. Moreover, we only use the training questions provided in SimpleQuestions to train our model, which cover about 24% of words in entity aliases on the test set. This demonstrates the robustness of the character-level model to unseen entities. In contrast, data augmentation is usually necessary to provide more coverage for unseen entities and predicates, as done in previous work (Bordes et al., 2015; Yih et al., 2014).

2 Related Work

Our work is motivated by three major threads of research in machine learning and natural language processing: semantic-parsing for open-domain question answering, character-level language modeling, and encoder-decoder methods.

Semantic parsing for open-domain question answering, which translates a question into a structured KB query, is a key component in question answering with a KB. While early approaches relied on building high-quality lexicons for domain-specific databases such as GeoQuery (Tang and Mooney, 2001), recent work has focused on building semantic parsing frameworks for general knowledge bases such as Freebase (Yih et al., 2014; Bordes et al., 2014a; Bordes et al., 2015; Berant and Liang, 2014; Fader et al., 2013; Dai et al., 2016).

Semantic parsing frameworks for large-scale knowledge bases have to be able to successfully generate queries for the millions of entities and thousands of predicates in the KB, many of which are unseen during training. To address this issue, recent work relies on producing embeddings for predicates and entities in a KB based on their textual descriptions (Bordes et al., 2014a; Bordes et al., 2015; Yih et al., 2014; Yih et al., 2015; Bishan Yang, 2015). A general interaction function can then be used to measure the semantic relevance of these embedded KB entries to the question and determine the most likely KB query.

Most of these approaches use word-level embeddings to encode entities and predicates, and therefore might suffer from the out-of-vocabulary (OOV) problem when they encounter unseen words during test time. Consequently, they often rely on significant data augmentation from sources such as Paralex (Fader et al., 2013), which contains 18 million question-paraphrase pairs scraped from WikiAnswers, to have sufficient examples for each word they encounter (Bordes et al., 2014b; Yih et al., 2014; Bordes et al., 2015).

As opposed to word-level modeling, character-level modeling can be used to handle the OOV issue. While character-level modeling has not been applied to factoid question answering before, it has been successfully applied to information retrieval, machine translation, sentiment analysis, classification, and named entity recognition (Huang et al., 2013; Shen et al., 2014; Chung et al., 2016; Zhang et al., 2015; Santos and Zadrozny, 2014; dos Santos and Gatti, 2014; Klein et al., 2003; dos Santos, 2014; dos Santos et al., 2015). Moreover, Chung et al. (2015) demonstrate that gated-feedback LSTMs on top of character-level embeddings can capture long-term dependencies in language modeling.

Lastly, encoder-decoder networks have been applied to many structured machine learning tasks. First introduced in Sutskever et al. (2014), in an encoder-decoder network, a source sequence is first encoded with a recurrent neural network (RNN) into a fixed-length vector which intuitively captures its "meaning", and then decoded into a desired target sequence. This approach and related memory-based or attention-based approaches have been successfully applied in diverse domains such as speech recognition, machine translation, image captioning, parsing, executing programs, and conversational dialogues (Amodei et al., 2015; Venugopalan et al., 2015; Bahdanau et al., 2015; Vinyals et al., 2015; Zaremba and Sutskever, 2014; Xu et al., 2015; Sukhbaatar et al., 2015).

Unlike previous work, we formulate question answering as a problem of decoding the KB query given the question and KB entries which are encoded in embedding spaces. We therefore integrate the learning of question and KB embeddings in a unified encoder-decoder framework, where the whole system is optimized end-to-end.

3 Model

Since we focus on single-relation question answering in this work, our model decodes every question into a KB query that consists of exactly two elements–the topic entity, and the predicate. More formally, our model is a function $f(q, \{e\}, \{p\})$ that takes as input a question q, a set of candidate entities $\{e\} = e_1, ..., e_n$, a set of candidate predicates $\{p\} = p_1, ..., p_m$, and produces a likelihood score $p(e_i, p_j|q)$ of generating entity e_i and predicate p_j given question q for all $i \in 1...n, j \in 1...m$.

As illustrated in Figure 1, our model consists of three components:

1. A character-level LSTM-based encoder for the question which produces a sequence of embedding vectors, one for each character (Figure 1a).

2. A character-level convolutional neural network (CNN)-based encoder for the predicates/entities in a knowledge base which produces a single embedding vector for each predicate or entity (Figure 1c).

3. An LSTM-based decoder with an attention mechanism and a relevance function for generating the topic entity and predicate to form the KB query (Figure 1b).

The details of each component are described in the following sections.

3.1 Encoding the Question

To encode the question, we take two steps:

1. We first extract one-hot encoding vectors for characters in the question, $x_1, ..., x_n$, where x_i represents the one-hot encoding vector for the i^{th} character in the question. We keep the space, punctuation and original cases without tokenization.

2. We feed $x_1, ..., x_n$ from left to right into a two-layer gated-feedback LSTM, and keep the outputs at all time steps as the embeddings for the question, i.e., these are the vectors $s_1, ..., s_n$.

3.2 Encoding Entities and Predicates in the KB

To encode an entity or predicate in the KB, we take two steps:

1. We first extract one-hot encoding vectors for characters in its English alias, $x_1, ..., x_n$, where x_i represents the one-hot encoding vector for the i^{th} character in the alias.

2. We then feed $x_1, ..., x_n$ into a temporal CNN with two alternating convolutional and fully-connected layers, followed by one fully-connected layer:

$$f(x_1, ..., x_n) = tanh(W_3 \times max(tanh(W_2 \times conv(tanh(W_1 \times conv(x_1, ..., x_n))))))$$

where $f(x_{1...n})$ is an embedding vector of size N, W_3 has size $R^{N \times h}$, $conv$ represents a temporal convolutional neural network, and max represents a max pooling layer in the temporal direction.

We use a CNN as opposed to an LSTM to embed KB entries primarily for computational efficiency. Also, we use two different CNNs to encode entities and predicates because they typically have significantly different styles (e.g., "Barack Obama" vs. "/people/person/place_of_birth").

3.3 Decoding the KB Query

To generate the single topic entity and predicate to form the KB query, we use a decoder with two key components:

1. An LSTM-based decoder with attention. Its hidden states at each time step i, h_i, have the same dimensionality N as the embeddings of entities/predicates. The initial hidden state h_0 is set to the zero vector: $\vec{0}$.

2. A pairwise semantic relevance function that measures the similarity between the hidden units of the LSTM and the embedding of an entity or predicate candidate. It then returns the mostly likely entity or predicate based on the similarity score.

In the following two sections, we will first describe the LSTM decoder with attention, followed by the semantic relevance function.

3.3.1 LSTM-based Decoder with Attention

The attention-based LSTM decoder uses a similar architecture as the one described in Bahdanau et al. (2015). At each time step i, we feed in a context vector c_i and an input vector v_i into the LSTM. At time $i = 1$ we feed a special input vector $v_{<S>} = \vec{0}$ into the LSTM. At time $i = 2$, during training, the input vector is the embedding of the true entity, while during testing, it is the embedding of the most likely entity as determined at the previous time step.

We now describe how we produce the context vector c_i. Let h_{i-1} be the hidden state of the LSTM at time $i-1$, s_j be the j^{th} question character embedding, n be the number of characters in the question, r be the size of s_j, and m be a hyperparameter. Then the context vector c_i, which represents the attention-weighted content of the question, is recomputed at each time step i as follows:

$$c_i = \sum_{j=1}^{n} \alpha_{ij} s_j,$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = v_a^\top \tanh\left(W_a h_{i-1} + U_a s_j\right),$$

where $\{\alpha\}$ is the attention distribution that is applied over each hidden unit s_j, $W_a \in R^{m \times N}, U_a \in R^{m \times r}$, and $v_a \in R^{1 \times m}$.

3.3.2 Semantic Relevance Function

Unlike machine translation and language modeling where the vocabulary is relatively small, there are millions of entries in the KB. If we try to directly predict the KB entries, the decoder will need an output layer with millions of nodes, which is computationally prohibitive. Therefore, we resort to a relevance function that measures the semantic similarity between the decoder's hidden state and the embeddings of KB entries. Our semantic relevance function takes two vectors x_1, x_2 and returns a distance measure of how similar they are to each other. In current experiments we use a simple cosine-similarity metric: $cos(x_1, x_2)$.

Using this similarity metric, the likelihoods of generating entity e_j and predicate p_k are:

RESULTS ON SIMPLEQUESTIONS DATASET									
KB	TRAIN SOURCES			AUTOGEN. QUESTIONS	EMBED TYPE	MODEL	ENSEMBLE	SQ Accuracy	# TRAIN EXAMPLES
	WQ	SIQ	PRP						
FB2M	no	yes	no	no	Char	Ours	1 model	**70.9**	76K
FB2M	no	yes	no	no	Word	Ours	1 model	53.9	76K
FB2M	yes	yes	yes	yes	Word	MemNN	1 model	62.7	26M
FB5M	no	yes	no	no	Char	Ours	1 model	**70.3**	76K
FB5M	no	yes	no	no	Word	Ours	1 model	53.1	76K
FB5M	yes	yes	yes	yes	Word	MemNN	5 models	63.9	27M
FB5M	yes	yes	yes	yes	Word	MemNN	Subgraph	62.9	27M
FB5M	yes	yes	yes	yes	Word	MemNN	1 model	62.2	27M

Table 1: Experimental results on the SimpleQuestions dataset. MemNN results are from Bordes et al. (2015). WQ, SIQ and PRP stand for WebQuestions, SimpleQuestions and paraphrases from WikiAnswers.

$$P(e_j) = \frac{exp(\lambda cos(h_1, e_j))}{\sum_{i=1}^{n} exp(\lambda cos(h_1, e_i))}$$
$$P(p_k) = \frac{exp(\lambda cos(h_2, p_k))}{\sum_{i=1}^{m} exp(\lambda cos(h_2, p_i))}$$

where λ is a constant, h_1, h_2 are the hidden states of the LSTM at times $t = 1$ and $t = 2$, $e_1, ..., e_n$ are the entity embeddings, and $p_1, ..., p_m$ are the predicate embeddings. A similar likelihood function was used to train the semantic similarity modules proposed in Yih et al. (2014) and Yih et al. (2015), Palangi et al. (2016), Huang et al. (2013).

During inference, $e_1, ..., e_n$ and $p_1, ..., p_m$ are the embeddings of candidate entities and predicates. During training $e_1, ..., e_n$, $p_1, ..., p_m$ are the embeddings of the true entity and 50 randomly-sampled entities, and the true predicate and 50 randomly-sampled predicates, respectively.

3.4 Inference

For each question q, we generate a candidate set of entities and predicates, $\{e\}$ and $\{p\}$, and feed it through the model $f(q, \{e\}, \{p\})$. We then decode the most likely (entity, predicate) pair:

$$(e^*, p^*) = argmax_{e_i, p_j}(P(e_i) * P(p_j))$$

which becomes our semantic parse.

We use a similar procedure as the one described in Bordes et al. (2015) to generate candidate entities $\{e\}$ and predicates $\{p\}$. Namely, we take all entities whose English alias is a substring of the question, and remove all entities whose alias is a substring of another entity. For each English alias, we sort each entity with this alias by the number of facts that it has in the KB, and append the top 10 entities from this list to our set of candidate entities. All predicates p_j for each entity in our candidate entity set become the set of candidate predicates.

3.5 Learning

Our goal in learning is to maximize the joint likelihood $P(e_c) \cdot P(p_c)$ of predicting the correct entity e_c and predicate p_c pair from a set of randomly sampled entities and predicates. We use back-propagation to learn all of the weights in our model.

All the parameters of our model are learned jointly without pre-training. These parameters include the weights of the character-level embeddings, CNNs, and LSTMs. Weights are randomly initialized before training. For the i^{th} layer in our network, each weight is sampled from a uniform distribution between $-\frac{1}{|l^i|}$ and $\frac{1}{|l^i|}$, where $|l^i|$ is the number of weights in layer i.

4 Dataset and Experimental Settings

We evaluate the proposed model on the SimpleQuestions dataset (Bordes et al., 2015). The dataset consists of 108,442 single-relation questions and their corresponding (*topic entity*, *predicate*, *answer entity*) triples from Freebase. It is split into 75,910 train, 10,845 validation, and 21,687 test questions. Only 10,843 of the 45,335 unique words in entity aliases and 886 out of 1,034 unique predicates in the test set were present in the train set. For the proposed dataset, there are two evaluation settings, called FB2M and FB5M, respectively. The former uses a KB for candidate generation which is a sub-

set of Freebase and contains 2M entities, while the latter uses subset of Freebase with 5M entities.

In our experiments, the Memory Neural Networks (MemNNs) proposed in Bordes et al. (2015) serve as the baselines. For training, in addition to the 76K questions in the training set, the MemNNs use 3K training questions from WebQuestions (Berant et al., 2013), 15M paraphrases from WikiAnswers (Fader et al., 2013), and 11M and 12M automatically generated questions from the KB for the FB2M and FB5M settings, respectively. In contrast, our models are trained only on the 76K questions in the training set.

For our model, both layers of the LSTM-based question encoder have size 200. The hidden layers of the LSTM-based decoder have size 100, and the CNNs for entity and predicate embeddings have a hidden layer of size 200 and an output layer of size 100. The CNNs for entity and predicate embeddings use a receptive field of size 4, $\lambda = 5$, and $m = 100$. We train the models using RMSProp with a learning rate of $1e^{-4}$.

In order to make the input character sequence long enough to fill up the receptive fields of multiple CNN layers, we pad each predicate or entity using three padding symbols P, a special start symbol, and a special end symbol. For instance, $Obama$ would become $S_{start}PPPObamaPPPS_{end}$. For consistency, we apply the same padding to the questions.

5 Results

5.1 End-to-end Results on SimpleQuestions

Following Bordes et al. (2015), we report results on the SimpleQuestions dataset in terms of SQ accuracy, for both FB2M and FB5M settings in Table 1. SQ accuracy is defined as the percentage of questions for which the model generates a correct KB query (i.e., both the topic entity and predicate are correct). Our single character-level model achieves SQ accuracies of 70.9% and 70.3% on the FB2M and FB5M settings, outperforming the previous state-of-art results by 8.2% and 6.4%, respectively. Compared to the character-level model, which only has 1.2M parameters, our word-level model has 19.9M parameters, and only achieves a best SQ accuracy of 53.9%. In addition, in contrast to previous work, the OOV issue is much more severe for our word-level model, since we use no data augmentation to cover entities unseen in the train set.

5.2 Ablation and Embedding Experiments

We carry out ablation studies in Sections 5.2.1 and 5.2.2 through a set of *random-sampling* experiments. In these experiments, for each question, we randomly sample 200 entities and predicates from the test set as noise samples. We then mix the gold entity and predicate into these negative samples, and evaluate the accuracy of our model in predicting the gold predicate or entity from this mixed set.

5.2.1 Character-Level vs. Word-Level Models

We first explore using word-level models as an alternative to character-level models to construct embeddings for questions, entities and predicates.

Both word-level and character-level models perform comparably well when predicting the predicate, reaching an accuracy of around 80% (Table 3). However, the word-level model has considerable difficulty generalizing to unseen entities, and is only able to predict 45% of the entities accurately from the mixed set. These results clearly demonstrate that the OOV issue is much more severe for entities than predicates, and the difficulty word-level models have when generalizing to new entities.

In contrast, character-level models have no such issues, and achieve a 96.6% accuracy in predicting the correct entity on the mixed set. This demonstrates that character-level models encode the semantic representation of entities and can match entity aliases in a KB with their mentions in natural language questions.

5.2.2 Depth Ablation Study

We also study the impact of the depth of neural networks in our model. The results are presented in Table 2. In the ablation experiments we compare the performance of a single-layer LSTM to a two-layer LSTM to encode the question, and a single-layer vs. two-layer CNN to encode the KB entries. We find that a two-layer LSTM boosts joint accuracy by over 6%. The majority of accuracy gains are a result of improved predicate predictions, possibly because entity accuracy is already saturated in this experimental setup.

# of LSTM Layers	# of CNN Layers	Joint Accuracy	Predicate Accuracy	Entity Accuracy
2	2	**78.3**	**80.0**	96.6
2	1	77.7	79.4	**96.8**
1	2	71.5	73.9	95.0
1	1	72.2	74.7	94.9

Table 2: Results for a random sampling experiment where we varied the number of layers used for convolutions and the question-encoding LSTM. We terminated training models after 14 epochs and 3 days on a GPU.

Embedding Type	Joint Accuracy	Predicate Accuracy	Entity Accuracy
Character	**78.3**	**80.0**	**96.6**
Word	37.6	78.8	45.5

Table 3: Results for a random sampling experiment where we varied the embedding type (word vs. character-level). We used 2 layered-LSTMs and CNNs for all our experiments. Our models were trained for 14 epochs and 3 days.

5.3 Attention Mechanisms

In order to further understand how the model performs question answering, we visualize the attention distribution over question characters in the decoding process. In each sub-figure of Figure 2, the x-axis is the character sequence of the question, and the y-axis is the attention weight distribution $\{\alpha_i\}$. The blue curve is the attention distribution when generating the entity, and green curve is the attention distribution when generating the predicate.

Interestingly, as the examples show, the attention distribution typically peaks at empty spaces. This indicates that the character-level model learns that a space defines an ending point of a complete linguistic unit. That is, the hidden state of the LSTM encoder at a space likely summarizes content about the character sequence before that space, and therefore contains important semantic information that the decoder needs to attend to.

Also, we observe that entity attention distributions are usually less sharp and span longer portions of words, such as "john" or "rutters", than predicate attention distributions (e.g., Figure 2a). For entities, semantic information may accumulate gradually when seeing more and more characters, while for predicates, semantic information will become clear only after seeing the complete word. For example, it may only be clear that characters such as "song by" refer to a predicate after a space, as opposed to the name of a song such as "song bye bye love" (Figures 2a, 2b). In contrast, a sequence of characters starts to become a likely entity after seeing an incomplete name such as "joh" or "rutt".

In addition, a character-level model can identify entities whose English aliases were never seen in training, such as "phrenology" (Figure 2d). The model apparently learns that words ending with the suffix "nology" are likely entity mentions, which is interesting because it reads in the input one character at a time.

Furthermore, as observed in Figure 2d, the attention model is capable of attending disjoint regions of the question and capture the mention of a predicate that is interrupted by entity mentions. We also note that predicate attention often peaks at the padding symbols after the last character of the question, possibly because sentence endings carry extra information that further help disambiguate predicate mentions. In certain scenarios, the network may only have sufficient information to build a semantic representation of the predicate after being ensured that it reached the end of a sentence.

Finally, certain words in the question help identify both the entity and the predicate. For example, consider the word "university" in the question "What type of educational institution is eastern new mexico university" (Figure 2c). Although it is a part of the entity mention, it also helps disambiguate the predicate. However, previous semantic parsing-based QA approaches (Yih et al., 2015; Yih et al., 2014) assume that there is a clear separation between predicate and entity mentions in the question. In contrast, the proposed model does not need to make this hard categorization, and attends the word "university" when predicting both the entity and predicate.

6 Error Analysis

We randomly sampled 50 questions where the best-performing model generated the wrong KB query and categorized the errors. For 46 out of the 50 examples, the model predicted a predicate with a very similar alias to the true predicate, i.e. "/music/release/track" vs. "/music/release/track_list". For 21 out of the 50 examples, the model predicted the wrong entity, e.g., "Album" vs. "Still Here" for the question "What type of album is still here?". Finally, for 18 of the 50 examples, the model predicted the wrong entity and predicate, i.e. ("Play", "/freebase/equivalent_topic/equivalent_type") for the question "which instrument does amapola cabase play?" Training on more data, augmenting the negative sample set with words from the question that are not an entity mention, and having more examples that disambiguate between similar predicates may ameliorate many of these errors.

7 Conclusion

In this paper, we proposed a new character-level, attention-based encoder-decoder model for question answering. In our approach, embeddings of questions, entities, and predicates are all jointly learned to directly optimize the likelihood of generating the correct KB query. Our approach improved the state-of-the-art accuracy on the SimpleQuestions benchmark significantly, using much less data than previous work. Furthermore, thanks to character-level modeling, we have a compact model that is robust to unseen entities. Visualizations of the attention distribution reveal that our model, although built on character-level inputs, can learn higher-level semantic concepts required to answer a natural language question with a structured KB. In the future we want extend our system to multi-relation questions.

8 Acknowledgements

We thank the anonymous reviewers, Luke Zettlemoyer, Yejin Choi, Joel Pfeiffer, and members of the UW NLP group for helpful feedback on the paper.

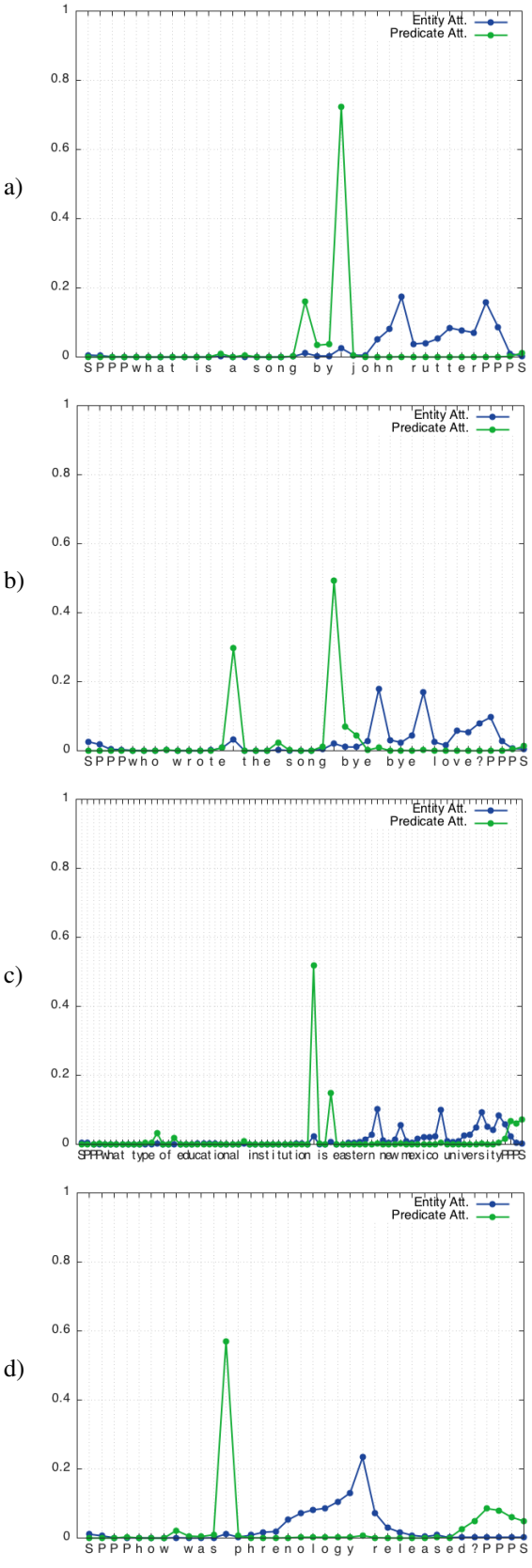


Figure 2: Attention distribution over outputs of a left-to-right LSTM on question characters.

References

[Amodei et al.2015] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan C. Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[Berant and Liang2014] J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

[Berant et al.2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Bishan Yang2015] Xiaodong He Jianfeng Gao Li Deng Bishan Yang, Scott Wen-tau Yih. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May.

[Bordes et al.2014a] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October. Association for Computational Linguistics.

[Bordes et al.2014b] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8724*, ECML PKDD 2014, pages 165–180, New York, NY, USA. Springer-Verlag New York, Inc.

[Bordes et al.2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *Proc. NIPS*.

[Chung et al.2015] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2067–2075.

[Chung et al.2016] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

[Dai et al.2016] Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *ACL*.

[dos Santos and Gatti2014] Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

[dos Santos et al.2015] Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

[dos Santos2014] Cicero dos Santos. 2014. Think positive: Towards twitter sentiment analysis from scratch. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 647–651, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

[Fader et al.2013] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Fader et al.2014] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 1156–1165.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

[Huang et al.2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

[Klein et al.2003] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language*

learning at HLT-NAACL 2003-Volume 4, pages 180–183. Association for Computational Linguistics.

[Ljubešić et al.2014] Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2014. Standardizing tweets with character-level machine translation. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 8404*, CICLing 2014, pages 164–175, New York, NY, USA. Springer-Verlag New York, Inc.

[Palangi et al.2016] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707.

[Santos and Zadrozny2014] Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

[Shen et al.2014] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. CIKM, November.

[Sukhbaatar et al.2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Tang and Mooney2001] Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477. Springer.

[Venugopalan et al.2015] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1494–1504, Denver, Colorado, May–June. Association for Computational Linguistics.

[Vinyals et al.2015] Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings.

[Yih et al.2014] Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*. Association for Computational Linguistics, June.

[Yih et al.2015] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*. ACL Association for Computational Linguistics, July.

[Zaremba and Sutskever2014] Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

[Zhang et al.2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Learning to Generate Textual Data

Guillaume Bouchard[†‡*] and **Pontus Stenetorp**[†*] and **Sebastian Riedel**[†]
{g.bouchard,p.stenetorp,s.riedel}@cs.ucl.ac.uk
[†]Department of Computer Science, University College London
[‡]Bloomsbury AI

Abstract

To learn text understanding models with millions of parameters one needs massive amounts of data. In this work, we argue that generating data can compensate for this need. While defining generic data generators is difficult, we propose to allow generators to be "weakly" specified in the sense that a set of parameters controls how the data is generated. Consider for example generators where the example templates, grammar, and/or vocabulary is determined by this set of parameters. Instead of manually tuning these parameters, we learn them from the limited training data at our disposal. To achieve this, we derive an efficient algorithm called GENERE that jointly estimates the parameters of the model and the undetermined generation parameters. We illustrate its benefits by learning to solve math exam questions using a highly parametrized sequence-to-sequence neural network.

1 Introduction

Many tasks require a large amount of training data to be solved efficiently, but acquiring such amounts is costly, both in terms of time and money. In several situations, a human trainer can provide their domain knowledge in the form of a generator of virtual data, such as a negative data sampler for implicit feedback in recommendation systems, physical 3D rendering engines as a simulator of data in a computer vision system, simulators of physical processes to solve science exam question, and math problem generators for automatically solving math word problems.

Domain-specific data simulators can generate an arbitrary amount of data that can be treated exactly the same way as standard observations, but since they are virtual, they can also be seen as regularizers dedicated to the task we want to solve (Scholkopf and Smola, 2001). While simple, the idea of data simulation is powerful and can lead to significantly better estimations of a predictive model because it prevents overfitting. At the same time it is subject to a strong model bias, because such data generators often generate data that is different from the observed data.

Creating virtual samples is strongly linked to transfer learning when the task to transfer is correlated to the objective (Pan and Yang, 2010). The computer vision literature adopted this idea very early through the notion of *virtual samples*. Such samples have a natural interpretation: by creating artificial perturbations of an image, its semantics is likely to be unchanged, i.e. training samples can be rotated, blurred, or slightly cropped without changing the category of the objects contained in the image (Niyogi et al., 1998).

However, for natural language applications the idea of creating invariant transformations is difficult to apply directly, as simple meaning-preserving transformations – such as the replacement of words by their synonyms or active-passive verb transformations – are quite limited. More advanced meaning-preserving transformations would require an already good model that understands natural language. A more structure-driven approach is to build top-down generators, such as probabilistic grammars, with a much wider coverage of linguistic phe-

[*] Contributed equally to this work.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1608–1616,
Austin, Texas, November 1-5, 2016.

nomena. This way of being able to leverage many years of research in computational linguistics to create good data generators would be a natural and useful reuse of scientific knowledge, and better than blindly believing in the current trend of "data takes all".

While the idea of generating data is straightforward, one could argue that it may be difficult to come up with good generators. What we mean by a good generator is the ability to help predicting test data when the model is trained on the generated data. In this paper, we will show several types of generators, some contributing more than others in their ability to generalize to unseen data. When designing a good generator there are several decisions one must make: should we generate data by modifying existing training samples, or "go wild" and derive a full probabilistic context-free grammar that could possibly generate unnatural examples and add noise to the estimator? While we do not arrive at a specific framework to build programs that generate virtual data, in this work we assume that a domain expert can easily write a program in a programming language of her choice, leaving some generation parameters unspecified. In our approach these unspecified parameters are automatically learned, by selecting the ones most compatible with the model and the training data.

In the next section, we introduce GENERE, a generic algorithm that extends any gradient-based learning approach with a data generator that can be tuned while learning the model on the training data using stochastic optimization. In Section 2.2, we show how GENERE can be adapted to handle a (possibly non-differentiable) black-box sampler without requiring modifications to it. We also illustrate how this framework can be implemented in practice for a specific use case: the automatic solving of math exam problems. Further discussion is given in the concluding section.

2 Regularization Based on a Generative Model

As with any machine learning approach, we assume that given the realisation of a variable $x \in \mathcal{X}$ representing the input, we want to predict the distribution of a variable $y \in \mathcal{Y}$ representing the output. The goal is to find this predictive distribution by learning it from examples $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$.

Building on the current success in the application of deep learning to NLP, we assume that there exists a good model family $\{f_\theta, \theta \in \Theta\}$ to predict y given x, where θ is an element of the parameter space Θ. For example, the stacked LSTM encoder-decoder is a general purpose model that has helped to improve results on relatively complex tasks, such as machine translation (Sutskever et al., 2014), syntactic parsing (Vinyals et al., 2014), semantic parsing (Dong and Lapata, 2016) and textual entailment (Rocktäschel et al., 2016).

For many applications, the amount of training data is too small or too costly to acquire. We hence look for alternative ways to regularize the model so that we can achieve good performance using few data points.

Let $p_\theta(y|x)$ be the target prediction model. Given the training dataset $\mathcal{D}$, the penalized maximum likelihood estimator is obtained by $\min_{\theta\in\Theta} \mathcal{L}(\theta)$ where:

$$\mathcal{L}(\theta) := \ell(\theta) + \lambda\Omega(\theta) \ . \tag{1}$$

where $\ell(\theta) := -\frac{1}{n}\sum_{i=1}^n \log p_\theta(y_i|x_i) = \mathbb{E}_{\hat{P}}[\log p_\theta(y|x)]$ is the negative log-likelihood. Here, $\Omega(\theta)$ is a regularizer that prevents over-fitting, $\lambda \in \mathbb{R}$ the regularization parameter that can be set by cross-validation, and $\hat{P}$ is the empirical distribution. Instead of using a standard regularizer Ω – such as the squared norm or the Lasso penalty which are domain-agnostic, – in this paper we propose to use a generative model to regularize the estimator.

Domain knowledge A natural way to inject background knowledge is to define a generative model that simulates the way the data is generated. In text understanding applications, such generative models are common and include probabilistic context-free grammars (PCFG) and natural language generation frameworks (e.g. SimpleNLG (Gatt and Reiter, 2009)). Let $P_\gamma(x, y)$ be such a generative model parametrized by a continuous parameter vector $\gamma \in \Gamma$, such as the concatenation of all the parameters of the production rules in a PCFG. One important difference between the discriminative and the generative probability distributions is that the in-

ference problem of y given x might be intractable[1] for the generative model, even if the joint model can be computed efficiently.

In this work, we use the following regularizer:

$$\Omega(\theta) \quad := \quad \min_{\gamma \in \Gamma} \mathbb{E}_{P_\gamma(x,y)} \left[\log \left(\frac{P_\gamma(y|x)}{p_\theta(y|x)} \right) \right] \quad . (2)$$

This regularizer makes intuitive sense as it corresponds to the smallest possible Kullback-Leibler divergence between the generative and discriminative models. We can see that if the generator p_γ is close to the distribution that generates the test data, the method can potentially yield good performance. However, in practice, γ is unknown and difficult to set. In this work, we focus on several techniques that can be used to estimate the generative parameter vector γ on the training data, making the regularizer data-dependent.

Minimizing the objective from Equation (1) is equivalent to minimize the following function over $\Theta \times \Gamma$:

$$\mathcal{L}(\theta, \gamma) \quad := \quad \ell(\theta) + \lambda \mathbb{E}_{P_\gamma(x,y)} \left[\log \left(\frac{p_\gamma(y|x)}{p_\theta(y|x)} \right) \right] \quad .$$

This estimator is called GENERE for *Generative Regularization* and can be viewed as a Generative-Discriminative Tradeoff estimator (GDT (Bouchard and Triggs, 2004)) that smoothly interpolates between a purely un-regularized discriminative model when $\lambda = 0$ and a generative model when λ tends to infinity.

2.1 The GENERE Algorithm

The objective $\mathcal{L}(\theta, \gamma)$ can also be written as an expectation under a mixture distribution $\tilde{P}_\gamma := \frac{1}{1+\lambda}\hat{P} + \frac{\lambda}{1+\lambda}P_\gamma$. The two components of this mixture are the empirical data distribution $\hat{P}$ and the generation distribution P_γ. The final objective is penalized by the entropy of the the generation $\mathcal{H}(\gamma) := \mathbb{E}_{P_\gamma}[\log p_\gamma(y|x)]$:

$$\mathcal{L}(\theta, \gamma) = -(1+\lambda)\mathbb{E}_{\tilde{P}_\gamma}[\log p_\theta(y|x)] - \lambda \mathcal{H}(\gamma) \quad . \quad (3)$$

This objective can be minimized using stochastic gradient descent by sampling real data or generated data according to the proportions $\frac{1}{1+\lambda}$ and $\frac{\lambda}{1+\lambda}$, respectively. The pseudocode is provided in Algorithm 1. It can be viewed as a variant of the REINFORCE algorithm which is commonly used in Reinforcement Learning (Williams, 1988) using the policy gradient. It is straightforward to verify that at each iteration, GENERE computes a noisy estimate of the exact gradient of the objective function $\mathcal{L}(\theta, \gamma)$ with respect to the model parameters θ and the generation parameters[2] γ.

An important quantity introduced in Algorithm 1 is the baseline value μ that approximates the average log-likelihood of a point sampled according to $\tilde{P}_\gamma$. Since it is unknown in general, an average estimate is obtained using a geometric averaging scheme with a coefficient α that is typically set to 0.98.

Algorithm 1 The GENERE Algorithm

Require: $\hat{P}$: real data sampler
Require: P_γ: parametric data generator
Require: λ: generative regularization strength
Require: η: learning rate
Require: α: baseline smoothing coefficient
1: Initialize parameters θ, sampling coefficients γ and baseline μ
2: **for** $t = 1, 2, \cdots$ **do**
3: $\quad x, y \sim \frac{1}{1+\lambda}\hat{P} + \frac{\lambda}{1+\lambda}P_\gamma$
4: $\quad g_\theta \leftarrow \nabla_\theta \log p_\theta(y|x)$
5: $\quad g_\gamma \leftarrow (\log p_\theta(y|x) - \mu) \nabla_\gamma \log p_\gamma(x, y)$
6: $\quad (\theta, \gamma) \leftarrow (\theta, \gamma) - \eta(g_\theta, g_\gamma)$
7: $\quad \mu \leftarrow \alpha\mu + (1-\alpha) \log p_\theta(y|x)$
8: **end for**

Generative models: interpretable sampling, intractable inference Generative modeling is natural because we can consider latent variables that add interpretable meaning to the different components of the model. For example, in NLP we can define the latent variable as being the relations that are mentioned in the sentence.

[1] Even if tractable, inference can be very costly: for example, PCFG decoding can be done using dynamic programming and has a cubic complexity in the length of the decoded sentence, which is still too high for some applications with long sentences.

[2] The derivative with respect to γ, leads to Algorithm 1 with $\mu = -1$, but the algorithm is also valid for different values of μ as the average gradient remains the same if we add a multiple of $\nabla_\gamma \log p_\gamma(x, y)$ to the gradient g_γ (line 5 in Algorithm 1) which has zero-mean on average. Choosing μ to be the average of the past gradient enables the gradient to have a lower variance.

We could consider two main types of approaches to choose a good structure for a parameterized data generator:

- Discrete data structure: we can use efficient algorithms, such as dynamic programming to perform sampling and which can propagate the gradient
- Continuous distribution: having a continuous latent variable enables easy handling of correlations across different parts of the model.

It is often laborious to design data generators which can return the probability of the samples it generates[3], as well as the gradient of this probability with respect to the input parameters γ.

In the next section, we show how to alleviate this constraint by allowing any data-generating code to be used with nearly no modification.

2.2 GENERE with a Black Box Sampler

Let us assume the data generator is a black box that takes a K-dimensional seed vector as input and outputs an input-output sample x, y. To enable GENERE to be applied without having to modify the code of data generators. The trick is to use a existing generator with parameter γ, and to create a new generator that essentially adds noise to γ. This noise will be denoted $\Delta \in \Gamma$. We used the following data generation process:

1. Sample a Gaussian seed vector $\Delta \sim \mathcal{N}(0, I)$
2. Use the data generator G_z with seed value $z := \Delta + \gamma$ to generate an input-output sample (x, y).

This two-step generation procedure enables the gradient information to be computed using the density of a Gaussian distribution. The use of a standardized centered variable for the seed is justified by the fact that the parametrization of G_z takes into account possible shifts and rescaling. Formally, this is equivalent to Algorithm 1 with the following generative model:

$$p_\gamma(x, y) = \mathbb{E}_{\Delta \sim \mathcal{N}(0,I)} \left[g_{\gamma+\Delta}(x, y) \right] \quad (4)$$

[3]This difficulty comes from the fact that generators may be using third-party code, such as rendering engines, grammars sampler, and deterministic operations such a sorting that are non-differentiable.

where g_z is the density of the black-box data generator G_z for the seed value $z \in \mathbb{R}^K$. Ideally, the second data generator that takes z as an input and generates the input/output pair (x, y) should be close to a deterministic function in order to allocate more uncertainty in the trainable part of the model which corresponds to the Gaussian distribution.[4]

Learning The pseudo-code for the Black Box GENERE variant is shown in Algorithm 2. It is similar to Algorithm 1, but here the sampling phase is decomposed into the two steps: A random Gaussian variable sampling followed by the black box sampling of generators.

Algorithm 2 Black Box GENERE

Require: $\hat{P}$: real data sampler
Require: $G(\gamma)$: black box data generator
Require: λ: generative regularization strength
Require: η_γ, η_θ: learning rates
Initialize parameters θ, sampling coefficients γ and baseline μ
for $t = 1, 2, \cdots$ **do**
  if $\frac{1}{1+\lambda} > \mathcal{U}([0, 1])$ **then**
    $x, y \sim \hat{P}$
  else
    $\Delta \sim \mathcal{N}(\mathbf{0}, I)$
    $x, y \sim G_{\gamma+\Delta}$
    $\gamma \leftarrow \gamma - \eta_\gamma (\log p_\theta(y|x) - \mu)\Delta$
  end if
  $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \log p_\theta(y|x)$
  $\mu \leftarrow \alpha\mu + (1 - \alpha) \log p_\theta(y|x)$
end for

3 Application to Encoder-Decoder

In this section, we show that the GENERE algorithm is well suited to tune data generators for problems that are compatible with the encoder-decoder architecture commonly used in NLP.

3.1 Mixture-based Generators

In the experiments below, we consider mixture-based generators with known components but unknown mixture proportions. Formally, we

[4]What we mean by deterministic is that the black-box sampler has the form $\delta\{f(\Delta + \gamma) = (x, y)\}$, where δ is the indicator function.

parametrize the proportions using a softmax link $\sigma(t) := \exp(t_k)/\sum_{k'=1}^{K}\exp(t_{k'})$. In other words, the data generator distribution is:

$$p_\gamma(x,y) = \sum_{k=1}^{K} \sigma_k(\gamma+\Delta)p_k(x,y),$$

where $p_k(x,y)$ are data distributions, called *base generators*, that are provided by domain experts, and Δ is a K-dimensional centered Gaussian with an identity covariance matrix. This class of generator makes sense in practice, as we typically build multiple base generators $p_k(x,y)$, $k = 1,\cdots,K$, without knowing ahead of time which one is the most relevant. Then, the training data is used by the GENERE algorithm to automatically learn the optimal parameter γ that controls the contribution $\{\pi_k\}_{k=1}^{K}$ of each of the base generators, equal to $\pi_k := \mathbb{E}_{\Delta\sim\mathcal{N}(\mathbf{0},I)}\left[\sigma_k(\gamma+\Delta)\right]$.

3.2 Synthetic Experiment

In this section, we illustrate how GENERE can learn to identify the correct generator, when the data generating family is a mixture of multiple data generators and only one of these distributions – say p_1 – has been used to generate the data. The other distributions $(p_2,\cdots,p_K)$ are generating input-output data samples (x,y) with different distributions.

We verified that the algorithm correctly identifies the correct data distribution, and hence leads to better generalization performances than what the model achieves without the generator.

In this illustrative experiment, a simple text-to-equation translation problem is created, where inputs are sentences describing an equation such as "compute one plus three minus two", and outputs are symbolic equations, such as "X = 1 + 3 - 2". Numbers were varying between -20 and 20, and equations could have 2 or 3 numbers with 2 or 3 operations.

As our model, we used a 20-dimensional sequence-to-sequence model with LSTM recurrent units. The model was initialized using 200 iterations of standard gradient descent on the log-probability of the output. GENERE was run for 500 iterations, varying the fraction of real and generated samples from 0% to 100%. A ℓ_2 regularization of magnitude 0.1 was applied to the model. The baseline smoothing coefficient was set to 0.98 and the shrinkage parameter was set to 0.99. All the experiments were repeated 10 times and a constant learning rate of 0.1 was used.

Results are shown on Figure 1, where the average loss computed on the test data is plotted against the fraction of real data used during learning.

We can see that the best generalization performance is obtained when there is a balanced mix of real and artificial data, but the proportion depends on the amount of training data: on the left hand side, the best performance is obtained with generated data only, meaning that the number of training samples is so small that GENERE only used the training data to select the best base generator (the component p_1), and the best performance is attained using only generated data. The plot on the right hand side is interesting because it contains more training data, and the best performance is not obtained using only the generator, but with 40% of the real data, illustrating the fact that it is beneficial to jointly use real and simulated data during training.

3.3 Math word problems

To illustrate the benefit of using generative regularization, we considered a class of real world problems for which obtaining data is costly: learning to answer math exam problems. Prior work on this problem focuses on standard math problems given to students aged between 8 and 10, such as the following:[5]

> For Halloween Sarah received 66 pieces of candy from neighbors and 15 pieces from her older sister. If she only ate 9 pieces a day, how long would the candy last her?

The answer is given by the following equation: $\mathrm{X} = (66+15)/9$. Note that similarly to real world school exams, giving the final answer of (9 in this case) is not considered enough for the response to be correct.

The only publicly available word problem datasets we are aware of contain between 400 and 600 problems (see Table 2), which is not enough to properly train sufficiently rich models that capture the link between the words and the quantities involved in the problem.

[5] From the Common Core dataset (Roy and Roth, 2015)

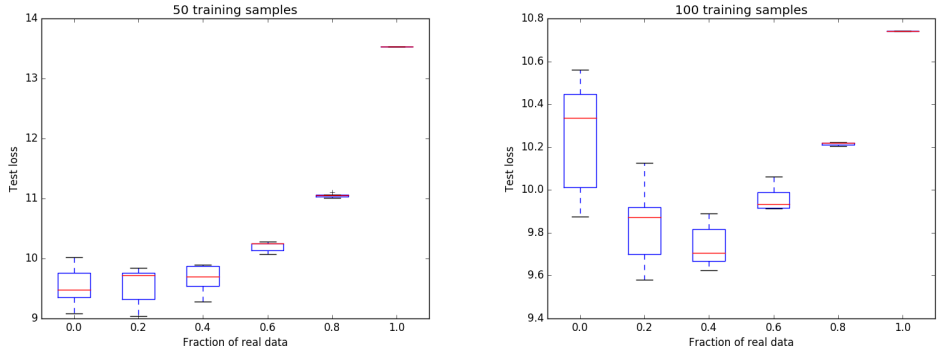


Figure 1: Test loss vs. fraction of real data used in GENERE on the text-to-equation experiment.

Sequence-to-sequence learning is the task of predicting an output sequence of symbols based on a sequence of input symbols. It is tempting to cast the problem of answering math exams as a sequence-to-sequence problem: given the sequence of words from the problem description, we can predict the sequence of symbols for the equation as output. Currently, the most successful models for sequence prediction are Recurrent Neural Nets (RNN) with non-linear transitions between states.

Treated as a translation problem, math word problem solving should be simpler than developing a machine translation model between two human languages, as the output vocabulary (the math symbols) is significantly smaller than any human vocabulary. However, machine translation can be learned on millions of pairs of already translated sentences, and such massive training datasets dwarf all previously introduced math exam datasets.

We used standard benchmark data from the literature. The first one, AI2, was introduced by Hosseini et al. (2014) and covers addition and subtraction of one or two variables or two additions scraped from two web pages. The second (IL), introduced by Roy et al. (2015), contains single operator questions but covers addition, subtraction, multiplication, and division, and was also obtained from two, although different from AI2, web pages. The last data set (CC) was introduced by Roy and Roth (2015) to cover combinations of different operators and was obtained from a fifth web page.

An overview of the equation patterns in the data is shown in Table 1. It should be noted that there are sometimes numbers mentioned in the problem description that are not used in the equation.

AI2	IL	CC
$X + Y$	$X + Y$	$X + Y - Z$
$X + Y + Z$	$X - Y$	$X * (Y + Z)$
$X - Y$	$X * Y$	$X * (Y - Z)$
	X/Y	$(X + Y)/Z$
		$(X - Y)/Z$

Table 1: Patterns of the equations seen in the datasets for one permutation of the placeholders.

	AI2	IL	CC
Train	198	214	300
Dev	66	108	100
Test	131	240	200
Total	395	562	600

Table 2: Math word problems dataset sizes.

As there are no available train/dev/test splits in the literature we introduced such splits for all three data sets. For AI2 and CC, we simply split the data randomly and for IL we opted to maintain the clusters described in Roy and Roth (2015). We then used the implementation of Roy and Roth (2015) provided by the authors, which is the current state-of-the-art for all three data sets, to obtain results to compare our model against. The resulting data sizes are shown on Table 2. We verified that there are no duplicate problems, and our splits and a fork of the baseline implementation are available online.[6]

3.4 Development of the Generator

Generators were organized as a set of 8 base generators p_k, summarized in Table 4. Each base generator

[6] https://github.com/ninjin/roy_and_roth_2015

John sprints to William's apartment. The distance is 32 yards from John's apartment to William's apartment. It takes John 2 hours to at the end get there. How fast did John go?	32 / 2
Sandra has 7 erasers. She grasps 7 more. The following day she grasps 18 whistles at the local supermarket. How many erasers does Sandra have in all?	7 + 7
A pet store had 81 puppies In one day they sold 41 of them and put the rest into cages with 8 in each cage. How many cages did they use?	(81 - 41) / 8
S1 V1 Q1 O1 C1 S1(pronoun) V2 Q2 of O1(pronoun) and V2 the rest into O3(plural) with Q3 in each O3. How many O3(plural) V3?	(Q1 - Q2) / Q3

Table 3: Examples of generated sentences (first 3 rows). The last row is the template used to generate the 3rd example where brackets indicate modifiers, symbols starting with 'S' or 'O' indicate a noun phrase for a subject or object, symbols with 'V' indicate a verb phrase, and symbols with 'Q' indicate a quantity. They are identified with a number to match multiple instances of the same token.

has several functions associated with it. The functions were written by a human over 3 days of full-time development. The first group of base generators is only based on the type of symbol the equation has, the second group is the pair (#1, #2) to represent equations with one or two symbols. Finally, the last two generators are more experimental as they correspond to simple modifications applied to the available training data. The Noise 'N' generator picks one or two random words from a training sample to create a new (but very similar) problem. Finally, the 'P' generator is based on computing the statistics of the words for the same question pattern (as one can see in Table 1), and generates data using simple biased word samples, where words are distributed according to their average positions in the training data (positions are computed relatively to the quantities appearing in the text, i.e. "before the first number", "between the 1st and the 2nd number", etc.).

3.5 Implementation Details

We use a standard stacked RNN encoder-decoder (Sutskever et al., 2014), where we varied the recurrent unit between LSTM and GRU (Cho et al., 2014), stack depth from 1 to 3, the size of the hidden states from 4 to 512, and the vocabulary threshold size. As input to the encoder, we downloaded pre-trained 300-dimensional embeddings trained on Google News data using the word2vec software (Mikolov et al., 2013). The development data was used to tune these parameters before performing the evaluation on the test set. We obtained the best performances with a single stack, GRU units, and a hidden state size of 256.

	The problem...
+	contains at least one addition
-	contains at least one subtraction
*	contains at least one multiplication
/	contains at least one division
1	has a single mathematical operation
2	has a couple of mathematical operations
N	is a training sample with words removed
P	is based on word position frequencies

Table 4: The base generators to create math exam problems.

The optimization algorithm was based on stochastic gradient descent using Adam as an adaptive step size scheme (Kingma and Ba, 2014), with mini-batches of size 32. A total of 256 epochs over the data was used in all the experiments.

To evaluate the benefit of learning the data generator, we used a hybrid method as a baseline where a fraction of the data is real and another fraction is generated using the default parameters of the generators (i.e. a uniform distribution over all the base generators). The optimal value for this fraction obtained on the development set was 15% real data, 85% generated data. For GENERE, we used a fixed

	AI2	IL	CC	Avg.
RR2015	82.4	75.4	55.5	71.1
100% Data	72.5	53.7	95.0	73.7
100% Gen	60.3	51.2	92.0	67.8
85%Gen + 15%Data	74.0	55.4	97.5	75.6
GENERE	77.9	56.7	98.5	77.7

Table 5: Test accuracies of the math-exam methods on the available datasets averaged over 10 random runs.

size learning rate of 0.1, the smoothing coefficient was selected to be 0.5, and the shrinkage coefficient to be 0.99.

We also compared our approach to the publicly available math exam solver RR2015 (Roy and Roth, 2015). This method is based on a combination of template-based features and categorizers. The accuracy performance was measured by counting the number of times the equation generated the correct results, so that 10 + 7 and 7 + 10 would both be considered to be correct. Results are shown on Table 5.

We can see that there is a large difference in performance between RR2015 and the RNN-based encoder-decoder approach. While their method seems to be very good on some datasets, it fails on CC, which is the dataset in which one needs two equations involving parentheses. On average, the trend is the following: using data only does not succeed in giving good results, and we can see that with generated data we are performing better already. This could be explained by the fact that the generators' vocabulary has a good overlap with the vocabulary of the real data. However, mixing real and generated data improves performance significantly. When GENERE is used, the sampling is tuned to the problem at hand and give better generalization performance.

To understand if GENERE learned a meaningful data generator, we inspected the coefficients $\gamma_1, \cdots, \gamma_8$ that are used to select the 8 data generators described earlier. This is shown is Figure 2.

The results are quite surprising at first sight: the AI2 dataset only involves additions and subtractions, but GENERE selects the generator generating divisions as the most important. Investigating, we noted that problems generated by the division generator were reusing some lexical items that were present in AI2, making the vocabulary very close to the problems in AI2, even if it does not cover division. We can also note that the differences in proportions are quite small among the 4 symbols $+, -, *$ and $/$ across all the datasets. We can also clearly see that the noisy generator 'N' and 'P' are not very relevant in general. We explain this by the fact that the noise induced by these generators is too artificial to generate relevant data for training. Their likelihood on the model trained on real data remains small.

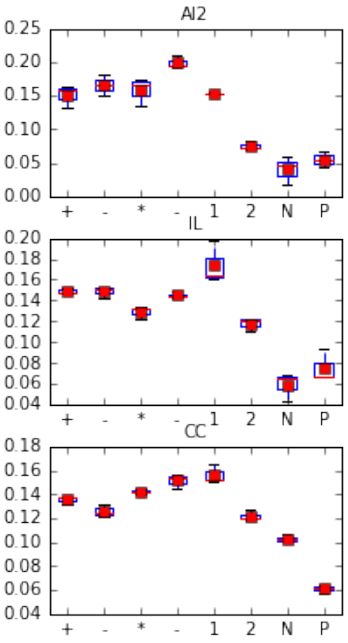


Figure 2: Base generators proportions learned by GENERE.

4 Conclusion

In this work, we argued that many problems can be solved by high-capacity discriminative probabilistic models, such as deep neural nets, at the expense of a large amount of required training data. Unlike the current trend which is to reduce the size of the model, or to define features well targeted for the task, we showed that we can completely decouple the choice of the model and the design of a data generator. We proposed to allow data generators to be "weakly" specified, leaving the undetermined coefficients to be learned from data. We derived an efficient algorithm called GENERE, that jointly estimates the parameters of the model and the undetermined sampling coefficients, removing the need for costly cross-validation. While this procedure could be viewed as a generic way of building informative priors, it does not rely on a complex integration procedure such as Bayesian optimization, but corresponds to a simple modification of the standard stochastic optimization algorithms, where the sampling alternates between the use of real and generated data. While the general framework assumes that the sampling distribution is differentiable with respect to its learnable parameters, we proposed a Gaussian integration trick that does not require the

data generator to be differentiable, enabling practitioners to use *any* data sampling code, as long as the generated data resembles the real data.

We also showed in the experiments, that a simple way to parametrize a data generator is to use a mixture of base generators, that might have been derived independently. The GENERE algorithm learns automatically the relative weights of these base generators, while optimizing the original model. While the experiments only focused on sequence-to-sequence decoding, our preliminary experiments with other high-capacity deep neural nets seem promising.

Another future work direction is to derive efficient mechanisms to guide the humans that are creating the data generation programs. Indeed, there is a lack of generic methodology to understand where to start and which training data to use as inspiration to create generators that generalize well to unseen data.

Acknowledgments

We would like to thank Subhro Roy for helping us run his model on our new data splits. We are very thankful to Thomas Demeester, Johannes Welbl and Matko Bošnjak for their valuable feedback. Lastly, we would like to thank the three anonymous reviewers for their helpful comments and feedback.

This work was supported by a Marie Curie Career Integration Award and an Allen Distinguished Investigator Award.

References

Guillaume Bouchard and Bill Triggs. 2004. The trade-off between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pages 721–728.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar, October. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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

P. Niyogi, F. Girosi, and T. Poggio. 1998. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 86(11):2196–2209, Nov.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. Association for Computational Linguistics.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Bernhard Scholkopf and Alexander J Smola. 2001. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

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

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *CoRR*, abs/1412.7449.

Ronald J Williams. 1988. On the use of backpropagation in associative reinforcement learning. In *Neural Networks, 1988., IEEE International Conference on*, pages 263–270. IEEE.

A Theme-Rewriting Approach for Generating Algebra Word Problems

Rik Koncel-Kedziorski
Ioannis Konstas
Luke Zettlemoyer
Hannaneh Hajishirzi
University of Washington
kedzior@uw.edu, {ikonstas, lsz, hannaneh}@cs.washington.edu

Abstract

Texts present coherent stories that have a particular theme or overall setting, for example science fiction or western. In this paper, we present a text generation method called *rewriting* that edits existing human-authored narratives to change their theme without changing the underlying story. We apply the approach to math word problems, where it might help students stay more engaged by quickly transforming all of their homework assignments to the theme of their favorite movie without changing the math concepts that are being taught. Our rewriting method uses a two-stage decoding process, which proposes new words from the target theme and scores the resulting stories according to a number of factors defining aspects of syntactic, semantic, and thematic coherence. Experiments demonstrate that the final stories typically represent the new theme well while still testing the original math concepts, outperforming a number of baselines. We also release a new dataset of human-authored rewrites of math word problems in several themes.

1 Introduction

Storytelling is the complex activity of expressing a plot, its events and participants in words meaningful to an audience. Automatic storytelling systems can be used for customized sport commentaries, enriching video games with personalized or dynamic plot-lines (Barros and Musse, 2007), or providing customized learning materials which meet each individual student's needs and interests (Bartlett, 2004). In this paper, we focus on generating narrative-style math word problems (Figure 1) and demonstrate that it is possible to design an algorithm that can automatically change the overall theme of a text without changing its underlying story, for example to create more engaging homework that is in the theme of a student's favorite movie.

Jim walked 0.2 of a mile from school to David's house and 0.7 of a mile from David's house to his own house. How many miles did Jim walk in all?
Star Wars Uncle Owen walked 0.2 of a mile from hangar to Luke Skywalker's room and 0.7 of a mile from Luke Skywalker's room to his own room. How many miles did Uncle Owen walk in all?
Cartoon Finn squished 0.2 of a mile from cupboard to Melissa's dock and 0.7 of a mile from Melissa's dock to his own dock. How many miles did Finn squish in all?
Western Duane strolled 0.2 of a mile from barn to Madeline's camp and 0.7 of a mile from Madeline's camp to his own camp. How many miles did Duane stroll in all?

Figure 1: An example story and rewrites in 3 themes.

A math word problem is a coherent story that provides the student with good clues to the correct mathematical operations between the numerical quantities described therein. However, the particular *theme* of a problem, whether it be about collecting apples or traveling distances through space, can vary significantly so long as the correlation between the story and underlying equation is maintained. Students' success at solving a word problem is tied to their interest in the problem's theme (Renninger

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1617–1628,
Austin, Texas, November 1-5, 2016.

et al., 2002), and personalizing word problems increases student understanding, engagement, and performance in the problem solving process (Hart, 1996; Davis-Dorsey et al., 1991).

Motivated by this need for thematically diverse, highly coherent stories, we address the problem of *story rewriting*, or transforming human-authored stories into novel, coherent stories in a new theme. Rather than synthesizing first a story plot (McIntyre and Lapata, 2009; McIntyre and Lapata, 2010) or script (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016) from scratch, we instead begin from an existing story and iteratively edit it towards a thematically novel but –most crucially– semantically compatible story. This approach allows us to reuse much, but not all, of the syntactic and semantic structure of the original text, resulting in the creation of more coherent and solvable math word problems.

We define a theme to be a collection of reference texts, such as a movie script or series of books. Given a theme, the *rewrite* algorithm constructs new texts by substituting thematically appropriate words and phrases, as measured with automatic metrics over the theme text collection, for parts of the original texts. This process optimizes for a number of metrics of overall text quality, including syntactic, semantics, and discourse scores. It uses no hand crafted templates and requires no theme-specific tuning data, making it easy to apply for new themes in practice. Tables 4–6 show example stories generated from the rewrite system.

To evaluate performance, we collected a corpus of 450 rewrites of math word problems in Star Wars and Children's Cartoon themes via crowdsourcing.[1] Experiments with automated metrics and human evaluations demonstrate that the approach described here outperforms a number of baselines and can produce solvable problems in multiple different themes, even with no in-domain tuning.

2 Related Work

Our approach is related to the previous work in story generation (e.g., McIntyre and Lapata (2010)) and sentence rewriting (e.g., text simplification (Xu et al., 2016)), as reviewed in this section. It has three major differences from all these approaches: First, we focus on multi-sentence stories where preserving the coherence, discourse relations, and solvability is essential. Previous work mainly focuses on rewriting single sentences. Second, we build a theme from a text corpus and show how the stories can be adapted to new themes. Third, our method leverages the human-authored story to capture the semantic skeleton and the plot of the current story, rather than synthesizing the story plot. To our knowledge, we are the first to introduce a text rewriting formulation for story generation.

Story generation has been of long interest to AI researchers (Meehan, 1976; Lebowitz, 1987; Turner, 1993; Liu and Singh, 2002; Mostafazadeh et al., 2016). Recent methods in story generation first synthesize candidate plots for a story and then compile those plots into text. Li et al. (2013) use crowdsourcing to build plot graphs. McIntyre and Lapata (2009; 2010) address story generation through the automatic deduction and reassembly of scripts (Schank and Abelson, 1977), or structured representations of events and their participants, and causal relationships involved. Leveraging the automatic script learning methods of Chambers and Jurafsky (2009), McIntyre and Lapata (2010) learn candidate entity-centered plot graphs, or possible events involving the entity and an ordering between these events, with the use of a genetic algorithm. Then plots are compiled into stories through the use of a rule-based text surface realizer (Lavoie and Rambow, 1997) and reranked using a language model.

Polozov et al. (2015) automatically generate math word problems tailored to a student's interest using Answer Set Programming to satisfy a collection of pedagogical and narrative requirements. This method naturally produces highly coherent, personalized story problems that meet pedagogical requirements, at the expense of building the thematic ontologies and discourse constraints by hand.[2]

Additionally, there is related work in text simplification (Wubben et al., 2012; Kauchak, 2013; Zhu et al., 2010; Vanderwende et al., 2007; Woodsend and Lapata, 2011b; Hwang et al., 2015), sentence

[1] Data and code available at `https://gitlab.cs.washington.edu/kedzior/Rewriter/`.

[2] According to Polozov et al. (2015) building small thematic ontologies of types, relations, and discourse tropes (100-200 entries) for each of only 3 literary settings took 1-2 person months.

compression (Filippova and Strube, 2008; Rush et al., 2015), and paraphrasing (Ganitkevitch et al., 2013; Chen and Dolan, 2011; Ganitkevitch et al., 2011). All these tasks are focused on rewriting sentences under a predefined set of constraints, such as simplicity. Different rule-based and data-driven approaches are introduced by Petersen and Ostendorf (2007), Vickrey and Koller (2008), and Siddharthan (2004). Most data-driven approaches take advantage of machine translation techniques, use source-target sentence pairs, and learn rewrite operations (Yatskar et al., 2010; Woodsend and Lapata, 2011a), or use additional external paraphrasing resources (Xu et al., 2016).

Finally, this work is related to those on automatically solving math word problems. Specific topics include number word problems (Shi et al., 2015), logic puzzle problems (Mitra and Baral, 2015), arithmetic word problems (Hosseini et al., 2014; Roy and Roth, 2015), algebra word problems (Kushman et al., 2014; Zhou et al., 2015; Koncel-Kedziorski et al., 2015a; Roy et al., 2016), and geometry word problems (Seo et al., 2015; Seo et al., 2014). Several datasets of word problems are available (Koncel-Kedziorski et al., 2016; Huang et al., 2016), though none address the need for thematic text.

3 Problem Formulation

Our system takes as input a story s and a theme t, and outputs the best rewrite s^* from generated candidates S.

A theme t is defined as a textual corpus that describes a topic or a domain. This is an intentionally broad definition that allows a variety of textual resources to serve as themes. For example, the collection of all Science Fiction stories from the Project Gutenberg can be a theme, or the script of a single movie, or a sampling of fan fiction from the Internet. This flexibility adds to the utility of our work, as varying amounts of thematic text may be available.

The generated candidate s^* is the most *thematically* fit problem that is *syntactically* and *semantically* coherent given the original problem s and the new theme t. We represent a story in terms of the words it contains, so that $s = \{w_1, w_2, \dots\ w_n\}$ and

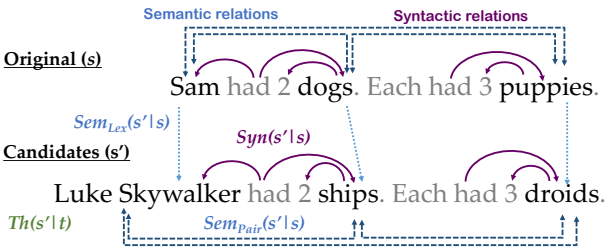


Figure 2: An overview of our method for scoring a candidate story s' given a human-authored story s and a theme t. Syn$(s'|s)$: compatibility of syntactic relations (purple arrows), Sem$_{pair}(s'|s)$: coherence of semantic relations (blue arrows), Sem$_{Lex}(s'|s)$: semantic mapping of individual words, and $Th(s'|s,t)$: thematicity.

$|s| = n$. The new story s' is defined as:

$$s' = \{f(w_1), f(w_2), \dots\ f(w_n)\}$$

where the function $f(w) : \mathcal{V}_o \rightarrow \mathcal{V}_t^K \cup \varnothing$, rewrites a word from the vocabulary of the original problem $\mathcal{V}_o$ to either a word, a trivial noun compound of length K (e.g., multi-word named entity) from the vocabulary of the the thematic vocabulary $\mathcal{V}_t$, or reduces to the empty symbol, i.e., omits the input word entirely; hence the length of s' can differ from that of the original problem.

Formally, our goal is to select the candidate $s' \in S$ by maximizing a scoring function $\mathcal{R}$ over thematic, syntactic and semantic constraints, subject to a set of parameters θ:

$$s^* = \arg\max_{s' \in S} \mathcal{R}(s'|s,t;\theta) \qquad (1)$$

In order to find the best story s^*, our problem reduces to generating candidate stories s' from the space of possible rewrites of the human-authored story s in a new theme t (Section 5). Since there are exponentially many rewrites, we follow a two-stage decoding approach: first we identify only the content words w_i in the input problem, and provide for each a list of the top-k most *salient* thematic candidate words and trivial noun compounds. We then search the space by progressively introducing more rewrites in the beam, and scoring them according to $\mathcal{R}$ (Section 4). Figure 2 shows the overview of the scoring function for a candidate sentence s'.

4 Scoring Stories

The scoring function $\mathcal{R}$ decomposes into three components, capturing aspects of syntactic compatibility, semantic coherence, and thematicity:

$$\begin{aligned}\mathcal{R}(s'|s,t;\theta) =& \alpha \times \text{Sem}(s'|s) \\ &+ \beta \times \text{Syn}(s'|s) \\ &+ \gamma \times \text{Th}(s'|s,t)\end{aligned} \quad (2)$$

The syntactic (Syn) and semantic (Sem) coherence components measure the coherence of the words in the new story s', as well as their compatibility to the syntactic and semantic relations in the original story s. On the other hand, thematicity (Th) scores the relevance and importance of words in the new story with respect to theme t.

We describe each of these components and the decoding process in the following sections.

4.1 Thematicity

Recall that a theme t is defined as a collection of documents that share a common topic, such as books in the science fiction genre, or scripts of horror movies. We define thematicity of a word w' as the measure of *salience*, or how discriminative that word is to a given theme.[3] For example, *robot* and *spaceship* are expected to be highly thematic with respect to Star Wars. In our setting we extend this definition to a candidate problem s' given s and t as:

$$\text{Th}(s'|s,t) = \sum_{i}^{|s|} Sal(w'_i, t) \quad (3)$$

where w'_i is a word from the candidate problem, and Sal is its salience score with respect to the theme. In the context of this work we argue that the thematic adaptation of the content words, i.e., nouns, verbs, named entities, and adjectives, plays the most important role in forming a new thematic problem. Therefore, we define their salience (except named entities) based on their tf-idf score over the theme t, and set it to zero for function words. Since named entities have relatively low frequencies in the theme corpus we set their salience to $1 - \frac{1}{c(w'_i)}$, where $c(w'_i)$ is the number of times w'_i occurs in the theme. In the example story in Figure 2 the thematicity score is derived as $Sal(\text{Luke Skywalker}) + Sal(\text{ships}) + Sal(\text{droids})$.

[3] We will be interchangeably referring to w' as either the word or the head of the multi-word noun compound that rewrites the equivalent word w in the original problem.

4.2 Syntactic compatibility

This work offers a new method for syntactic and discourse coherence based on preserving human-authored syntactic structure in generated text (hence our use of the term *rewriting*). The syntactic constructs in a document play a distinctive role in maintaining cohesion across sentences. We consider the human-authored syntax of the original story s as gold standard, and use it to score a candidate problem s' by considering how well the syntactic relations of s apply to s'.

Formally, given a dependency triple (w_i, w_j, l) from a parse of a sentence in s, we compute the likelihood for the corresponding triple (w'_i, w'_j, l) for w'_i, w'_j in s'. We define the syntactic score for all sentences in s' as:

$$\text{Syn}(s'|s) = \sum_{i,j,l|(w_i,w_j,l)\in \text{Dep}(s)} \mathcal{L}_{Dep}(w'_i, w'_j, l) \quad (4)$$

where $Dep(s)$ are the dependency parse trees for all sentences in s; $\mathcal{L}_{Dep}$ is a 3-gram language model over dependency triples which gives the likelihood of an arc label l being used between a pair of words (w'_i, w'_j). For example in Figure 2, the syntactic compatibility score includes dependency likelihoods of $\mathcal{L}_{Dep}(\text{ship}, 2, \textit{num})$, $\mathcal{L}_{Dep}(\text{had}, \text{ship}, \textit{dobj})$.

Therefore, the Syn function prefers stories s' that (a) have similar dependency structure to the original story s and (b) make use of a common syntactic configuration.

4.3 Semantic Coherence

The semantic coherence component expresses how well a candidate s' rewrites individual words and realizes the semantic relationships that exist in the human-authored story s. Ideally, we would like to preserve enough of the semantics of s in order to produce a coherent story s', yet we are populating s' with words taken from an unrelated theme. Therefore, we model the semantics of a story s' in terms of the lexical semantics contributed by individual words as well as semantic relationships that exist between its elements. Note that the relationships can

cross the sentence boundaries, promoting discourse coherence.

We decompose semantic relations in a story into a set of local, lexical relationships between pairs of words. Specifically, we consider semantic relations for noun-noun and verb-verb pairs as provided by WordNet (Miller, 1995). Since some relations are not directly outlined in these resources (e.g., the selectional preferences of nouns with regard to their adjectival modifiers), we also consider the word-embedding similarity between words. For example in Figure 2 the semantic relationships are denoted with blue arrows between pairs of content words in the story (e.g., {Sam, dogs}, {dogs, puppies}, etc).

More formally, we define the semantic coherence of s' with respect to s as:

$$\text{Sem}(s'|s) = \sum_{i}^{|s'|} \text{Sem}_{Lex}(w_i, w'_i) + \sum_{i,j \in CW} \text{Sem}_{Pair}(\{w_i, w_j\}, \{w'_i, w'_j\}) \quad (5)$$

where CW is the set of pairs of indices of content words (nouns, verbs, adjectives, and named entities) from s. We focus on the content words of the original problem, as they carry most of the semantic information. Sem_{Lex} and Sem_{Pair} functions are semantic adaptation scores for individual words and semantic relations respectively, described below.

Semantic Compatibility between words (Sem_{Lex}) is defined as:

$$\text{Sem}_{Lex}(w_i, w'_i) = cos(w_i, w'_i) + Resnik(w_i, w'_i) \quad (6)$$

where $cos(w_i, w'_i)$ denotes the cosine similarity between the vector space embeddings of two words w_i and w'_i[4], and $Resnik(w_i, w'_i)$ expresses the information content of the lowest subsumer of $\{w_i, w'_i\}$ in WordNet. For example in Figure 2, the semantic compatibility score incorporates lexical similarities $\text{Sem}_{Lex}(\text{dog}, \text{ship})$, etc.

Compatibility score between semantic relations (Sem_{Pair}) is defined by adding two components: $PairSim$ and $Analogy$ that compute how semantic relations between pairs of words are preserved in the new story:

$$PairSim = \cos(w_i, w_j) * \cos(w'_i, w'_j) \quad (7)$$

$$Analogy = \cos(w'_i + w_j - w_i, w'_j) \quad (8)$$

$PairSim$ preserves the similarity between pairs of words $\{w_i, w_j\}$ in s and the corresponding pair $\{w'_i, w'_j\}$ in the new story s'. Intuitively, if w_i and w_j are semantically close to each other, we would like the corresponding words to be close in the new story as well. For example in Figure 2, 'dog' and 'puppy' are similar in the original story, we expect the corresponding words 'ship' and 'droid' to be similar in the new story. The $Analogy$ function, inspired by Mikolov et al. (2013), computes the analogy of w'_j from w'_i given the relationship that holds between w_i and w_j in the vector space. For example in Figure 2, the relation between 'Sam' and 'dog' is similar to the relation between 'Luke Skywalker' and 'ship'.

5 Decoding

Our decoding process begins by first identifying the content words w_i (nouns, verbs, adjectives and named entities) in the original problem s that will be considered as *initial points* for rewriting. For each of these lexical classes we extract the top-k most thematic words and trivial noun compounds from the theme t. For example, in Figure 2, candidate nouns are: 'ships', 'robots', 'droids', etc., and for verbs: 'blast', 'soar', 'command', etc. Recall that the space of candidate rewrites is large, prohibiting an exhaustive enumeration. We therefore do approximate search with a beam by considering simultaneously all possible *paths* that start at the different initial points. At each step the decoder considers an additional rewrite from the list of candidates, adds it to the existing hypothesis path, and scores it according to function $\mathcal{R}$ (Equation 2).

All the counterpart scores are locally optimal, as they factor over each new word w'_i or pair of $\{w'_i, w'_j\}$, where w'_j is a rewrite *already* existing in the hypothesis path. At any given step we may recombine hypotheses that share the same prefix hypothesis path, and keep the top scoring one. The process terminates when there are no more rewrites left. We also experimented decoding with a variety of orderings of the text in the original problem s, including left-to-right, and head-first following the

[4]For the ease of notation, we represent the embedding of the words with w_i as well.

dependency tree of each sentence and then concatenating these linearizations; we observed that considering multiple paths achieves the best performance.

6 Data Collection

For the set of human-authored stories $\{s\}$, we use a corpus of math word problems described in Koncel-Kedziorski et al. (2016). We select a subset of 150 problems targeting 5th and 6th grade levels, all of which involve a single equation in one variable. These problems have 2.7 sentences and 29.4 words on average, 12.6 of which are considered content words by our system. In order to tune and evaluate our model, we collect a corpus of human-authored rewrites produced by workers from Amazon Mechanical Turk based on two themes: Star Wars, and Adventure Time (a children's cartoon).

We experimented with different ways of helping to define the theme for the workers, including offering automatically generated word clouds or enforcing that a response includes one of several keywords. In practice, we have found that using specific cultural elements as themes (such as famous movie or cartoon franchises) attracts workers who already have a strong knowledge of the theme, resulting in higher quality work.

To help explain the rewriting process, we show workers three examples of thematic rewrites with varying degrees of correlation to the original problems. We then show workers a random problem from the original set $\{s\}$ and a corresponding equation for that problem. We instruct the workers to "rewrite" the problem according to the theme, ensuring that their rewritten problem can be solved by the provided equation. The final dataset collection comprises of 450 human-authored rewrites. We collect 3 rewrites for 100 of the original problems for the Star Wars theme (based on the popular Star Wars sequel movies), and 3 rewrites for the rest of the 50 original problems, for the Children Cartoons Theme (CARTOON), based on the Adventure Time TV show. We keep 150 examples from the Star Wars theme for development (STAR$_{dev}$), and the rest 150 for testing (STAR$_{test}$).

We collected the STAR$_{dev}$ and CARTOON data based on workers with the "master" designation and at least 95% approval rating. Then we proceeded collecting STAR$_{test}$ by a subset of the authors of STAR$_{dev}$ who self-identify as theme experts and whose quality of work is manually confirmed.

7 Experiments

7.1 Setup

Implementation Details We pre-process the themes using the Stanford CoreNLP tools (Manning et al., 2014) for tokenization, Named Entity Recognition (Finkel et al., 2005), and dependency parsing (Chen and Manning, 2014). For calculating salience scores, we use the ScriptBase dataset of movie scripts (Gorinski and Lapata, 2015). The Star Wars theme is constructed from the available script, roughly 7300 words. The Cartoon theme is constructed from fan-authored scripts of the first 10 episodes of the show (Springfield, 2016) totaling 1370 words. Since our thematic options are taken from arbitrary text, we use the lists of offensive terms published by The Racial Slur database (Database, 2016) and FrontGate Media (Media, 2016) to filter out offensive content. To prohibit overgeneration, we forbid the transformation of stop words or math-specific words (Survivors, 2013; Koncel-Kedziorski et al., 2015b).

For syntactic compatibility score Syn (Equation 4) we use the English Fiction subset of the Google Syntactic N-grams corpus (Goldberg and Orwant, 2013) and train a 3-gram language model using KenLM (Heafield, 2011). For Sem_{Lex}, $PairSim$ and $Analogy$ (Equations 6-8) we use the pretrained word embeddings of Levy and Goldberg (2014). These embeddings are trained using dependency contexts rather than windows of adjacent words, allowing them to capture functional word similarity. Finally, we tune the parameters of our model (Equation 2) on the development set STAR$_{dev}$ and pick those values[5] that maximize METEOR score (Denkowski and Lavie, 2014) against 3 human references.

Evaluation We compare two ablated configurations of our method against our full model (FULL): -SYN that only uses semantic and thematicity components and does not incorporate the syntactic compatibility score, -SEM replaces the semantic coher-

[5] We set $\alpha = 0.1$, $\beta = 0.1$ and $\gamma = 1$

Model	STAR$_{dev}$	STAR$_{test}$	CARTOON
FULL	31.82	29.16	32.08
-SEM	28.72	25.55	27.55
-SYN	31.92	29.14	32.04

Table 1: METEOR results for different configuration of our model on STAR$_{dev}$, STAR$_{test}$ and CARTOON datasets.

ence score with the simpler $\cos(w_i, w_i')$, effectively rewriting only single words, and not pairs. We refrained from ablating the thematicity score as it is the core part of our model that drives the rewriting process into a new theme.

We evaluate our method using an automatic metric, and via eliciting human judgments on Amazon Mechanical Turk. For automatic evaluation, we compute the METEOR score, comparing the output of each model for a given problem and theme to the 3 human rewrites we collected, on STAR$_{dev}$, STAR$_{test}$ and CARTOON. METEOR is a recall-oriented metric, widely used in the MT community; the additional stemming, synonym and paraphrase matching modules make it more applicable for our use, given the nature of our rewriting task.[6]

For human evaluation, we conduct pairwise comparison tests, pairing FULL against a human rewrite (HUMAN), FULL against -SYN, and FULL against -SEM. Participants were given a short description of the theme, and the output of each system. For each test we asked 40 subjects to select which problem they preferred over 5 pairs of outputs; we obtained a total of 200 (5x40) responses for STAR$_{test}$ and CARTOON.

In order to better understand the strengths and weaknesses of the generated stories, we conducted a more detailed human evaluation. 8 participants were presented with the output of the three automatic systems, human rewrites (HUMAN), and a theme. The participants were asked to rate the stories across three dimensions: coherence (how coherent is the text of the problem?), solvability (can elementary school students solve it?), and thematicity (how well does the problem express them?) on a scale from 1 to 5. We collected ratings over 16 outputs from STAR$_{test}$, resulting in 128 responses.

[6]The average METEOR score comparing 1 annotator against the other 2 is 0.26, indicating that there are diverse correct strategies for solving the rewriting problem.

Model	STAR$_{test}$	CARTOON
FULL	65.0	57.9
-SYN	35.0	42.1
FULL	68.8	69.4
-SEM	31.2	30.6
FULL	17.9	10.0
HUMAN	82.1	90.0

Table 2: Human evaluation results on pairwise comparisons between FULL and -SYN, and FULL and HUMAN, on STAR$_{test}$ and CARTOON datasets.

Model	Thematicity	Coherence	Solvability
HUMAN	3.7	3.175	4.025
FULL	3.7	3.025	3.9
-SYN	3.375	3.075	3.825
-SEM	3.325	2.65	3.7

Table 3: Human evaluation results for FULL, -SYN, -SEMand HUMAN on thematicity, coherence and solvability on STAR$_{test}$.

7.2 Results

Table 1 reports METEOR; we notice that removing the semantic coherence scores in -SEM hurts the performance compared to FULL; this confirms our claim that semantic compatibility is crucial for building coherent stories. On the other hand, -SYN performs similarly to FULL. Closer inspection of the -SYN system's output reveals a greater diversity in thematic elements as a result of the relaxed syntactic compatibility constraints. Hence it is more likely to have greater overlap with any of the reference rewrites, resulting in higher METEOR scores.

However, a pairwise comparison between FULL and -SYN (Table 2) reveals that human subjects consistently prefer the output of FULL instead of -SYN both for STAR$_{test}$ and CARTOON. Table 2 also reports that HUMAN outperforms the output of the FULL model, and a pairwise comparison of FULL and -SEM which yields a result in line with the METEOR scores.

Table 3 shows the results of the detailed comparison of Thematicity, Coherence, and Solvability. This table clearly shows the strong contribution of the semantic component of our system. The specific contribution of the syntactic component is to pro-

Star Wars
s_1. Wendy bought 4 new chairs and 4 new tables for her house. If she spent 6 minutes on each piece furniture putting it together, how may minutes did it take her to finish?
s'_1. Leia bought 4 new ships and 4 new guns for her room. If she spent 6 minutes on each wasteland weapon putting it together, how many minutes did it take her to terminate?
s_2. My car gets 20 miles per gallon of gas. How many miles can I drive on 5 gallons of gas?
s'_2. My cruiser gets 20 miles per gallon of light. How many miles can I drive on 5 gallons of light?
s_3.Tyler had 15 dogs. Each dog had 5 puppies. How many puppies does Tyler now have?
s'_3. Biggs had 15 creatures. Each creature had 5 creatures. How many creatures does Biggs now have?

Table 4: Examples of the original stories s_i and rewritten math word problems s'_i in Star War theme.

Cartoon
s_7. Dave was helping the cafeteria workers pick up lunch trays, but he could only carry 9 trays at a time. If he had to pick up 17 trays from one table and 55 trays from another, how many trips will he make?
s'_7. Finn was helping the cupboard men pick up candy bottles, but he could only carry 9 bottles at a time. If he had to pick up 17 bottles from one ring and 55 bottles from another, how many swords will he make?
s_8. If books came from all the 4 continents that Bryan had been into and he collected 122 books per continent, how many books does he have from all 4 continents combined?
s'_8. If dances came from all the 4 mountains that Finn had been into and he collected 122 dances per mountain, how many dances does he have from all 4 mountains combined?
s_9. A bucket contains 3 gallons of water. If Derek adds 6.8 gallons more, how many gallons will there be in all?
s'_9. A bottle makes 3 gallons of serum. If Finn adds 6.8 gallons more, how many gallons will there be in all?

Table 5: Examples of the original stories s_i and rewritten math word problems s'_i in Cartoon theme.

duce overall more solvable and thematically satisfying problems, although it can slightly affect coherence especially when automatic parses fail. Finally, the overall high ratings for human-authored stories across all three dimensions, confirm the high quality of the crowd-sourced stories.

7.3 Qualitative Examples

Table 4–6 shows some problems generated by our method. Recall that since our system needs no annotated thematic training data, we can easily generate from any theme where thematic text is available. To demonstrate this fact, we include generated examples in a Western theme from novels from the Project Gutenberg corpus. Many of the results of our system are very legible, with only minor agreement errors. Coherent, thematic semantic relations are evident in problems such as s'_1, where ships, guns, and weapons combine to effect the Star Wars theme; this is also evident in s'_5, where people with western sounding names like Kurt and Madeline trade in cigarettes, an old-fashioned pre-cursor to e-cigarettes.

In some cases, semantic inconsistencies result in weird sounding problems, such as in s'_6 where the main character receives "wheat of grub". But because of the syntactic compatibility component, our model scores this candidate higher because of the connection between "wheat" and "graze".

Semantic incoherence is less of a problem in the cartoon theme, where absurd interactions between characters are expected. However, a difficulty for our system is demonstrated in s'_7, where the physical entity "swords" is substituted for the nominalization of an event "trips". Improvements to the semantic coherence component could resolve such issues.

Table 7 shows some instances where the rewrite algorithm produces unusable results. An example of under-generation is s'_{10}. Here, too many words are left untouched, resulting in both ungrammaticality and semantic incoherence. In s'_{11}, we witness some limitations of using word vectors. The rare word "Ferris" is not close to anything in the Star Wars theme, and is thus mapped almost arbitrarily to "int" (movie script shorthand for an interior shot). Better treatment of noun compounds and the use of phrase vectors would reduce such errors.

8 Conclusion

We formalized the problem of story rewriting as automatically changing the theme of a text without

Western
s_4. Christians father and the senior ranger gathered firewood as they walked towards the lake in the park ...
s'_4. Christian 's partner and the lone sheriff harvested barley as they strolled towards the hip in the orchard ...
s_5. Sally had 27 cards. Dan gave her 41 new cards. Sally bought 20 cards. How many cards does Sally have now?
s'_5. Madeline had 27 cigarettes. Kurt gave her 41 new cigarettes. Madeline bought 20 cigarettes. How many cigarettes does madeline have now?
s_6. For Halloween Megan received 11 pieces of candy from neighbors and 5 pieces from her older sister. If she only ate 8 pieces a day, how long would the candy last her?
s'_6. For Halloween Madeline received 11 wheat of grub from proprietors and 5 wheat from her nameless partner. If she only grazed 8 wheat a day, how long would the grub last her?

Table 6: Examples of the original stories s_i and rewritten math word problems s'_i in Western theme.

Poor Rewrites
s_{10}. It rained 0.9 inches on Monday. On Tuesday, it rained 0.7 inches less than on Monday. How much did it rain on Tuesday?
s'_{10}. It blasted 0.9 inches on Monday. On Tuesday, it blasted 0.7 inches less than on Monday. How much did it light on Tuesday?
s_{11}. The Ferris wheel in Paradise Park has 14 seats. Each seat can hold 6 people. How many people can ride the Ferris wheel at the same time?
s'_{11}. The int grab in chewbacca mesa has 14 areas. Each area can hold 6 troops. How many troops can ride the int grab at the same time?

Table 7: Examples of the original stories s_i and poorer rewrites s'_i in the Star Wars theme.

altering the underlying story and developed an approach for rewriting algebra word problems where the rewriting model optimized for a number of measures of overall text coherence. Experiments on a newly gathered dataset demonstrated our model can produce themed texts that are usually solvable.

Future work could improve the thematicity and solvability components by incorporating domain-specific and commonsense knowledge, leveraging information extraction. Additionally, neural network architectures (e.g., LSTMs, seq2seq) can be trained to rewrite coherently with less reliance on brittle syntactic parses. Additionally, we plan to study rewriting in other domains such as children's short stories and extend the model to generate math word problems directly from equations. Finally, we intend to incorporate the generated problems in educational technology and tutoring systems.

Acknowledgments

This research was supported by the NSF (IIS 1616112), Allen Institute for AI (66-9175), Allen Distinguished Investigator Award, DARPA (FA8750-13-2-0008) and a Google research faculty award. We thank the anonymous reviewers for their helpful comments.

References

Leandro Motta Barros and Soraia Raupp Musse. 2007. Planning algorithms for interactive storytelling. *Computers in Entertainment (CIE)*, 5(1):4.

Lora Bartlett. 2004. Expanding teacher work roles: a resource for retention or a recipe for overwork? *Journal of Education Policy*, 19(5):565–582.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 602–610. Association for Computational Linguistics.

David Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

The Racial Slur Database. 2016. The racial slur database.

Judy Davis-Dorsey, Steven M Ross, and Gary R Morrison. 1991. The role of rewording and context personalization in the solving of mathematical word problems. *Journal of Educational Psychology*, 83(1):61.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for

Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

K. Filippova and M. Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG)).*

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Ganitkevitch, C. Callison-Burch, C. Napoles, and B. Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Yoav Goldberg and Jon Orwant. 2013. A Dataset of syntactic-Ngrams over Time from a Very Large Corpus of English Books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona.

Janis M Hart. 1996. The Effect of Personalized Word Problems. *Teaching Children Mathematics*, 2(8):504–505.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *EMNLP*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word probl ems? large-scale dataset construction and evaluation. In *Proceedings of the 2016 North American Chapter of the ACL (NAACL HLT)*.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning Sentences from Standard Wikipedia to Simple Wikipedia. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015a. Parsing Algebraic Word Problems into Equations. *TACL*, 3.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015b. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Aimini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A Math Word Problem Repository. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to Automatically Solve Algebra Word Problems. In *ACL*, pages 271–281.

Benoit Lavoie and Owen Rambow. 1997. A Fast and Portable Realizer for Text Generation Systems. In *Proceedings of the fifth conference on Applied natural language processing*, pages 265–268. Association for Computational Linguistics.

Michael Lebowitz. 1987. Planning Stories. In *Proceedings of the cognitive science society, Hillsdale*, pages 234–242.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *ACL*, pages 302–308.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Hugo Liu and Push Singh. 2002. MAKEBELIEVE: Using Commonsense Knowledge to Generate Stories. In *AAAI/IAAI*, pages 957–958.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Conference of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 55–60.

Neil McIntyre and Mirella Lapata. 2009. Learning to Tell Tales: A Data-driven Approach to Story Generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2010. Plot Induction and Evolutionary Search for Story Generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572. Association for Computational Linguistics.

FrontGate Media. 2016. Terms to block.

James Richard Meehan. 1976. The Metanovel: Writing Stories by Computer. Technical report, DTIC Document.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

George A Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Arindam Mitra and Chitta Baral. 2015. Learning to automatically solve logic grid puzzles. In *EMNLP*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *Proceedings of the 2016 North American Chapter of the ACL (NAACL HLT)*.

Sarah Petersen and Mari Ostendorf. 2007. Text simplification for langauge learners: A corpus analysis. In *Proceedings of the Speech and Language Technology in Education Workshop (SLaTE)*.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona.

Oleksandr Polozov, Eleanor ORourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popovic. 2015. Personalized Mathematical Word Problem Generation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). To appear*.

KA Renninger, L Ewen, and AK Lasher. 2002. Individual interest as context in expository text and mathematical word problems. *Learning and Instruction*, 12(4):467–490.

Subhro Roy and Dan Roth. 2015. Solving General Arithmetic Word Problems. In *EMNLP*.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing : Mapping sentences to grounded equations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Roger C Schank and Robert P Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram Understanding in Geometry Questions. In *AAAI*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving Geometry Problems: Combining Text and Diagram Interpretation. In *EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. In *EMNLP*.

Advaith Siddharthan. 2004. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Springfield. 2016. Adventure time with finn & jake episode scripts.

Ladder Survivors. 2013. Key words for math problems.

Scott R Turner. 1993. Minstrel: A Computer Model of Creativity and Storytelling.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 344–352.

Kristian Woodsend and Mirella Lapata. 2011a. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kristian Woodsend and Mirella Lapata. 2011b. Wikisimple: Automatic simplification of wikipedia articles. In *Proceedings of the Association for Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, pages 927–932, San Francisco, CA.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual

machine translation. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 1015–1024.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of Association of Computational Linguistics*.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to Solve Algebra Word Problems Using Quadratic Programming. In *EMNLP*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Context-Sensitive Lexicon Features for Neural Sentiment Analysis

Zhiyang Teng, Duy-Tin Vo and **Yue Zhang**
Singapore University of Technology and Design
{zhiyang_teng, duytin_vo}@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Abstract

Sentiment lexicons have been leveraged as a useful source of features for sentiment analysis models, leading to the state-of-the-art accuracies. On the other hand, most existing methods use sentiment lexicons without considering context, typically taking the count, sum of strength, or maximum sentiment scores over the whole input. We propose a context-sensitive lexicon-based method based on a simple weighted-sum model, using a recurrent neural network to learn the sentiments strength, intensification and negation of lexicon sentiments in composing the sentiment value of sentences. Results show that our model can not only learn such operation details, but also give significant improvements over state-of-the-art recurrent neural network baselines without lexical features, achieving the best results on a Twitter benchmark.

1 Introduction

Sentiment lexicons (Hu and Liu, 2004; Wilson et al., 2005; Esuli and Sebastiani, 2006) have been a useful resource for opinion mining (Kim and Hovy, 2004; Agarwal et al., 2011; Moilanen and Pulman, 2007; Choi and Cardie, 2008; Mohammad et al., 2013; Guerini et al., 2013; Vo and Zhang, 2015). Containing sentiment attributes of words such as polarities and strengths, they can serve to provide a word-level foundation for analyzing the sentiment of sentences and documents. We investigate an effective way to use sentiment lexicon features.

A traditional way of deciding the sentiment of a document is to use the sum of sentiment values of all words in the document that exist in a sentiment lexicon (Turney, 2002; Hu and Liu, 2004). This simple method has been shown to give surprisingly competitive accuracies in several sentiment analysis benchmarks (Kiritchenko et al., 2014), and is still the standard practice for specific research communities with mature domain-specific lexicons, such as finance (Kearney and Liu, 2014) and product reviews (Ding et al., 2008).

It's an **insignificant** [criticism]$_{-1\rightarrow-0.5}$.
Nobody gives a [good]$_{+3\rightarrow-1}$ performance in this movie
She's **not** [terrific]$_{+5\rightarrow+1}$ but **not** [terrible]$_{-5\rightarrow-1}$ either.
It's **not** a **very** [good]$_{+3\rightarrow-0.25}$ movie song!
It **removes** my [doubts]$_{-3\rightarrow+1}$.

Figure 1: Example sentiment compositions.

More sophisticated sentence-level features such as the counts of positive and negative words, their total strength, and the maximum strength, etc, have also been exploited (Kim and Hovy, 2004; Wilson et al., 2005; Agarwal et al., 2011). Such lexicon features have been shown highly effective, leading to the best accuracies in the SemEval shared task (Mohammad et al., 2013). On the other hand, they are typically based on bag-of-word models, hence suffering two limitations. First, they do not explicitly handle *semantic compositionality* (Polanyi and Zaenen, 2006; Moilanen and Pulman, 2007; Taboada et al., 2011), some examples of which are shown in Figure 1. The composition effects can exhibit intricacies such as negation over intensification (e.g. not very good), shifting (e.g. not terrific) vs flip-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1629–1638,
Austin, Texas, November 1-5, 2016.

ping negation (e.g. not acceptable), content word negation (e.g. removes my doubts) and unbounded dependencies (e.g. No body gives a good performance).

Second, they cannot effectively deal with *word sense variations* (Devitt and Ahmad, 2007; Denecke, 2009). Guerini et al. (2013) show challenges in modeling the correlation between context-dependent posterior word sentiments and their context independent priors. For example, the sentiment value of "cold" varies between "cold beer", "cold pizza" and "cold person" due to sense and context differences. Such variations raise difficulties for a sentiment classifier with bag-of-word nature, since they can depend on semantic information over long phrases or the full sentence.

We investigate a method that can potentially address the above issues, by using a recurrent neural network to capture context-dependent semantic composition effects over sentences. Shown in Figure 2, the model is conceptually simple, using a weighted sum of lexicon sentiments and a sentence-level bias to estimate the sentiment value of a sentence. The key idea is to use a bi-directional long-short-term-memory (LSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2013) model to capture global syntactic dependencies and semantic information, based on which the weight of each sentiment word together with a sentence-level sentiment bias score are predicted. Such weights are context-sensitive, and can express flipping negation by having negative values.

The advantages of the recurrent network model over existing semantic-composition-aware discrete models such as (Choi and Cardie, 2008) include its capability of representing non-local and subtle semantic features without suffering from the challenge of designing sparse manual features. On the other hand, compared with neural network models, which recently give the state-of-the-art accuracies (Li et al., 2015; Tai et al., 2015), our model has the advantage of leveraging sentiment lexicons as a useful resource. To our knowledge, we are the first to integrate the operation into sentiment lexicons and a deep neural model for sentiment analysis.

The conceptually simple model gives strong empirical performances. Results on standard sentiment benchmarks show that our method gives competitive

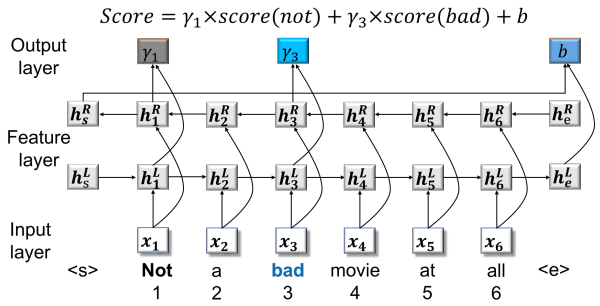


Figure 2: Overall model structure. The sentiment score of the sentence "not a bad movie at all" is a weighted sum of the scores of sentiment words "not", "bad" and a sentence-level bias score b. $score(not)$ and $score(bad)$ are prior scores obtained from sentiment lexicons. γ_1 and γ_3 are context-sensitive weights for sentiment words "not" and "bad", respectively.

accuracies to the state-of-the-art models in the literature. As a by-product, the model can also correctly identify the compositional changes on the sentiment values of each word given a sentential context.

Our code is released at https://github.com/zeeeyang/lexicon_rnn.

2 Related Work

There exist many statistical methods that exploit sentiment lexicons (Kim and Hovy, 2004; Agarwal et al., 2011; Mohammad et al., 2013; Guerini et al., 2013; Tang et al., 2014b; Vo and Zhang, 2015; Cambria, 2016). Mohammad et al. (2013) leverage a large sentiment lexicon in a SVM model, achieving the best results in the SemEval 2013 benchmark on sentence-level sentiment analysis (Nakov et al., 2013). Compared to these methods, our model has two main advantages. First, we use a recurrent neural network to model context, thereby exploiting *non-local* semantic information. Second, our model offers *context-sensitive* operational details on each word.

Several previous methods move beyond bag-of-word models in leveraging lexicons. Most notably, Moilanen and Pulman (2007) introduce the ideas from compositional semantics (Montague, 1974) into sentiment operations, developing a set of composition rules for handling negations. Along the line, Taboada et al. (2011) developed a lexicon and a collection of sophisticated rules for addressing intensification, negation and other phenomena. Differ-

ent from these *rule-based* methods, Choi and Cardie (2008) use a structured linear model to *learn* semantic compositionality relying on a set of *manual* features. In contrast, we leverage a recurrent neural model for inducing semantic composition features *automatically*. Our weighted-sum representation of semantic compositionality is formally simpler compared with fine-grained rules such as (Taboada et al., 2011). However, it is sufficient for describing the *resulting effect* of complex and context-dependent operations, with the semantic composition process being modeled by LSTM. Our sentiment analyzer also enjoys a more competitive LSTM baseline compared to a traditional discrete models.

Our work is also related to recent work on using deep neural networks for sentence-level sentiment analysis, which exploits convolutional (Kalchbrenner et al., 2014; Kim, 2014; Ren et al., 2016), recursive (Socher et al., 2013; Dong et al., 2014; Nguyen and Shirai, 2015) and recurrent neural networks (Liu et al., 2015; Wang et al., 2015; Zhang et al., 2016), giving highly competitive accuracies. As our baseline, LSTM (Tai et al., 2015; Li et al., 2015) stands among the best neural methods. Our model is different from these prior methods in mainly two aspects. First, we introduce sentiment lexicon features, which effectively improve classification accuracies. Second, we learn extra operation details, namely the weights on each word, automatically as hidden variables. While the baseline uses LSTM features to perform end-to-end mapping between sentences and *sentiments*, our model uses them to induce the *lexicon weights*, via which word level sentiment are composed to derive sentence level sentiment.

3 Model

Formally, given a sentence $s = w_1 w_2 ... w_n$ and a sentiment lexicon *D*, denote the subjective words in s as $w^D_{j_1} w^D_{j_2} ... w^D_{j_m}$. Our model calculates the sentiment score of s according to *D* in the form of

$$Score(s) = \sum_{t=1}^{m} \gamma_{j_t} score(w^D_{j_t}) + b, \qquad (1)$$

where $Score(w^D_{j_t})$ is the sentiment value of w_{j_t}, γ_{j_t} are sentiment weights and b is a sentence-level bias. The sentiment values of words and sentences are real numbers, with the sign indicating the polarity and the absolute value indicating the strength.

As shown in Figure 2, our neural model consists of three main layers, namely the *input layer*, the *feature layer* and the *output layer*. The input layer maps each word in the input sentence into a dense real-value vector. The feature layer exploits a bidirectional LSTM (Graves and Schmidhuber, 2005; Graves et al., 2013) to extract non-local semantic information over the sequence. The output layer calculates a weight score for each sentiment word, as well as an overall sentiment bias of the sentence.

In this figure, the score of the sentence "not a bad movie at all" is decided by a weighted sum of the sentiments of "bad" and "not"[1], and a sentiment shift bias based on the sentence structure. Ideally, the weight on "not" should be a small negative value, which results in a slightly positive sentiment shift. The weight on "bad" should be negative, which represents a flip in the polarity. These weights jointly model a negation effect that involves both shifting and flipping.

3.1 Bidirectional LSTM

We use LSTM (Hochreiter and Schmidhuber, 1997) for feature extraction, which recurrently processes sentence s token by token. For each word w_t, the model calculate a hidden state vector $\mathbf{h_t}$. A LSTM cell block makes use of an input gate $\mathbf{i_t}$, a memory cell $\mathbf{c_t}$, a forget gate $\mathbf{f_t}$ and an output gate $\mathbf{o_t}$ to control information flow from the history $\mathbf{x_1}...\mathbf{x_t}$ and $\mathbf{h_1}...\mathbf{h_{t-1}}$ to the current state $\mathbf{h_t}$. Formally, $\mathbf{h_t}$ is computed as follows:

$$\begin{aligned}
\mathbf{i_t} &= \sigma(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{V_i c_{t-1}} + \mathbf{b_i}) \\
\mathbf{f_t} &= \mathbf{1.0} - \mathbf{i_t} \\
\mathbf{g_t} &= \tanh(\mathbf{W_g x_t} + \mathbf{U_g h_{t-1}} + \mathbf{b_g}) \\
\mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \mathbf{g_t} \\
\mathbf{o_t} &= \sigma(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{V_o c_t} + \mathbf{b_o}) \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned}$$

Here $\mathbf{x_t}$ is the word embedding of word w_t, σ denotes the sigmoid function, $\odot$ is element-wise multiplication. $\mathbf{W_i}$, $\mathbf{U_i}$, $\mathbf{V_i}$, $\mathbf{b_i}$, $\mathbf{W_g}$, $\mathbf{U_g}$, $\mathbf{b_g}$, $\mathbf{W_o}$, $\mathbf{U_o}$, $\mathbf{V_o}$ and $\mathbf{b_o}$ are LSTM parameters.

[1] Most sentiment lexicons assign a negative score to the word "not".

We apply a bidirectional extension of LSTM (BiLSTM) (Graves and Schmidhuber, 2005; Graves et al., 2013), shown in Figure 2, to encode the input sentence s both left-to-right and right-to-left. The BiLSTM model maps each word w_t to a pair of hidden vectors $\mathbf{h_t^L}$ and $\mathbf{h_t^R}$, which denote the hidden vector of the left-to-right LSTM and right-to-left LSTM, respectively. We use different parameters for the left-to-right LSTM and the right-to-left LSTM. These state vectors are used as features for calculating the sentiment weights γ.

In addition, we append a sentence end marker $w_{<e>}$ to the left-to-right LSTM and a sentence start marker $w_{<s>}$ to the right-to-left LSTM. The hidden state vector of $w_{<s>}$ and $w_{<e>}$ are denoted as $\mathbf{h_s^R}$ and $\mathbf{h_e^L}$, respectively.

3.2 Output Layer

The base score. Given a lexicon word w_{j_t} in the sentence s ($w_{j_t} \in D$), we use the hidden state vectors $\mathbf{h_{j_t}^L}$ and $\mathbf{h_{j_t}^R}$ in the feature layer to calculate a weight value τ_{j_t}. As shown in Figure 3, a two-layer neural network is used to induce τ_{j_t}. In particular, a hidden layer combines $\mathbf{h_t^L}$ and $\mathbf{h_t^R}$ using a nonlinear $tanh$ activation

$$\mathbf{p_{j_t}^s} = \tanh(\mathbf{W_{ps}^L h_{j_t}^L} + \mathbf{W_{ps}^R h_{j_t}^R} + \mathbf{b_{ps}}) \quad (2)$$

The resulting hidden vector $\mathbf{p_{j_t}^s}$ is then mapped into τ_{j_t} using another $tanh$ layer.

$$\tau_{j_t}^s = 2\tanh(\mathbf{W_{pw} p_{j_t}^s} + \mathbf{b_{pw}}) \quad (3)$$

We choose the $2tanh$ function to make the learned weights conceptually useful. The factor 2 is introduced for modelling the effect of intensification. Since the range of $tanh$ function is $[-1, 1]$, the range of $2tanh$ is $[-2, 2]$. Intuitively, a weight value of 1 maps the word sentiment directly to the sentence sentiment, such as the weight for "good" in "This is good". A weight value in $(1, 2]$ represents intensification, such as the weight for "bad" in "very bad". Similarly, a weight value in $(0, 1)$ represents weakening, and a weight in $(-2, 0)$ represents various scales of negations.

Given all lexicon words $w_{j_t}^D$ in the sentence, we calculate a base score for the sentence

$$S_{base} = \frac{\sum_{t=1}^{m} \tau_{j_t} score(w_{j_t}^D)}{m} \quad (4)$$

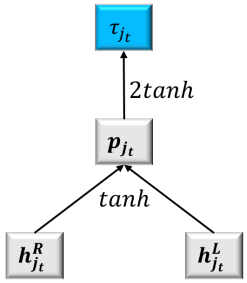


Figure 3: Weight score calculation.

By averaging the score of each word, the resulting S_{base} is confined to $[-2\alpha, 2\alpha]$, where α is the maximum absolute value of word sentiment. In the above equations, $\mathbf{W_{ps}^L}$, $\mathbf{W_{ps}^R}$, $\mathbf{b_{ps}}$, $\mathbf{W_{pw}}$ and $\mathbf{b_{pw}}$ are model parameters.

The bias score. We use the same neural network structure in Figure 3 to calculate the overall bias of the input sentence. The input to the neural network includes $\mathbf{h_s^R}$ and $\mathbf{h_e^L}$, and the output is a bias score S_{bias}. Intuitively, the calculation of S_{bias} relies on information of the full sentence. $\mathbf{h_s^R}$ and $\mathbf{h_e^L}$ are chosen because they have commonly been used in the research literature to represent overall sentential information (Graves et al., 2013; Cho et al., 2014).

We use a dedicated set of parameters for calculating the bias, where

$$\mathbf{p^B} = \tanh(\mathbf{W_{pb}^L h_e^L} + \mathbf{W_{pb}^R h_s^R} + \mathbf{b_{pb}}) \quad (5)$$

and

$$S_{bias} = 2\tanh(\mathbf{W_b p^B} + \mathbf{b_p}) \quad (6)$$

$\mathbf{W_{pb}^L}$, $\mathbf{W_{pb}^R}$, $\mathbf{b_{pb}}$, $\mathbf{W_b}$ and $\mathbf{b_p^L}$ are parameters.

3.3 Final Score Calculation

The base S_{base} and bias S_{bias} are linearly interpolated to derive the final sentiment value for the sentence s.

$$Score(s) = \lambda S_{base} + (1 - \lambda) S_{bias} \quad (7)$$

$\lambda \in [0, 1]$ reflects the relative importance of the base score in the sentence. It offers a new degree of model flexibility, and should be calculated for each sentence specifically. We use the attention model (Bahdanau et al., 2014) to this end. In particular, the base score features $\mathbf{h_t^L}$/$\mathbf{h_t^R}$ and the bias score features $\mathbf{h_e^L}$/$\mathbf{h_s^R}$ are combined in the calculation

$$\lambda = \sigma(\mathbf{W_{s\lambda} h_{base}} + \mathbf{W_{b\lambda} h_{bias}} + \mathbf{b}_\lambda) \quad (8)$$

where

$$\mathbf{h_{bias}} = \mathbf{h_e^L} \oplus \mathbf{h_s^R} \quad (9)$$

and

$$\mathbf{h_{base}} = \frac{\sum_{t=1}^{m} \mathbf{h_{jt}^L} \oplus \mathbf{h_{jt}^R}}{m} \quad (10)$$

Here σ denotes the sigmoid activation function and $\oplus$ denotes vector concatenation. $\mathbf{W_{s\lambda}}$, $\mathbf{W_{b\lambda}}$ and $\mathbf{b}_\lambda$ are model parameters.

The final score of the sentence is

$$\begin{aligned} Score(s) &= \lambda S_{base} + (1-\lambda) S_{bias} \\ &= \frac{\lambda}{m} \sum_{t=1}^{m} \tau_{jt} score(w_{jt}^D) + (1-\lambda) S_{bias} \end{aligned}$$

This corresponds to the original Equation 1 by $\gamma_{j_t} = \frac{\lambda}{m}\tau_{j_t}$ and $b = (1-\lambda)S_{bias}$.

3.4 Training and Testing

Our training data contains two different settings. The first is binary sentiment classification. In this task, every sentence s_i is annotated with a sentiment label l_i, where $l_i = 0$ and $l_i = 1$ to indicate negative and positive sentiment, respectively. We apply logistic regression on the output layer. Denote the probability of a sentence s_i being positive and negative as $p_{s_i}^1$ and $p_{s_i}^0$ respectively. $p_{s_i}^0$ and $p_{s_i}^1$ are estimated as

$$\begin{aligned} p_{s_i}^1 &= \sigma(Score(s_i)) \\ p_{s_i}^0 &= 1 - p_{s_i}^1 \end{aligned} \quad (11)$$

Suppose that there are N training sentences, the loss function over the training set is defined as

$$L(\mathbf{\Theta}) = -\sum_{i=1}^{N} \log p_{s_i}^{l_i} + \frac{\lambda_r}{2} ||\mathbf{\Theta}||^2, \quad (12)$$

where Θ is the set of model parameters. λ_r is a parameter for L2 regularization.

The second setting is multi-class classification. In this task, every sentence s_i is assigned a sentiment label l_i from 0 to 4, which represent *very negative*, *negative*, *neutral*, *positive* and *very positive*, respectively. We apply least square regression on the output layer. Since the output range of $2tanh$ is [-2, 2], the value of the base score and the bias score both belongs to [-2, 2]. The final score is a weighted sum of the base score and the bias score, also belonging to [-2, 2]. However, the gold sentiment label ranges from 0 to 4. We add an offset -2 to every gold sentiment label to both adapt our model to the training data and to increase the interpretability of the learned weights. The loss function for this problem is then defined as

$$L(\mathbf{\Theta}) = \sum_{i=1}^{N} (Score(s_i) - l_i)^2 + \frac{\lambda_r}{2} ||\mathbf{\Theta}||^2 \quad (13)$$

During testing, we predict the sentiment label l_i^* of a sentence s_i by

$$l_i^* = \begin{cases} -2 \text{ if } Score(s_i) \leq -1.5 \\ -1 \text{ if } -1.5 < Score(s_i) \leq -0.5 \\ 0 \text{ if } -0.5 < Score(s_i) \leq 0.5 \\ 1 \text{ if } 0.5 < Score(s_i) \leq 1.5 \\ 2 \text{ if } Score(s_i) > 1.5 \end{cases} \quad (14)$$

	Positive	**Negative**	**Total**
Train	3,009	1,187	4,196
Dev	483	283	766
Test	1,313	490	1,803

Table 1: Statistics of the Twitter dataset.

Task	Label	Training Sentences	Dev Sentences	Test Sentences
5-class	-2	1,092	139	279
	-1	2,218	289	633
	0	1,624	229	389
	1	2,322	279	510
	2	1,288	165	399
2-class	0	3,310	444	909
	1	3,610	428	912

Table 2: Statistics of SST.

4 Experiments

4.1 Experimental Settings

Data. We test our model on three datasets, including a dataset on Twitter sentiment classification, a dataset on movie review and a dataset with mixed domains. The Twitter dataset is taken from SemEval 2013 (Nakov et al., 2013). We downloaded the dataset according to the released *ids*. The statistics of the dataset are shown in Table 1.

The movie review dataset is Stanford Sentiment Treebank[2] (SST) (Socher et al., 2013). For each sentence in this treebank, a corresponding constituent

[2]http://nlp.stanford.edu/sentiment/index.html

Polarity	books	dvds	electronics	music	videogames
Positive	19	19	19	20	20
Negative	29	20	19	20	20

Table 3: Document distribution of the mixed domain dataset.

tree is given. Each internal constituent node is annotated with a sentiment label ranging from 0 to 4. We follow Socher et al. (2011) and Li et al. (2015) to perform five-class and binary classification, with the data statistics being shown in Table 2.

In order to examine cross-domain robustness, we apply our model on a product review corpus (Täckström and McDonald, 2011), which contains 196 documents covering 5 domains: books, dvds, electronics, music and videogames. The document distribution is listed in Table 3.

Lexicons. We use four sentiment lexicons, namely *TS-Lex*, *S140-Lex*, *SD-Lex* and *SWN-Lex*. **TS-Lex**[3] is a large-scale sentiment lexicon built from Twitter by Tang et al. (2014a) for learning sentiment-specific phrase embeddings. **S140-Lex**[4] is the *Sentiment140* lexicon, which is built from point-wise mutual information using distant supervision (Go et al., 2009; Mohammad et al., 2013).

SD-Lex is built from SST. We construct a sentiment lexicon from the training set by excluding all neutral words and adding the aforementioned offset -2 to each entry. **SWN-Lex** is a sentiment lexicon extracted from SentimentWordNet3.0 (Baccianella et al., 2010). For words with different part-of-speech tags, we keep the minimum negative score or the maximum positive score. The original score in the SentimentWordNet3.0 is a probability value between 0 and 1, and we scale it to [-2, 2][5].

When building these lexicons, we only use the sentiment scores for unigrams. Ambiguous words are discarded. Both TS-Lex and S140-Lex are Twitter-specific sentiment lexicons. They are used in the Twitter sentiment classification task. SD-Lex and SWN-Lex are exploited for the Stanford dataset. The statistics of lexicons are listed in Table 4.

[3]http://ir.hit.edu.cn/ dytang/paper/14coling/data.zip

[4]http://saifmohammad.com/Lexicons/Sentiment140-Lexicon-v0.1.zip

[5]Taboada et al. (2011) also mentioned two methods to derive sentiment score for a sentiment word from SentimentWordNet. We leave them for future work.

Lexicon	Positive	Negative	Total
SD-Lex	2,547	2,448	4,995
SWN-Lex	15,568	17,412	32,980
TS-Lex	33,997	32,026	66,023
S140-Lex	24,156	38,312	62,468

Table 4: Statistics of sentiment lexicons.

4.2 Implementation Details

We implement our model based on the CNN toolkit.[6] Parameters are optimized using stochastic gradient descent with momentum (Sutskever et al., 2013). The decay rate is 0.1. For initial learning rate, L2 and other hyper-parameters, we adopt the default values provided by the CNN toolkits. We select the best model parameter according to the classification accuracy on the development set.

For the Twitter data, we use the *glove.twitter.27B*[7] as pretrained word embeddings. For the Stanford dataset, following Li et al. (2015), we use *glove.840B.300d*[8] as pretrained word embeddings. Words that do not exist in both the training set and the pretrained lookup table are treated as out-of-vocabulary (OOV) words. Following Dyer et al. (2015), singletons in the training data are randomly mapped to UNK with a probability p_{unk} during training. We set $p_{unk} = 0.1$. All word embeddings are fine-tuned. We use dropout (Srivastava et al., 2014) in the input layer to prevent overfitting during training.

One-layered BiLSTM is used for all tasks. The dimension of the hidden vector in LSTM is 150. The size of the second layer in Figure 3 is 64.

4.3 Development Results

Table 5 shows results on the Twitter development set. **Bi-LSTM** is our model using the bias score S_{bias} only, which is equivalent to bidirectional LSTM model of Li et al. (2015) and Tai et al. (2015), since they use same features and only differ in the output layer. **Bi-LSTM+avg.lexicon** is a baseline model integrating the average sentiment scores of lexicon words as a feature, and **Bi-LSTM+flex.lexicon** is our final model, which considers both the Bi-LSTM score (S_{bias}) and the context-sensitive score (S_{base}).

[6]https://github.com/clab/cnn

[7]http://nlp.stanford.edu/data/glove.twitter.27B.zip

[8]http://nlp.stanford.edu/data/glove.840B.300d.zip

Method	Dict	Dev(%)
Bi-LSTM	None	84.2
Bi-LSTM+avg.lexicon	S140-Lex	84.9
Bi-LSTM+flex.lexicon	S140-Lex	**86.4**

Table 5: Results on the Twitter development set.

Method	Test(%)
SVM6 (Zhu et al., 2014)	78.5
Tang et al. (2014a)	82.4
Bi-LSTM	86.7
Bi-LSTM + TS-Lex	87.6
Bi-LSTM + S140-Lex	**88.0**

Table 6: Results on the Twitter test set.

Bi-LSTM+avg.lexicon improves the classification accuracy over **Bi-LSTM** by 0.7 point, which shows the usefulness of sentiment lexicons to *recurrent neural models* using a vanilla method. It is consistent with previous research on *discrete* models. By considering *context-sensitive* weighting for sentiment words **Bi-LSTM+flex.lexicon** further outperforms **Bi-LSTM+avg.lexicon**, improving the accuracy by 1.5 points ($84.9 \rightarrow 86.4$), which demonstrates the strength of context-sensitive scoring. Base on the development results, we use **Bi-LSTM+flex.lexicon** for the remaining experiments.

4.4 Main Results

Twitter. Table 6 shows results on the Twitter test set. **SVM6** is our implementation of Zhu et al. (2014), which extracts six types of manual features from TS-Lex for SVM classification. The features include: (1) the number of sentiment words in the sentence; (2) the total sentiment scores of the sentence; (3) the maximum sentiment score; (4) the total positive and negative sentiment scores; (5) the sentiment score of the last word in the sentence. The system of Tang et al. (2014a) is a state-of-the-art system that extracts various manually designed features from TS-Lex, such as bag-of-words, term frequency, parts-of-speech, the sum of sentiment scores of all words in a tweet, etc, for SVM. The **Bi-LSTM** rows are our final models with different lexicons.

Both **SVM6** and Tang et al. (2014a) exploit discrete features. Compared to them, **Bi-LSTM** gives better accuracies without using lexicons, which demonstrates the relative strength of deep neural network for sentiment analysis. Compared with Tang et al. (2014a), our **Bi-LSTM+TS-Lex** model improves the sentiment classification accuracy from 82.4 to 87.6, which again shows the strength of context-sensitive features. S140-Lex gives slight improvements over TS-Lex.

Method	5-class	2-class
RAE (Socher et al., 2011)	43.2	82.4
MV-RNN (Socher et al., 2012)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DRNN (Irsoy and Cardie, 2014)	49.8	88.6
Dependency TreeLSTM (Tai et al., 2015)	48.4	85.7
Constituency TreeLSTM (Tai et al., 2015)	51.0	88.0
Constituency TreeLSTM (Li et al., 2015)	50.4	86.7
S-LSTM (Zhu et al., 2015)	50.1	-
LSTM-RNN (Le and Zuidema, 2015)	49.9	88.0
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	88.1
DCNN (Kalchbrenner et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
NBoW (Kalchbrenner et al., 2014)	42.4	80.5
SVM (Socher et al., 2013)	40.7	79.4
BiLSTM (Tai et al., 2015)	49.1	87.5
BiLSTM (Li et al., 2015)	49.8	86.7
Hier-Sequence (Li et al., 2015)	50.7	86.9
Bi-LSTM+SD-Lex	50.0	88.1
Bi-LSTM+SWN-Lex	**51.1**	**89.2**

Table 7: Results on SST. **5-class** shows fine-grained classification. The last block lists our results.

SST. Table 7 shows the results on SST. We include various results of recursive (the first block), convolutional (the second block), and sequential LSTM models (the fourth block). These neural models give the recent state-of-the-art on this dataset. Our method achieves highly competitive accuracies. In particular, compared to sequential LSTMs, our best model gives the top result both on the binary and fine-grained classification task. This shows the usefulness of lexicons to neural models. In addition, SWN-Lex gives better results compared with SD-Lex. This is intuitive because SD-Lex is a smaller lexicon compared to SWN-Lex (4,999 entries v.s. 32,980 entries). SD-Lex does not bring external knowledge to this dataset, while SWN-Lex does.

Cross-domain Results. Lexicon-based methods can be robust for cross-domain sentiment analysis (Taboada et al., 2011). We test the robustness of our model in the mixed domain dataset of product reviews (Täckström and McDonald, 2011). This dataset contains document level sentiments. We take the majority voting strategy to transform sentiment

Model	Train	Test	Books	Dvds	Electronics	Music	Videogames	Average
Bi-LSTM	None	None	71.79	89.74	65.79	95	**85**	81.63
Bi-LSTM+flex.lexicon	SD-Lex	SD-Lex	76.92	84.62	**78.95**	92.5	80	82.65
Bi-LSTM+flex.lexicon	SD-Lex	SWN-Lex	82.05	**92.31**	73.68	92.5	80	84.18
Bi-LSTM+flex.lexicon	SWN-Lex	SWN-Lex	**84.62**	**92.31**	68.42	**100**	**85**	**86.22**

Table 8: Cross-domain sentiment analysis. Training domain is movie review.

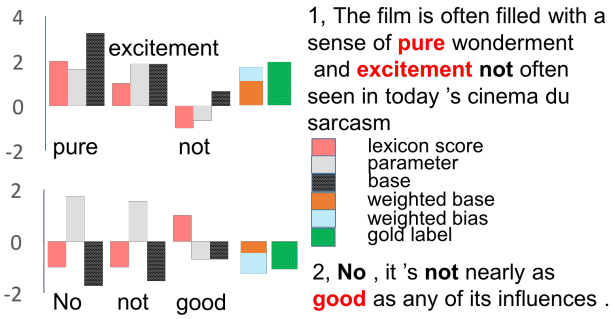


Figure 4: Sentiment composition examples.

ID	Sentence	Bi-LSTM	SWN-Lex
1	The issue of faith is not explored very deeply	0	-1
2	Steers turns in a snappy screenplay that curls at the edges; it 's so clever you want to hate it.	2	1
3	A film so tedious that it is impossible to care whether that boast is true or not.	-2	-1

Table 9: Example predictions made by the Bi-LSTM model and our Bi-LSTM+SWN-Lex model for fine-grained classification task. Red words and blue words are positive and negative entries in the SentimentWordNet3.0 lexicon, respectively.

of sentences to the document level. We compare the effects of different lexicons over a baseline Bi-LSTM trained on SST (movie domain).

Table 8 shows the results. Introducing the sentiment lexicons SD-Lex and SWN-Lex consistently improves the classification accuracy across five domains compared with the baseline **Bi-LSTM** model. When trained and tested using the same lexicon, SWN-Lex gives better performances on three out of five domains. SD-Lex gives better results only on Electronics. This shows that the results are sensitive to the domain of the sentiment lexicon, which is intuitive.

We also investigate a model trained using SD-Lex but tested by replacing SD-Lex with SWN-Lex. This is to examine the generalizability of a source-domain model on different target domains by plugging in relevant domain-specific lexicons, without being retrained. Results show that the mode still outperforms the SD-Lex lexicon on two out of five domains, but is less accurate than full retraining using SWN-Lex.

4.5 Discussion

Figure 4 shows the details of sentiment composition for two sentences in the SST, learned automatically by our model. For the first sentence, the three subjective words in the lexicon "pure", "excitement" and "not" receives weights of 1.6, 1.9 and -0.6, respectively, and the overall bias of the sentence is positive. A λ value (0.58) that slightly biases towards the base score leads to a final sentiment score is 1.8, which is close to the gold label 2.

In the second example, both negation words received positive weight values, and the bias over the sentence is negative. A λ (0.3) value that biases towards the bias score results in a final score of -1.2, which is close to the gold label -1. These results demonstrate the capacity of the model to decide how word-level sentiments composite according to sentence-level context.

Table 9 shows three sentences in the Stanford test set which are incorrectly classified by Bi-LSTM model, but correctly labeled by our Bi-LSTM+SWN-Lex model. These examples show that our model is more sensitive to context-dependent sentiment changes, thanks to the use of lexicons as a basis.

5 Conclusion

We proposed a conceptually-simple, yet empirically effective method of introducing sentiment lexicon features to state-of-the-art LSTM models for sentiment analysis. Compared to the simple averag-

ing method in traditional bag-of-word models, our system leverages the strength of semantic feature learning by LSTM models to calculate a context-dependent weight for each word given an input sentence. The method gives competitive results on various sentiment analysis benchmarks. In addition, thanks to the use of lexicons, our model can improve the cross-domain robustness of recurrent neural models for sentiment analysis.

Acknowledgments

Yue Zhang is the corresponding author. Thanks to anonymous reviewers for their helpful comments and suggestions. Yue Zhang is supported by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10, pages 2200–2204.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of EMNLP*, pages 793–801.

Kerstin Denecke. 2009. Are sentiwordnet scores suited for multi-domain sentiment classification? In *ICDIM*, pages 1–6. IEEE.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 49–54.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

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

A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks.

Marco Guerini, Lorenzo Gatti, and Marco Turchi. 2013. Sentiment analysis: How to derive prior polarities from SentiWordNet. In *Proceedings of EMNLP*, pages 1259–1269.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*, KDD '04, pages 168–177.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Colm Kearney and Sha Liu. 2014. Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis*, 33:171–185.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. *J. Artif. Intell. Res. (JAIR)*, 50:723–762.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 10–19.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations?

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of EMNLP*, pages 2326–2335.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of SemEval-2013*, June.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague. Ed. and with an Introd. by Richmond H. Thomason*. Yale University Press.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of SemEval-2013*, pages 312–320.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of EMNLP*.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive twitter sentiment classification using neural network. In *Proceedings of AAAI*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of EMNLP*.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Oscar Täckström and Ryan McDonald. 2011. Discovering fine-grained sentiment with latent variable structured prediction models. In *Advances in Information Retrieval*, pages 368–374.

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

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of COLING*, pages 172–182, August.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, pages 1555–1565, June.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, pages 1347–1353, July.

Xin Wang, Yuanchao Liu, Chengjie SUN, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of ACL*, pages 1343–1353.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M Mohammad. 2014. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *CoRR*, abs/1503.04881.

Event-Driven Emotion Cause Extraction with Corpus Construction

Lin Gui[1], **Dongyin Wu**[1], **Ruifeng Xu**[1,2*], **Qin Lu**[3] and **Yu Zhou**[1]
1. School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China
2. Guangdong Provincial Engineering Technology Research Center for Data Science
3. Department of Computing, the Hong Kong Polytechnic University, Hong Kong
guilin.nlp@gmail.com;wudongyinhit@gmail.com;xuruifeng@hitsz.edu.cn;
csluqin@comp.polyu.edu.hk;zhouyu.nlp@gmail.com

Abstract

In this paper, we present our work in emotion cause extraction. Since there is no open dataset available, the lack of annotated resources has limited the research in this area. Thus, we first present a dataset we built using SINA city news. The annotation is based on the scheme of the W3C Emotion Markup Language. Second, we propose a 7-tuple definition to describe emotion cause events. Based on this general definition, we propose a new event-driven emotion cause extraction method using multi-kernel SVMs where a syntactical tree based approach is used to represent events in text. A convolution kernel based multi-kernel SVM are used to extract emotion causes. Because traditional convolution kernels do not use lexical information at the terminal nodes of syntactic trees, we modify the kernel function with a synonym based improvement. Even with very limited training data, we can still extract sufficient features for the task. Evaluations show that our approach achieves 11.6% higher F-measure compared to referenced methods. The contributions of our work include resource construction, concept definition and algorithm development.

1 Introduction

With the rapid growth of Internet, people can easily share experiences and emotions through this powerful medium anywhere and anytime. How to analyze the emotions of individuals through their writings becomes a new challenge for NLP. In recent years, studies in emotion analysis focus on emotion classification including detection of emotions expressed by writers of text (Gao et al., 2013) as well as prediction of reader emotions (Chang et al., 2015). There are also some information extraction tasks in emotion analysis, such as extracting the feeler of emotion (Das and Bandyopadhyay, 2010). However, these methods need to observe emotion linked expressions. Sometimes, however, we care more about the stimuli, or the cause of an emotion. For instance, manufacturers want to know why people love, or hate a certain product. The White House may also prefer to know the cause of the emotional text "Let us hit the streets" rather than the distribution of different emotions.

There are three main challenges in the study of emotion cause extraction. The first is that, up to now, there is no open dataset available for emotion cause extraction. This may explain why there are only few studies on emotion causes. The second is that, there is no formal definition about event in emotion cause extraction even though some researches claim that they extract events of emotion causes (Lee et al., 2010; Chen et al., 2010). The third is that, due to the complexity in annotation, the size of corpus for emotion cause extraction is usually very small. Due to this limitation, many machine learning methods are not suited for emotion cause detection. How to mine deep knowledge of a language for emotion causes is another thorny issue.

In this paper, we first present an annotated dataset for emotion cause extraction to be released to the public. We then propose to use a 7-tuple to define emotion cause events. Based on this general defi-

*corresponding author

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1639–1649,
Austin, Texas, November 1-5, 2016.

nition, we then present a new event-driven emotion cause extraction method. The basic idea is to extract events in the context of emotional text through dependency parsing. Then, a syntactic structure is used to represent nearby events. Based on this structured representation of events, a modified convolution kernel which also takes lexical features(as terminal nodes) is used to determine whether an event is emotion cause relevant. This method can detect all possible combinations of syntactic structures to obtain sufficient features for emotion analysis using a limited training set. Compared to existing methods, which either use manual rules or commonsense knowledge to extend information, our approach is completely machine learning based and it still achieves state-of-the-art performance. The contributions of this work include both resource development and algorithm development.

The rest of the paper is organized as follows. Section 2 provides a review of related works on emotion analysis. Section 3 presents emotion cause related definitions and the construction of emotion cause extraction corpus. Section 4 gives the event-driven emotion cause extraction method and section 5 is the evaluations and discussions. Section 6 concludes this work and gives the future directions.

2 Related Works

Identifying emotion categories in text is an essential subject in NLP and its applications (Liu, 2015). Moreover, emotion causes can provide important information on why there is any emotion changes. In this section, we introduce related works on the emotion analysis and emotion cause extraction.

The first issue in emotion analysis is to determine the taxonomy of emotions. Researchers have proposed a list of primary emotions(Plutchik, 1980; Ekman, 1984; Turner, 2000). In this study, we adopt Ekman's emotion classification (Ekman, 1984), which identifies six primary emotions, namely *happiness, sadness, fear, anger, disgust* and *surprise*, known as the "Big6"[1] scheme in the W3C Emotion Markup Language. This list is agreed upon by most previous works in Chinese emotion analysis.

The second issue is how to do emotion classification and emotion information extraction. Beck (Beck et al., 2014) proposed a Multi-task Gaussian-process based method for emotion classification. Xu (Xu et al., 2012) used a coarse to fine method to classify emotions in Chinese blog. Gao (Gao et al., 2013) proposed a joint model to co-train a polarity classifier and an emotion classifier. Chang (Chang et al., 2015) used linguistic template to predict reader's emotions. Das (Das and Bandyopadhyay, 2010) used an unsupervised method to extract emotion feelers from Bengali blog. There are other studies focused on joint learning with sentiment (Luo et al., 2015; Mohtarami et al., 2013), emotion in tweets or blog (Hasegawa et al., 2013; Qadir and Riloff, 2014; Ou et al., 2014; Liu et al., 2013; Quan and Ren, 2009), and emotional lexicon construction (Yang et al., 2014; Staiano and Guerini, 2014; Mohammad and Turney, 2013). However, these related works all focused on analysis of emotion expressions rather than emotion causes..

Sophia M. Y. Lee first proposed a task on emotion cause extraction (Lee et al., 2010). They manually constructed a corpus from Academia Sinica Balanced Chinese Corpus. Based on this corpus, Chen and Lee (Chen et al., 2010) proposed a rule based method to detect emotion causes. The basic idea is to make linguistic rules for cause extraction. Some studies (Gui et al., 2014; Li and Xu, 2014; Gao et al., 2015) extended the rule based method to in-formal text in Weibo text (Chinese tweets).

Other than rule based methods, Ghazi (Ghazi et al., 2015) used CRFs to extract emotion causes. However, it requires emotion cause and emotion keywords to be in the same sentence. I. Russo (Russo et al., 2011) proposed a crowd-sourcing method to obtain emotion cause related commonsense knowledge. But it is challenging to extend the commonsense knowledgebase automatically.

Resources used in the above works are not publicly accessible. Most of the methods used are rule based. Learning based methods are quite limited because annotated data is quite small in size due to high cost for annotation. Thus, rule based methods seem to be the easiest way to achieve acceptable performance. Since machine learning methods require more knowledge, which is difficult to generalize. So automatic methods only focused on simple text genre.

[1] http://www.w3.org/TR/emotion-voc/xml#big6

3 Construction of Emotion Cause Corpus

In this section, we first describe the linguistic phenomenon in emotion expressions. It serves as the inspiration to develop the annotated dataset. We then introduce details of the annotation scheme, followed by the construction of the dataset.

3.1 Linguistic Phenomenon of Emotion Causes

Emotion causes play an important role in emotion expressions. An emotion cause reveals the stimulus of an emotion. Considering linguistic phenomenon of emotion causes, we follow three basic principles in corpus construction: (1) Keep the whole context of emotion expression; (2) The basic processing unit is at the clause level; and (3) Use of formal text.

In written text, there is an emotion keyword, which is used to express an emotion, in the context of the emotion cause. Thus, finding the appropriate context of emotion keywords in the annotation is the pre-requisite to identify its cause. It is the reason why we keep the whole context of emotion keywords.

Another important kind of cues is the presence of conjunctions and prepositions. These words indicate the discourse information between clauses. In order to make use of discourse information, the basic analysis unit should be at clause level rather than at sentence level.

In the third principle, we choose the formal text in corpus construction. According to the related works, emotion expressions can have overlapping emotion cause and emotion target (Gui et al., 2014) in informal text. This is why some studies even incorporate cause extraction with target identification to improve performance. However, our focus is on emotion cause identification. We use formal news text to avoid the potential mix up.

3.2 Collection and Annotation

We first take 3 years (2013-15) Chinese city news from NEWS SINA[2] containing 20,000 articles as the raw corpus. Based on a list of 10,259 Chinese primary emotion keywords (**keywords** for short) (Xu et al., 2008), we extract 15,687 instances by keyword matching from the raw data. Here, we call the presence of an emotion keyword as an **instance** in the corpus. For each matched keyword, we extract three preceding clauses and three following clauses as the context of an instance. If a sentence has more than 3 clauses in each direction, the context will include the rest of the sentence to make the context complete. For simplicity, we omit cross paragraph context.

Note that the presence of keywords does not necessarily convey emotional information due to different possible reasons such as negative polarity and sense ambiguity. For example, "祝愿/wishes" is an emotion word of "happiness". It can also be the name of a song. Also, the presence of emotion keywords does not necessarily guarantee the existence of emotional cause neither. After removing those irrelevant instances, there are 2,105 instances remain. For each emotional instance, two annotators manually annotate the emotion categories and the cause(es) in the W3C Emotion Markup Language (EML) format. Ex1 shows an example of an annotated emotional sentence in the corpus, presented by the original simplified Chinese, followed by its English translation. To save space, we remove the xml tags in the annotation. The original annotated data is in a subsidiary file[3]. The basic analysis unit is a clause. Emotion cause is marked by <cause>, and the emotion keyword is marked by <keywords>. Emotion type, POS, position and the length of annotation are also annotated in Emotionml format.

Ex.1: 朱某今年*55*岁，*1979*年参加工作时才*19*岁，已有*36*年的手艺。"我当时被分到丹阳南京理发店工作，这是当时丹阳最大的理发店。我在那儿获得了好多证书和荣誉。" *<cause POS="v" Dis="-1">*说起自己的荣誉*</cause>*，朱某很是*<keywords type=happiness>*自豪*</keywords>*。

Mr. Zhu is 55 years old. He started working in 1979 as a barber when he was 19 , and has 36 years of experience. "I was assigned to work at the Barbershop in Danyang, Nanjing. It is the largest barbershop in Danyang. I won many awards and honors there." <cause POS="v" Dis="-1">Talking about his honors</cause>, Mr. Zhu is so <keywords type="happiness"> proud </keywords>.

Ex.1 only contains one cause. However, one keyword may have more than one corresponding emotion causes. In Ex.2, there are two relevant causes

[2] http://news.sina.com.cn/society/

[3] **http://hlt.hitsz.edu.cn/?page_id=694**

Item	Number
Instance	2,105
Clauses	11,799
Emotion Cause	2,167
Document with 1 emotion	2,046
Document with 2 emotion	56
Document with 3 emotion	3

Table 1: Details of the Dataset

Emotion	Number	Percentage
Happiness	544	25.83%
Sadness	567	26.94%
Fear	379	18.00%
Anger	302	14.35%
Disgust	225	10.69%
Surprise	88	4.18%

Table 2: Distribution of Emotion Types

Position	Number	Percentage
Previous 3 clauses	37	1.71%
Previous 2 clauses	167	7.71%
Previous 1 clauses	1,180	54.45%
In the same clauses	511	23.58%
Next 1 clauses	162	7.47%
Next 2 clauses	48	2.22%
Next 3 clauses	11	0.51%
Other	42	1.94%

Table 3: Cause Position of Each Emotion

for one keyword. In our dataset, only 59 instances have two or more causes.

Ex.2: 劝说过程中，消防官兵了解到，该女子是由于*<cause POS="v" Dis="-2">*对方拖欠工程款*</cause>*，*<cause POS="v" Dis="-1">*家中又急需用钱*</cause>*，*<keywords type=sadness>*无奈*</keywords>*才选择跳楼轻生。

During persuasion, firemen realized that the woman attempted suicide because of <cause POS="v" Dis="-2">the hold back of wages by the employer</cause>, and <cause POS="v" Dis="-1">her family asked for money urgently</cause>, she feels <keywords type=sadness>helpless</keywords> and thus

3.3 Details of Dataset and Its Annotations

Each instance in our dataset contains only one emotion keyword and at least one emotion cause. It is ensured that the keyword instance and the causes are relevant. The number of extracted instances, clauses, and emotion causes are listed in Table 1. Note that 97.2% of the instances has only one emotion cause, and instances that have two and three emotion causes hold 2.6% and 0.2% respectively. Table 2 shows the distribution of emotion types and Table 3 shows the distribution of cause positions. In the latter we can see that 78% emotion causes adjoin the emotion keywords at the clause level. Apparently, position plays a very important role in emotion cause extraction. Thus, using distance based features for emotion cause extraction is rational and necessary. Table 4 lists the phrase types of emotion causes. Verbs and verb phrases cover 93% of all cause events. Thus, our learning algorithm mainly focus on them.

Two annotators work independently during the annotation process. The key point is to distinguish clause level and phrase level in cause annotation. The clause level labels the clause which contains the emotion cause. The phrase level determines the boundary of an emotion cause. When two annotators have different opinion on one instance at clause level, we involve a third annotator as the arbitrator. In the phrase level, we use the larger boundary of the two annotations when they have the same annotation at the clause level. We reach 0.9287 for the kappa value on clause level annotation which confirmed the reliability of our annotation.

4 Event-Driven Emotion Cause Extraction

Due to the complexity of annotation in emotion cause identification, the size of annotated corpus is usually small. Since we aim to use machine learning methods to automatically learn and identify causes, we use a convolution kernel to detect all possible combinations in the syntactic structure. This allows learning from syntactic representations for emotion cause extraction. The basic idea of our proposed method is to use a tree-structure representation to capture features for emotion cause identification. For training data, we extract all valid tree structures for each event, referred to as the ETs (Event Trees). If an event is a cause, the corresponding ET is positive. Otherwise, the corresponding ET is negative. Then, we train a convolution kerneland a

POS/phrase type	Number	Percentage
Noun/Noun phrase	147	6.78%
Verb/Verb phrase	2020	93.21%

Table 4: Distribution of the POS Tag

multi-kernel SVMs using the training set to classify candidate ETs in the testing set. Since more than 97% emotion keywords only have one cause, and more than 95% causes are near the emotion keywords, candidate ETs are extracted from the context of emotion keywords. We only choose the ET with the highest probability in the classification result as the emotion cause.

4.1 Event Tree Construction

Even though there are related works on event identification in emotion cause detection, there is no formal definition of events In area of artificial intelligence (AI), researchers, such as Radinsky (Radinsky et al., 2012), gave a formal definition of an event as "action, actor, object, instrument, location and time". In our work, we need to give clear definition of event first.

In emotion cause extraction, the components of an event should be simpler. We are only interested in the action, the actor and the object, which are denoted as P, O_1, O_2, respectively, following the conventions in AI. Since Chinese is a SVO language, the actor is the subject and the action is the verb. The subject and the object of a sentence may have attributes and a predicate may have adverbial and complement. Since these components may also be helpful in emotion cause extraction, we formally define an emotion cause event as a 7-tuple:

$e = (Att_{O_1}, O_1, Adv, P, Cpl, Att_{O_2}, O_2)$.

Here, Att_{O_1} is the attribute of O_1; Att_{O_2} is the attribute of O_2; Adv is the adverbial of the predicate P; and Cpl is P's complement. In case syntactic components are not present, NIL values are used. Note that the main cue in an event is P, the action. So, in our algorithm, we extract all verbs from the text, and use dependency parsing[4] to extract all relevant syntactic components specified in e. Then, we can construct an ET.

An ET has has a fixed height of four levels. The top level is the root node. Since Chinese is a SVO language, the descendant of the root is S(subject), V(verb), and O(object). Then, the seven event components can be categorized and filled up in the relevant slots. (Att_{O_1},O_1) belongs to $S(O_1)$, (Adv,P,Cpl) belong to V, and (Att_{O_2},O_2) belongs to O. Then we can get the ET based on the definition of an event.

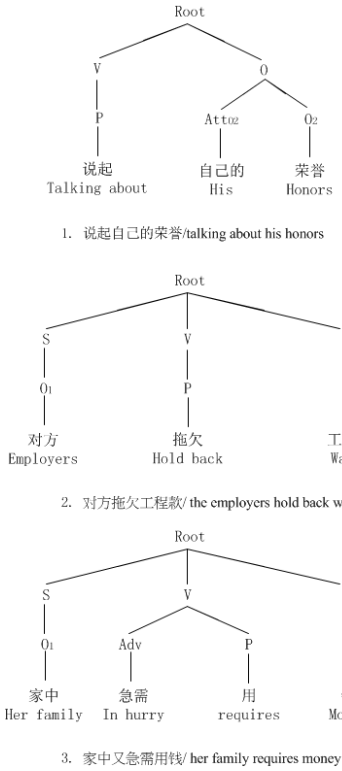


Figure 1: Example ETs of Emotion Causes.

Let us review Ex.1 and Ex.2 again. There are three emotion cause events below with their corresponding ETs shown in Figure 1.

1."说起自己的荣誉/Talking about his honors"

2."对方拖欠工程款/ the hold back wages by employers"

3."家中又急需用钱/ her family asked for money urgently"

After the construction of the ETs, emotion cause extraction becomes a classification problem. If an ET is an emotion cause, the label should be positive. Otherwise, the label should be negative. A binary classifier should be used.

4.2 Emotion Cause Extraction

After the construction of ETs, we obtain positive and negative ET samples. Due to small amount of training samples, it is necessary to capture all features in the ETs. We choose convolution kernel based SVMs because it can search all possible syntactic features under a tree structure.

Convolution kernel function

[4]https://github.com/HIT-SCIR/ltp

The convolution kernel, also known as the tree kernel (Collins and Duffy, 2002), is widely used in many NLP tasks (Srivastava et al., 2013; Moschitti, 2006). For any two inputs T_1 and T_2 based on a tree structure , the kernel is defined as:

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \delta(n_1, n_2). \quad (1)$$

Here, n_1 and n_2 are tree nodes. δ is a function defined recursively:
1.$\delta(n_1, n_2) = 0$ if the productions of n_1 and n_2 are different; 2.Else, $\delta(n_1, n_2) = 1$ if n_1 and n_2 are matching in pre-terminals; 3.Otherwise,

$$\delta(n_1, n_2) = \prod_i (1 + \delta(c(n_1, i), c(n_2, i))).$$

Here, $c(n, i)$ is the *i*-th node of n.

However, the above tree kernel definition does not consider terminals, which means that the actual words in a sentence are ignored. As emotions causes are semantically meaningful, we need to incorporate lexical information into the convolution kernel.

Modified kernel function

In order to distinguish different ETs, we need to modify the definition of the tree kernel to include lexical words in a clause. So we add one more definition to include the terminals:
4.If n_1 and n_2 are terminal nodes, $\delta(n_1, n_2) = 1$ if and only if n_1 and n_2 are synonyms. Otherwise $\delta(n_1, n_2) = 0$.

Here a synonym is defined in Tongyici Cilin (Extended).[5] which has 17,817 synonyms and 77,343 words. We use the synonym rather than word matching because the size of the corpus is limited. simple word matching is quite sparse.

Let K_{ET-O} denote the original kernel and K_{ET-M} denote the modified kernel, respectively. It can be easily proven that K_{ET-M} is a valid kernel function. Following the notation in (Collins and Duffy, 2002), $K_{ET-O} = \sum_i h_i(T_1) \cdot h_i(T_2)$, where $h_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$, $h_i(T_2) = \sum_{n_2 \in N_2} I_i(n_2)$ and the function $I_i(n)$ is 1 if the sub-tree i is rooted at node n and 0 otherwise. So the original tree kernel is an inner product and the kernel matrix is semi-definite. In our modified kernel, the function $I_i(n)$ is more complicated. Beside the definition above, it has the following additional definition : $I_i(n)$ is 1 if i is a terminal node and it is a synonym of n. The new indicator is marked as $I'_i(n)$. Then we have: $K_{ET-M}(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \sum_i I'_i(n_1) I'_i(n_2)$. This means that the modified kernel is symmetrical and the kernel matrix is semi-definite. In our work, K_{ET-M} uses SVM optimization and the code is from SVM-light-TK[6].

Multi-kernel function

Since there are only syntactic information and synonyms in the convolution kernel based method, we need to add some lexical features. Given a 7-tuple event e, we obtain the bag-of-words based or word embedding based representation for each component in e, and the distance between a component and emotion keywords are used as the features, respectively. Let the features of each component in e be R_i, for every $i \in e$. Then, we can capture the feature set, F, of an ET by a joint operation, called the ET features:

$$F = \{R_{Att_{O_1}} \oplus R_{O_1} \oplus ... \oplus R_{Att_{O_2}}\}. \quad (2)$$

We can join the ET features with syntactic information by a multi-kernel function. For any two ETs T_1 and T_2, with the respective features F_1 and F_2, the two new multi-kernels can be defined as:

$$K_{new+O}(T_1, T_2) = K_{ET-O}(T_1, T_2) + K_{vec}(F_1, F_2), \quad (3)$$

$$K_{new*O}(T_1, T_2) = K_{ET-O}(T_1, T_2) \times K_{vec}(F_1, F_2), \quad (4)$$

$$K_{new+M}(T_1, T_2) = K_{ET-M}(T_1, T_2) + K_{vec}(F_1, F_2), \quad (5)$$

$$K_{new*M}(T_1, T_2) = K_{ET-M}(T_1, T_2) \times K_{vec}(F_1, F_2). \quad (6)$$

Here, K_{vec} denotes a kernel function which can be a linear kernel, a polynomial kernel or a Gaussian kernel. The next step is to train the classifier based on the multi-kernel function.

The training data is already in labeled ET format. To prepare testing data, we extract all ETs from a given instance as candidate ETs. A classifier is used to obtain the probability of emotion cause for each ET to produce a ranked list of candidate ETs. The ET with the highest rank serves as the cause event for the current instance.

[5]http://ir.hit.edu.cn/demo/ltp/Sharing_Plan.htm

[6]http://disi.unitn.it/moschitti/Tree-Kernel.htm

5 Performance Evaluations

5.1 Experimental Setup

In the experiments, we stochastically select 90% of the dataset as training data and 10% as testing data. In order to obtain statistically credible results, we evaluate our methods and the reference methods 25 times. We conduct two sets of experiments. The first one evaluates the performance at the clause level to identify the clauses that contain emotion causes. The second one evaluates emotion causes using verb classification. This is because 93.21% of emotion causes are verb/verb phrase and verbs serve as the action component in event definition.

5.2 Emotion Cause Extraction

We use the commonly accepted measure proposed by Lee (Lee et al., 2010) for emotion cause extraction (Gao et al., 2015; Li and Xu, 2014). In this measure, if a roposed emotion cause covers the annotated answer, the sequence is considered correct. Te precision, recall, and F-measure are defined by

$$Precision = \frac{\sum correct\_cause1}{\sum proposed\_cause1},$$
$$Recall = \frac{\sum correct\_cause1}{\sum annotated\_cause1},$$
$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

In the experiment, evaluation is conducted for the following works:

1.**RB**(Rule based method): Among several rule based methods (Lee et al., 2010; Gui et al., 2014; Li and Xu, 2014). We use lee2010's rules (listed in **Appendix** of this paper).

2.**CB**(Commonsense based method): In order to reproduce this method (Russo et al., 2011), we use the Chinese Emotion Cognition Lexicon (Xu et al., 2013) as the commonsense. The lexicon contains more than 5,000 emotion stimulations and their corresponding reflection words.

3.**ML**(Rule base features for machine learning): Rules are used as features with other manual features for emotion cause classification (Chen et al., 2010).

4. K_{vec} : Features are defined in Formula (2) in the training of classifier.

Method	Precision	Recall	F-measure
RB	**0.6747**	0.4287	0.5243
CB	0.2672	**0.7130**	0.3887
RB+CB	0.5435	0.5307	0.5370
RB+CB+ML	0.5921	0.5307	**0.5597**
K_{vec}	0.4200	0.4375	0.4285
$K_{word2vec}$	0.4301	0.4233	0.4136
K_{ET-O}	0.3982	0.4134	0.4057
K_{ET-M}	0.4583	0.4745	0.4662
K_{new+O}	0.6446	0.6779	0.6608
K_{new*O}	0.6492	0.6701	0.6595
K_{new+M}	0.6588	**0.6927**	0.6752
K_{new*M}	**0.6673**	0.6841	**0.6756**

Table 5: Performance on the Dataset

5.$K_{word2vec}$: Word2vec (Mikolov et al., 2013) is used to learn word representation. Use the representation according to Formula (2) in the training of classifier.

6.K_{ET-O} : This is the original tree kernel.

7.K_{ET-M} : This is the modified tree kernel in Formula (1).

8.K_{new+O}, K_{new*O}, K_{new+M} and K_{new*M}: Use the multi-kernel gives by formulas from (3) to (6).

The performance result is given in Table 5. Among all methods, K_{new*M} achieves the top performance in F-measure. Compared to other methods, the improvement is significant with p-value less than 0.01 in t-test.

Even though RB achieves the top precision, its F-measure is limited by the low recall. Since CB is opposite to RB, the performance by RB+CB is improved. However, the improvement is quite limited, at 0.0127 in F-measure. The F-measure of our reproduced RB is similar to mentioned result of other references (Gui et al., 2014; Li and Xu, 2014). They repeat Lee's (Lee et al., 2010) method and achieve the F-measure with 0.55 more or less.

(Chen et al., 2010) reported that by using handcrafted rules as features to train a classifier with some additional features such as conjunction, action and epistemic verbs, performance can be improved significantly. In our experiment, the result is opposite to this claim. The main reason is the samples in (Chen et al., 2010) are less complex. About 85% of the emotion causes are in the same clause where the emotion keywords are. Our corpus is quite different. The percentage of causes in the same clause where the emotion keyword itself is has only about

23.6%. (Chen et al., 2010)'s method does not handle long distance relations well. This explains why it does not work well for our dataset. Although (RB+CB+ML) does not perform well, there is still 0.0334 improvement in F-measure compare to RB. Among our proposed methods, K_{vec} on the ET feature achieves 0.4285 in F-measure. Compare to CB and ML, the performance is not satisfactory. However, as a simple feature to represent lexical information, the performance is acceptable. word2vec also yield similar result. Maybe the joint operation is too simple to handle composition.

For the modified tree kernel K_{ET-M}, the performance is 0.0605 higher than the original tree kernel K_{ET-O} in F-measure. It means that the consideration of terminal node improves the performance of the tree kernel significantly. The modified tree kernel K_{ET-M} is also 0.0377 higher than K_{vec}, and 0.0526 higher than $K_{word2vec}$ in F-measure. This means kernel based syntactic representation does have better generalization ability. The original kernel function K_{ET-O} has syntactic information but no lexicon, and it not only underperforms compared to K_{ET-M} but also K_{vec} and $K_{word2vec}$. This demonstrates our modified kernel function can effectively turn an inferior method into a superior one. Compared to rule based method, the performance still needs to be enhanced and a multi-kernel is necessary. After the combination with ET feature using a multi-kernel, the performance of K_{new*M} achieves a higher level with 0.6756 in F-measure. Compare to RB, the improvement in F-measure is 0.1513. Compare to the combination of existing methods, the improvement is 0.1159. The reason is that our method represents events at the syntactic level. Synonym information gives the model more generalization ability.

5.3 Verb Classification for Emotion Cause

In this section, we examine the performance of ETs classification with respect to verbs identified in the emotion clauses.

ETs Classification

Our method is based on ETs classification to choose the candidate ET with the highest probability. The performance is measured by the verbs in the identified ET. Results are shown in Table 6.

Note that $K_{word2vec}$ performs much better than K_{vec} in verb identification, contrary to their similar performance in clause identification. The reason is that extraction result is based on ranking and only top ranked event affects the performance. In other words, precision is more important than recall here. For the same reason, K_{new+M} is better than K_{new*M} in classification of ETs, although only marginally. Nonetheless, using revised convolution kernel with multi-kernel training is still significantly better than the original kernel K_{new*M} which achieves the best performance in Table 5. When the precision of the two methods are similar, such as K_{ET-O} and K_{ET-M}, the effect of recall becomes important.

Method	Precision	Recall	F-measure
K_{vec}	0.3500	0.2951	0.3192
$K_{word2vec}$	0.3200	0.4833	0.3848
K_{ET-O}	0.3906	0.2773	0.3228
K_{ET-M}	0.3978	0.3303	0.3473
K_{new+O}	0.4211	0.7219	0.5319
K_{new*O}	0.4197	0.7305	0.5331
K_{new+M}	0.4407	**0.7694**	**0.5651**
K_{new*M}	**0.4532**	0.7504	0.5646

Table 6: Performance on ETs Classification

The multi-kernel not only achieves the best performance on both precision and recall, the increase in performance is also significant with at least 0.2173 (between K_{ET-M} and K_{new*M}). Obviously, multi-kernel is not just a simple voting or joint for the components, it benefits from two kernels to achieve better performance.

5.4 Error Analysis

There are mainly three types of errors in our model. We use case examples to show them.

a) Cascading Events

In some cases, events may happen like a chain reaction. An event that leads to an emotion may be the consequent of another event. Identifying the right event in a chain is more challenging. In the following example:

Ex.3: 约兰·沃森坠入冰冷的水中。<*cause*>刺骨的冰水</*cause*>让他感到极其寒冷与<*keywords*>害怕</*keywords*>，约兰·沃森慌忙用不太流利的中文大呼“救命”。

John Watson fell into icy water. <*cause*>*The chilly water*</*cause*> *made him feel so cold and* <*keywords*>*scared*</*keywords*> *John Watson had to*

use his broken Chinese to call for help.

the emotion cause should be "刺骨的冰水*/the chilly water*". Our method output "坠入冰冷的水中*/fell into icy water*" as the emotion cause with probability 60.83%. The probability of the correct cause is 58.89%. As a probability based method, our method does not have the ability to analyze the sequence of events nor the relation between them.

b) Sensory verbs

Sensory verbs usually indicate the emotion cause. There are exceptional cases as shown below:

Ex.4: 了解霸凌事件后。教务主任说，这三名学生知道错了也感到很<*keywords*>害怕</*keywords*>，他们<*cause*>可能面临劳动服务</*cause*>

After investigation on bullying, the head says that the students realized their mistake and were also <keywords>scared</keywords>. They <cause> may need to do community service</cause>

In this case, the cause of "scared" is the punishment of community service. But the template of *"*知道…感到*/realized ... and felt"* usually indicate that there is an emotion cause between the two sensory verbs. Our algorithm gives *"*知道错了*/ realized their mistake"* a probability of 61.65% as a cause, although this is incorrect. But, this actually indicates that our method can learn latent patterns in text.

c) Coverage of cause candidates

In the construction of ETs, we use actions as the cue to construct candidate events. However, 6.78% of our clauses do not have action words. So, these clauses are not selected as candidates.

6 Conclusion

In this paper, we present our work on emotion cause extraction. Due to the lack of open resources for this area of study, we first construct an annotated dataset from news text which will be released for public use. We also propose an event-driven emotion cause extraction method to capture the triggering events emotion changes. In this method, we propose a 7-tuple representation of events using syntactic structures to identify events. Based on this structured representation of events and the inclusion of lexical features, a convolution kernel based learning method is designed to train a multi-kernel classifier to identify emotion cause events. Compared to manually constructed rules and commonsense knowledge based methods, our proposed model can automatically obtain structure features and lexical features to achieve state-of-the-art performance on this dataset.

Acknowledgment

This work was supported by the National Natural Science Foundation of China 61370165, 61632011, National 863 Program of China 2015AA015405, Shenzhen Peacock Plan Research Grant KQCX20140521144507925 and Shenzhen Foundational Research Funding JCYJ20150625142543470, Guangdong Provincial Engineering Technology Research Center for Data Science 2016KF09. The project is also partially supported by HK GRF grant PolyU 152111/14E.

Appendix

No.	Rules
1	i) C(B/F) + I(F) + E(F) + K(F) ii) E = the nearest Na/Nb/Nc/Nh after I in F iii) C = the nearest (N)+(V)+(N) before I in F/B
2	i) E(B/F) + II/IV/V/VI(B/F) + C(B/F) + K(F) ii) E=the nearest Na/Nb/Nc/Nh before II/IV/V/VI in B/F iii) C = the nearest (N)+(V)+(N) before K in F
3	i) II/IV/V/VI (B) + C(B) + E(F) + K(F) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) after II/IV/V/VI in B
4	i) E(B/F) + K(F) + IV/VII(F) + C(F/A) ii) E = a: the nearest Na/Nb/Nc/Nh before K in F; b: the first Na/Nb/Nc/Nh in B iii) C = the nearest (N)+(V)+(N) after IV/VII in F/A
5	i) E(F)+K(F)+VI(A)+C(A) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) after VI in A
6	i) I(F) + E(F) + K(F) + C(F/A) ii) E = the nearest Na/Nb/Nc/Nh after I in F iii) C = the nearest (N)+(V)+(N) after K in F or A
7	i) E(B/F) + yue4 C yue4 K "the more C the more K" (F) ii) E = the nearest Na/Nb/Nc/Nh before the first yue4 in B/F iii) C = the V in between the two yue4' s in F
8	i) E(F) + K(F) + C(F) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) after K in F
9	i) E(F) + IV(F) + K(F) ii) E = the nearest Na/Nb/Nc/Nh before IV in F iii) C = IV+(an aspectual marker) in F
10	i) K(F) + E(F) + de "possession"(F) + C(F) ii) E = the nearest Na/Nb/Nc/Nh after K in F iii) C = the nearest (N)+V+(N)+"的"+N after de in F
11	i) C(F) + K(F) + E(F) ii) E = the nearest Na/Nb/Nc/Nh after K in F iii) C = the nearest (N)+(V)+(N) before K in F
12	i) E(B) + K(B) + III (B) + C(F) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) after III in F
13	i) III(B) + C(B) + E(F) + K(F) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) after III in B
14	i) C(B) + E(F) + K(F) ii) E = the nearest Na/Nb/Nc/Nh before K in F iii) C = the nearest (N)+(V)+(N) before K in B
15	i) E(B) +C(B) + K(F) ii) E = the first Na/Nb/Nc/Nh in B iii) C = the nearest (N)+(V)+(N) before K in B

Table 7: Linguistic Rules in **RB**

Here, C = Cause event; E = Experiencer; K = Keyword/emotion verb; B = Clause before the focus clause; F = Focus clause/the clause containing the emotion verb; A = Clause after the focus clause; I to VII are cue words in (Lee et al., 2010); Na/Nb/Nc/Nh is common noun/proper noun/place noun/pronoun.

References

Daniel Beck, Trevor Cohn, and Lucia Specia. 2014. Joint emotion analysis via multi-task gaussian processes. In *EMNLP*, pages 1798–1803.

Yung-Chun Chang, Cen-Chieh Chen, Yu-Lun Hsieh, and WL Hsu. 2015. Linguistic template extraction for recognizing reader-emotion and emotional resonance writing assistance. *ACL-IJCNLP*, pages 775–780.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 179–187. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Dipankar Das and Sivaji Bandyopadhyay. 2010. Finding emotion holder from bengali blog texts—an unsupervised syntactic approach. In *PACLIC*, pages 621–628.

Paul Ekman. 1984. Expression and the nature of emotion. *Approaches to emotion*, 3:19–344.

Wei Gao, Shoushan Li, Sophia Yat Mei Lee, Guodong Zhou, and Chu-Ren Huang. 2013. Joint learning on sentiment and emotion classification. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1505–1508. ACM.

Kai Gao, Hua Xu, and Jiushuo Wang. 2015. A rule-based approach to emotion cause detection for chinese micro-blogs. *Expert Systems with Applications*, 42(9):4517–4528.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Computational Linguistics and Intelligent Text Processing*, pages 152–165. Springer.

Lin Gui, Li Yuan, Ruifeng Xu, Bin Liu, Qin Lu, and Yu Zhou. 2014. Emotion cause detection with linguistic construction in chinese weibo text. In *Natural Language Processing and Chinese Computing*, pages 457–464. Springer.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *ACL (1)*, pages 964–972.

Sophia Yat Mei Lee, Ying Chen, and Chu-Ren Huang. 2010. A text-driven rule-based system for emotion cause detection. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 45–53. Association for Computational Linguistics.

Weiyuan Li and Hua Xu. 2014. Text-based emotion classification using emotion cause extraction. *Expert Systems with Applications*, 41(4):1742–1749.

Huanhuan Liu, Shoushan Li, Guodong Zhou, Chu-Ren Huang, and Peifeng Li. 2013. Joint modeling of news reader's and comment writer's emotions. In *ACL (2)*, pages 511–515.

Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Kun-Hu Luo, Zhi-Hong Deng, Liang-Chen Wei, and Hongliang Yu. 2015. Jeam: A novel model for cross-domain sentiment classification based on emotion analysis. In *EMNLP*, pages 2503–2508.

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

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Mitra Mohtarami, Man Lan, and Chew Lim Tan. 2013. Probabilistic sense sentiment similarity through hidden emotions. In *ACL (1)*, pages 983–992.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer.

Gaoyan Ou, Wei Chen, Tengjiao Wang, Zhongyu Wei, Binyang Li, Dongqing Yang, and Kam-Fai Wong. 2014. Exploiting community emotion for microblog event detection. In *EMNLP*, pages 1159–1168.

Robert Plutchik. 1980. Emotion: A psychoevolutionary synthesis.

Ashequl Qadir and Ellen Riloff. 2014. Learning emotion indicators from tweets: Hashtags, hashtag patterns, and phrases. In *EMNLP*, pages 1203–1209.

Changqin Quan and Fuji Ren. 2009. Construction of a blog emotion corpus for chinese emotional expression analysis. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1446–1454. Association for Computational Linguistics.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning to predict from textual data. *Journal of Artificial Intelligence Research*, pages 641–684.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. Emocause: an easy-adaptable approach to emotion cause contexts. In *Proceedings of the 2nd Workshop on*

Computational Approaches to Subjectivity and Sentiment Analysis, pages 153–160. Association for Computational Linguistics.

Shashank Srivastava, Dirk Hovy, and Eduard H Hovy. 2013. A walk-based semantically enriched tree kernel over distributed word representations. In *EMNLP*, pages 1411–1416.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprint arXiv:1405.1605*.

Jonathan H Turner. 2000. *On the origins of human emotions: A sociological inquiry into the evolution of human affect*. Stanford University Press Stanford, CA.

Linhong Xu, Hongfei Lin, Yu Pan, Hui Ren, and Jianmei Chen. 2008. Constructing the affective lexicon ontology. *Journal of the China Society for Scientific and Technical Information*, 27(2):180–185.

Jun Xu, Ruifeng Xu, Qin Lu, and Xiaolong Wang. 2012. Coarse-to-fine sentence-level emotion classification based on the intra-sentence features and sentential context. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2455–2458. ACM.

Ruifeng Xu, Chengtian Zou, Yanzhen Zheng, Xu Jun, Lin Gui, Bin Liu, and Xiaolong Wang. 2013. A new emotion dictionary based on the distinguish of emotion expression and emotion cognition. *Journal of Chinese Information Processing*, 27(6):82–90.

Min Yang, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *ACL (2)*, pages 421–426.

Neural Sentiment Classification with User and Product Attention

Huimin Chen[1], **Maosong Sun**[1,2]*, **Cunchao Tu**[1], **Yankai Lin**[1], **Zhiyuan Liu**[1]
[1]Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing, China
[2]Beijing Advanced Innovation Center for Imaging Technology,
Capital Normal University, Beijing, China

Abstract

Document-level sentiment classification aims to predict user's overall sentiment in a document about a product. However, most of existing methods only focus on local text information and ignore the global user preference and product characteristics. Even though some works take such information into account, they usually suffer from high model complexity and only consider word-level preference rather than semantic levels. To address this issue, we propose a hierarchical neural network to incorporate global user and product information into sentiment classification. Our model first builds a hierarchical LSTM model to generate sentence and document representations. Afterwards, user and product information is considered via attentions over different semantic levels due to its ability of capturing crucial semantic components. The experimental results show that our model achieves significant and consistent improvements compared to all state-of-the-art methods. The source code of this paper can be obtained from `https://github.com/thunlp/NSC`.

1 Introduction

Sentiment analysis aims to analyze people's sentiments or opinions according to their generated texts and plays a critical role in the area of data mining and natural language processing. Recently, sentiment analysis draws increasing attention of researchers with the rapid growth of online review sites such as Amazon, Yelp and IMDB, due to its importance to personalized recommendation.

In this work, we focus on the task of document-level sentiment classification, which is a fundamental problem of sentiment analysis. Document-level sentiment classification assumes that each document expresses a sentiment on a single product and targets to determine the overall sentiment about the product.

Most existing methods take sentiment classification as a special case of text classification problem. Such methods treat annotated sentiment polarities or ratings as categories and apply machine learning algorithms to train classifiers with text features, e.g., bag-of-words vectors (Pang et al., 2002). Since the performance of text classifiers heavily depends on the extracted features, such studies usually attend to design effective features from text or additional sentiment lexicons (Ding et al., 2008; Taboada et al., 2011).

Motivated by the successful utilization of deep neural networks in computer vision (Ciresan et al., 2012), speech recognition (Dahl et al., 2012) and natural language processing (Bengio et al., 2006), some neural network based sentiment analysis models are proposed to learn low-dimensional text features without any feature engineering (Glorot et al., 2011; Socher et al., 2011; Socher et al., 2012; Socher et al., 2013; Kim, 2014). Most proposed neural network models take the text information in a sentence or a document as input and generate the semantic representations using well-designed neural networks. However, these methods only focus

*Corresponding author: M. Sun (sms@tsinghua.edu.cn)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1650–1659,
Austin, Texas, November 1-5, 2016.

on the text content and ignore the crucial characteristics of users and products. It is a common sense that the user's preference and product's characteristics make significant influence on the ratings.

To incorporate user and product information into sentiment classification, (Tang et al., 2015b) bring in a text preference matrix and a representation vector for each user and product into CNN sentiment classifier. It modifies the word meaning in the input layer with the preference matrix and concatenates the user/product representation vectors with generated document representation before softmax layer. The proposed model achieves some improvements but suffers the following two problems: (1) The introduction of preference matrix for each user/product is insufficient and difficult to be well trained with limited reviews. For example, most users in IMDB and Yelp only have several tens of reviews, which is not enough to obtain a well-tuned preference matrix. (2) The characteristics of user and product should be reflected on the semantic level besides the word level. For example, a two star review in Yelp said "great place to grab a steak and I am a huge fan of the hawaiian pizza $\cdots$ but I don't like to have to spend 100 bucks for a diner and drinks for two". It's obvious that the poor rating result mainly relies on the last sentence compared with others.

To address these issues, we propose a novel hierarchical LSTM model to introduce user and product information into sentiment classification. As illustrated in Fig. 1, our model mainly consists of two parts. Firstly, we build a hierarchical LSTM model to generate sentence-level representation and document-level representation jointly. Afterwards, we introduce user and product information as attentions over different semantic levels of a document. With the consideration of user and product information, our model can significantly improve the performance of sentiment classification in several real-world datasets.

To summarize, our effort provide the following three contributions:

(1) We propose an effective Neural Sentiment Classification model by taking global user and product information into consideration. Comparing with (Tang et al., 2015b), our model contains much less parameters and is more efficient for training.

(2) We introduce user and product information based attentions over different semantic levels of a document. Traditional attention-based neural network models only take the local text information into consideration. In contrast, our model puts forward the idea of user-product attention by utilizing the global user preference and product characteristics.

(3) We conduct experiments on several real-world datasets to verify the effectiveness of our model. The experimental results demonstrate that our model significantly and consistently outperforms other state-of-the-art models.

2 Related Work

With the trends of deep learning in computer vision, speech recognition and natural language processing, neural models are introduced into sentiment classification field due to its ability of text representation learning. (Glorot et al., 2011) use Stacked Denoising Autoencoder in sentiment classification for the first time. Socher conducts a series of recursive neural network models to learn representations based on the recursive tree structure of sentences, including Recursive Autoencoder (RAE) (Socher et al., 2011), Matrix-Vector Recursive Neural Network (MV-RNN) (Socher et al., 2012) and Recursive Neural Tensor Network (RNTN) (Socher et al., 2013). Besides, (Kim, 2014) and (Johnson and Zhang, 2014) adopt convolution neural network (CNN) to learn sentence representations and achieve outstanding performance in sentiment classification.

Recurrent neural network also benefits sentiment classification because it is capable of capturing the sequential information. (Li et al., 2015), (Tai et al., 2015) investigate tree-structured long-short term memory (LSTM) networks on text or sentiment classification. There are also some hierarchical models proposed to deal with document-level sentiment classification (Tang et al., 2015a; Bhatia et al., 2015), which generate different levels (e.g., phrase, sentence or document) of semantic representations within a document. Moreover, attention mechanism is also introduced into sentiment classification, which aims to select important words from a sen-

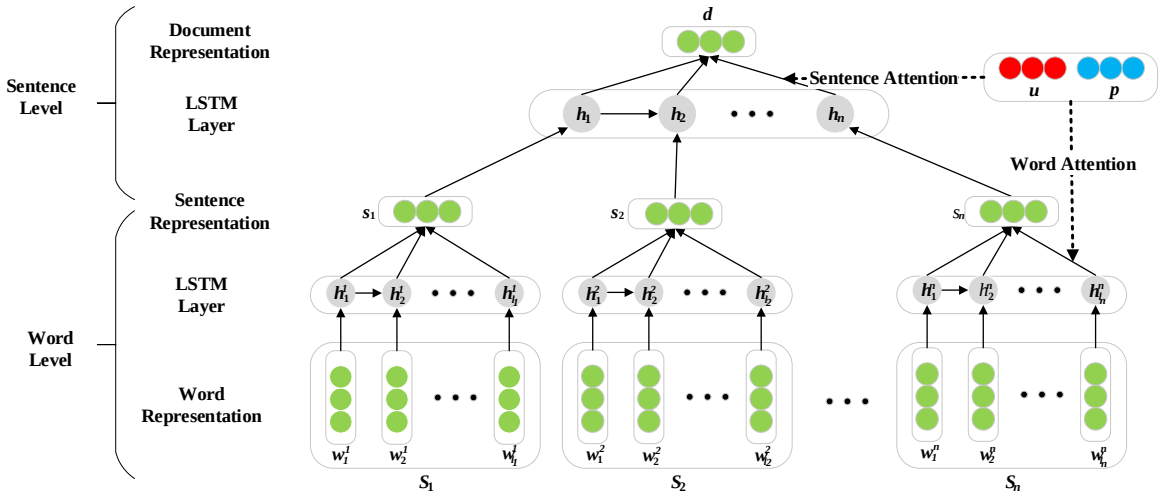


Figure 1: The architecture of User Product Attention based Neural Sentiment Classification model.

tence or sentences from a document (Yang et al., 2016).

Most existing sentiment classification models ignore the global user preference and product characteristics, which have crucial effects on the sentiment polarities. To address this issue, (Tang et al., 2015b) propose to add user/product preference matrices and representation vectors into CNN models. Nevertheless, it suffers from high model complexity and only considers word-level preference rather than semantic levels. In contrast, we propose an efficient neural sentiment classification model with users and products to serve as attentions in both word and semantic levels.

3 Methods

In this section, we will introduce our User Product Attention (UPA) based Neural Sentiment Classification (NSC) model in detail. First, we give the formalizations of document-level sentiment classification. Afterwards, we discuss how to obtain document semantic representation via the Hierarchical Long Short-term Memory (HLSTM) network . At last, we present our attention mechanisms which incorporates the global information of users and products to enhance document representations. The enhanced document representation is used as features for sentiment classification. An overall illustration of UPA based NSC model is shown in Fig. 1.

3.1 Formalizations

Suppose a user $u \in U$ has a review about a product $p \in P$. We represent the review as a document d with n sentences $\{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$. Here, l_i is the length of i-th sentence. The i-th sentence $\mathbf{S}_i$ consists of l_i words as $\{w_1^i, w_2^i, \cdots, w_{l_i}^i\}$. Document-level sentiment classification aims to predict the sentiment distributions or ratings of these reviews according to their text information.

3.2 Neural Sentiment Classification Model

According to the principle of compositionality (Frege, 1892), we model the semantic of a document through a hierarchical structure composed of word-level, sentence-level and document-level. To model the semantic representations of sentences, we adopt Long Short-Term Memory (LSTM) network because of its excellent performance on sentiment classification, especially for long documents. Similarly, we also use LSTM to learn document representations.

In word level, we embed each word in a sentence into a low dimensional semantic space. That means, each word w_j^i is mapped to its embedding $\mathbf{w}_j^i \in \mathbb{R}^d$. At each step, given an input word $\mathbf{w}_j^i$, the current cell state $\mathbf{c}_j^i$ and hidden state $\mathbf{h}_j^i$ can be updated with the previous cell state $\mathbf{c}_{j-1}^i$ and hidden state $\mathbf{h}_{j-1}^i$ as

follows:

$$\begin{bmatrix} \mathbf{i}_j^i \\ \mathbf{f}_j^i \\ \mathbf{o}_j^i \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix} (\mathbf{W} \cdot \left[\ \mathbf{h}_{j-1}^i, \mathbf{w}_j^i \ \right] + \mathbf{b}), \quad (1)$$

$$\hat{\mathbf{c}}_j^i = \tanh(\mathbf{W} \cdot \left[\ \mathbf{h}_{j-1}^i, \mathbf{w}_j^i \ \right] + \mathbf{b}), \quad (2)$$

$$\mathbf{c}_j^i = \mathbf{f}_j^i \odot \mathbf{c}_{j-1}^i + \mathbf{i}_j^i \odot \hat{\mathbf{c}}_j^i, \quad (3)$$

$$\mathbf{h}_j^i = \mathbf{o}_j^i \odot \tanh(\mathbf{c}_j^i), \quad (4)$$

where $\mathbf{i}, \mathbf{f}, \mathbf{o}$ are gate activations, $\odot$ stands for element-wise multiplication, σ is sigmoid function, $\mathbf{W}, \mathbf{b}$ are the parameters we need to train. We then feed hidden states $[\mathbf{h}_1^i, \mathbf{h}_2^i, \cdots, \mathbf{h}_{l_i}^i]$ to an average pooling layer to obtain the sentence representation $\mathbf{s}_i$.

In sentence level, we also feed the sentence embeddings $[\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n]$ into LSTM and then obtain the document representation $\mathbf{d}$ through an average pooling layer in a similar way.

3.3 User Product Attention

We bring in User Product Attention to capture the crucial components over different semantic levels for sentiment classification. Specifically, we employ word-level UPA to generate sentence representations and sentence-level UPA to obtain document representation. We give the detailed implementations in the following parts.

It is obvious that not all words contribute equally to the sentence meaning for different users and products. Hence, in word level, instead of feeding hidden states to an average pooling layer, we adopt a user product attention mechanism to extract user/product specific words that are important to the meaning of sentence. Finally, we aggregate the representations of those informative words to form the sentence representation. Formally, the enhanced sentence representation is a weighted sum of hidden states as:

$$\mathbf{s}_i = \sum_{j=1}^{l_i} \alpha_j^i \mathbf{h}_j^i, \quad (5)$$

where α_j^i measures the importance of the j-th word for current user and product. Here, we embed each user u and each product p as continuous and real-valued vectors $\mathbf{u} \in \mathbb{R}^{d_u}$ and $\mathbf{p} \in \mathbb{R}^{d_p}$, where d_u and d_p are the dimensions of user embeddings and product embeddings respectively. Thus, the attention weight α_j^i for each hidden state can be defined as:

$$\alpha_j^i = \frac{\exp(e(\mathbf{h}_j^i, \mathbf{u}, \mathbf{p}))}{\sum_{k=1}^{l_i} \exp(e(\mathbf{h}_k^i, \mathbf{u}, \mathbf{p}))}, \quad (6)$$

where e is a score function which scores the importance of words for composing sentence representation. The score function e is defined as:

$$\begin{aligned} &e(\mathbf{h}_j^i, \mathbf{u}, \mathbf{p}) = \\ &\mathbf{v}^{\mathrm{T}} \tanh(\mathbf{W}_H \mathbf{h}_{ij} + \mathbf{W}_U \mathbf{u} + \mathbf{W}_P \mathbf{p} + \mathbf{b}), \end{aligned} \quad (7)$$

where $\mathbf{W}_H$, $\mathbf{W}_U$ and $\mathbf{W}_P$ are weight matrices, $\mathbf{v}$ is weight vector and $\mathbf{v}^{\mathrm{T}}$ denotes its transpose.

The sentences that are clues to the meaning of the document vary in different users and products. Therefore, in sentence level, we also use a attention mechanism with user vector $\mathbf{u}$ and product vector $\mathbf{p}$ in word level to select informative sentences to compose the document representation. The document representation $\boldsymbol{d}$ is obtained via:

$$\mathbf{d} = \sum_{i=1}^{n} \beta_i \mathbf{h}_i, \quad (8)$$

where β_i is the weight of hidden state $\boldsymbol{h_i}$ in sentence level which can be calculated similar to the word attention.

3.4 Sentiment Classification

Since document representation $\boldsymbol{d}$ is hierarchically extracted from the words and sentences in the documents, it is a high level representation of the document. Hence, we regard it as features for document sentiment classification. We use a non-linear layer to project document representation $\boldsymbol{d}$ into the target space of C classes:

$$\hat{\mathbf{d}} = \tanh(\mathbf{W}_c \mathbf{d} + \mathbf{b}_c). \quad (9)$$

Afterwards, we use a softmax layer to obtain the document sentiment distribution:

$$p_c = \frac{\exp(\hat{d}_c)}{\sum_{k=1}^{C} \exp(\hat{d}_k)}, \quad (10)$$

where C is the number of sentiment classes, p_c is the predicted probability of sentiment class c. In

Datasets	#classes	#docs	#users	#products	#docs/user	#docs/product	#sens/doc	#words/sen
IMDB	10	84,919	1,310	1,635	64.82	51.94	16.08	24.54
Yelp 2014	5	231,163	4,818	4,194	47.97	55.11	11.41	17.26
Yelp 2013	5	78,966	1,631	1,633	48.42	48.36	10.89	17.38

Table 1: Statistics of IMDB, Yelp2013 and Yelp2014 datasets

our model, cross-entropy error between gold sentiment distribution and our model's sentiment distribution is defined as loss function for optimization when training:

$$L = -\sum_{d \in D} \sum_{c=1}^{C} p_c^g(d) \cdot \log(p_c(d)), \qquad (11)$$

where p_c^g is the gold probability of sentiment class c with ground truth being 1 and others being 0, D represents the training documents.

4 Experiments

In this section, we introduce the experimental settings and empirical results on the task of document-level sentiment classification.

4.1 Experimental Settings

We evaluate the effectiveness of our NSC model on three sentiment classification datasets with user and product information: IMDB, Yelp 2013 and Yelp 2014, which are built by (Tang et al., 2015b). The statistics of the datasets are summarized in Table 1. We split the datasets into training, development and testing sets in the proportion of 8:1:1, with tokenization and sentence splitting by Stanford CoreNLP (Manning et al., 2014). We use two metrics including $Accuracy$ which measures the overall sentiment classification performance and $RMSE$ which measures the divergences between predicted sentiment classes and ground truth classes. The $Accuracy$ and $RMSE$ metrics are defined as:

$$Accuracy = \frac{T}{N} \qquad (12)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(gd_i - pr_i)^2}{N}}, \qquad (13)$$

where T is the numbers of predicted sentiment ratings that are identical with gold sentiment ratings, N is the numbers of documents and gd_i, pr_i represent the gold sentiment rating and predicted sentiment rating respectively.

Word embeddings could be randomly initialized or pre-trained. We pre-train the 200-dimensional word embeddings on each dataset in (Tang et al., 2015a) with SkipGram (Mikolov et al., 2013). We set the user embedding dimension and product embedding dimension to be 200, initialized to zero. The dimensions of hidden states and cell states in our LSTM cells are set to 200. We tune the hyper parameters on the development sets and use adadelta (Zeiler, 2012) to update parameters when training. We select the best configuration based on performance on the development set, and evaluate the configuration on the test set.

4.2 Baselines

We compare our NSC model with several baseline methods for document sentiment classification:

Majority regards the majority sentiment category in training set as the sentiment category of each document in test set.

Trigram trains a SVM classifier with unigrams, bigrams and trigrams as features.

TextFeature extracts text features including word and character n-grams, sentiment lexicon features, etc, and then train a SVM classifier.

UPF extracts use-leniency features (Gao et al., 2013) and corresponding product features from training data, which is further concatenated with the features in **Trigram** an **TextFeature**.

AvgWordvec averages word embeddings in a document to obtain document representation which is fed into a SVM classifier as features.

SSWE generates features with sentiment-specific word embeddings (SSWE) (Tang et al., 2014) and then trains a SVM classifier.

RNTN + RNN represents each sentence with the Recursive Neural Tensor Network (RNTN) (Socher et al., 2013) and feeds sentence representations into

Models	IMDB		Yelp2013		Yelp2014	
	Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
Models without user and product information						
Majority	0.196	2.495	0.411	1.060	0.392	1.097
Trigram	0.399	1.783	0.569	0.814	0.577	0.804
TextFeature	0.402	1.793	0.556	0.845	0.572	0.800
AvgWordvec + SVM	0.304	1.985	0.526	0.898	0.530	0.893
SSWE + SVM	0.312	1.973	0.549	0.849	0.557	0.851
Paragraph Vector	0.341	1.814	0.554	0.832	0.564	0.802
RNTN + Recurrent	0.400	1.764	0.574	0.804	0.582	0.821
UPNN (CNN and no UP)	0.405	1.629	0.577	0.812	0.585	0.808
NSC	0.443	1.465	0.627	**0.701**	**0.637**	**0.686**
NSC + LA	**0.487**	**1.381**	**0.631**	0.706	0.630	0.715
Models with user and product information						
Trigram + UPF	0.404	1.764	0.570	0.803	0.576	0.789
TextFeature + UPF	0.402	1.774	0.561	1.822	0.579	0.791
JMARS	N/A	1.773	N/A	0.985	N/A	0.999
UPNN (CNN)	0.435	1.602	0.596	0.784	0.608	0.764
UPNN (NSC)	0.471	1.443	0.631	0.702	N/A	N/A
NSC+UPA	**0.533**	**1.281**	**0.650**	**0.692**	**0.667**	**0.654**

Table 2: Document-level sentiment classification results. Acc.(Accuracy) and RMSE are the evaluation metrics. The best performances are in bold in both groups.

the Recurrent Neural Network (RNN). Afterwards, the hidden vectors of RNN are averaged to obtain document representation for sentiment classification.

Paragraph Vector implements the PVDM (Le and Mikolov, 2014) for document sentiment classification.

JMARS considers the information of users and aspects with collaborative filtering and topic modeling for document sentiment classification.

UPNN brings in a text preference matrix and a representation vector for each user and product into CNN sentiment classifier (Kim, 2014). It modifies the word meaning in the input layer with the preference matrix and concatenates the user/product representation vectors with generated document representation before softmax layer.

For all baseline methods above, we report the results in (Tang et al., 2015b) since we use the same datasets.

4.3 Model Comparisons

We list the experimental results in Table 2. As shown in this table, we manually divide the results into two parts, the first one of which only considers the local text information and the other one incorporates both local text information and the global user product information.

From the first part in Table 2, we observe that NSC, the basic implementation of our model, significantly outperforms all the other baseline methods which only considers the local text information. To be specific, NSC achieves more than 4% improvements over all datasets compared to typical well-designed neural network models. It demonstrates that NSC is effective to capture the sequential information, which can be a crucial factor to sentiment classification. Moreover, we employ the idea of local semantic attention (LA) in (Yang et al., 2016) and implement it in NSC model (denoted as NSC+LA). The results shows that the attention based NSC obtains a considerable improvements than the original one. It proves the importance of selecting more meaningful words and sentences in sentiment classification, which is also a main reason of introducing global user and product information in an attention form.

In the second part of Table 2, we show the performance of models with user product information. From this part, we have the following observations:

(1) The global user and product information is

Basic Model	Level		IMDB		Yelp2013		Yelp2014	
	Word	Sentence	Acc	RMSE	Acc	RMSE	Acc	RMSE
NSC	AVG	AVG	0.443	1.465	0.627	0.701	0.637	0.686
	AVG	ATT	0.498	1.336	0.632	0.701	0.653	0.672
	ATT	AVG	0.513	1.330	0.640	**0.686**	0.662	0.657
	ATT	ATT	**0.533**	**1.281**	**0.650**	0.692	**0.667**	**0.654**

Table 3: Effect of attention mechanisms in word and sentence level. AVG means an average pooling layer, and ATT represents the attention mechanism in word or sentence level.

Basic Model	Attention Type	IMDB		Yelp2013		Yelp2014	
		Acc	RMSE	Acc	RMSE	Acc	RMSE
NSC	ATT	0.487	1.381	0.631	0.706	0.630	0.715
	PA	0.485	1.456	0.630	0.704	0.644	0.676
	UA	0.525	**1.276**	0.645	0.699	0.644	0.680
	UPA	**0.533**	1.281	**0.650**	**0.692**	**0.667**	**0.654**

Table 4: Effect of user and product attention mechanisms. UA represents the user attention mechanism, and PA indicates the product attention mechanism.

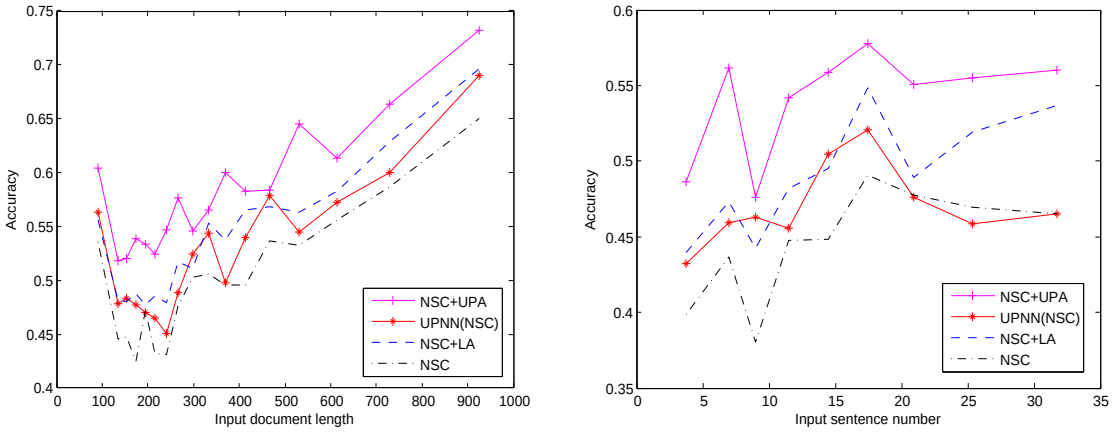
helpful to neural network based models for sentiment classification. With the consideration of such information in IMDB, UPNN achieves 3% improvement and our proposed NSC+UPA obtains 9% improvement in accuracy. The significant improvements state the necessity of considering these global information in sentiment classification.

(2) Our proposed NSC model with user production attention (NSC+UPA) significantly and consistently outperforms all the other baseline methods. It indicates the flexibility of our model on various real-world datasets. Note that, we also implement (Tang et al., 2015b)'s method to deal with user and product information on NSC (denoted as UPNN (NSC)). Though the employment of NSC improves the performance of UPNN, it is still not comparable to our model. More specifically, UPNN exceed the memory of our GPU (12G) when dealing with Yelp2014 dataset due to the high complexity of its parameters. Compared to UPNN which utilizes the user product information with matrices and vectors simultaneously, our model only embeds each user and product as a vector, which makes it suitable to large-scale datasets. It demonstrates that our NSC model is more effective and efficient to handle additional user and product information.

Observations above demonstrate that NSC with user product attention (NSC+UPA) is capable of capturing meanings of multiple semantic layers within a document. Comparing with other user product based models, our model incorporates global user product information in an effective and efficient way. Furthermore, the model is also robust and achieves consistent improvements than state-of-the-art methods on various real-world datasets.

4.4 Model Analysis: Effect of Attention Mechanisms in Word and Sentence Level

Table 3 shows the effect of attention mechanisms in word or sentence level respectively. From the table, we can observe that: (1) Both the attention mechanisms applied in word level and sentence level improve the performance for document sentiment classification compared with utilizing average pooling in word and sentence level; (2) The attention mechanism in word level improves more for our model as compared to sentence level. The reason is that the word attention mechanism can capture the informative words in all documents, while the sentence attention mechanism may only work in long documents with various topics. (3) The model considering both word level attention and sentence level attention outperforms the ones considering only one semantic level attention. It proves that the characteristics of users and products are reflected on multiple semantic levels, which is also a critical motivation of introducing User Product Attention into sentiment classification.



(a) Accuracy over document length

(b) Accuracy over sentence number

Figure 2: Accuracy over various input document lengths on IMDB test set

4.5 Model Analysis: Effect of User Product Attention Mechanisms

Table 4 shows the performance of attention mechanisms with the information of users or products. From the table, we can observe that:

(1) The information of both users and products contributes to our model as compared to a semantic attention. It demonstrates that our attention mechanism can catch the specific characteristic of a user or a product.

(2) The information of users is more effective than the products to enhance document representations. Hence, the discrimination of user preference is more obvious than product characteristics.

4.6 Model Analysis: Performance over Sentence Numbers and Lengths

To investigate the performance of our model over documents with various lengths, we compare the performance of different implementations of NSC under different document lengths and sentence number settings. Fig. 2 shows the accuracy of sentiment classification generated by NSC, NSC+ATT, UPNN(NSC) and NSC+UPA on the IMDB test set with respect to input document lengths and input sentence numbers in a document. From Fig. 2, we observe that our model NSC with attention mechanism of user and product information consistently outperforms other baseline methods for all input document lengths and sentence numbers. It indicates the robustness and flexibility of NSC on different datasets.

4.7 Case Study

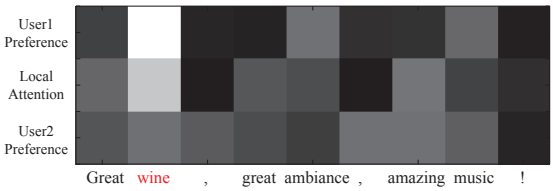


Figure 3: Visualization of attentions over words

To demonstrate the effectiveness of our global attention, we provide a review instance in Yelp2013 dataset for example. The content of this review is "Great wine, great ambiance, amazing music!". We visualize the attention weights in word-level for two distinct users and the local semantic attention (LA) in Fig 3. Here, the local semantic attention represents the implementation in (Yang et al., 2016), which calculates the attention without considering the global information of users and products. Note that, darker color means lower weight.

According to our statistics, the first user often mentions "wine" in his/her review sentences. On the contrary, the second user never talks about "wine" in his/her review sentences. Hence, we infer that the first user may has special preference to wine while the second one has no concern about wine. From the figure, we observe an interesting phenomenon which confirms to our inference. For the word "wine", the first user has the highest atten-

tion weight and the second user has the lowest attention weight. It indicates that our model can capture the global user preference via our user attention.

5 Conclusion and Future Work

In this paper, we propose a hierarchical neural network which incorporates user and product information via word and sentence level attentions. With the user and product attention, our model can take account of the global user preference and product characteristics in both word level and semantic level. In experiments, we evaluate our model on sentiment analysis task. The experimental results show that our model achieves significant and consistent improvements compared to other state-of-the-art models.

We will explore more in future as follows:

(1) In this paper, we only consider the global user preference and product characteristics according to their personal behaviors. In fact, most users and products usually have some text information such as user and product profiles. We will take advantages of those information in sentiment analysis in future.

(2) Aspect level sentiment classification is also a fundamental task in the field of sentiment analysis. The user preference and product characteristics may also implicitly influence the sentiment polarity of the aspect. We will explore the effectiveness of our model on aspect level sentiment classification.

6 Acknowledgements

This work is supported by the National Social Science Foundation of China (13&ZD190) and the National Natural Science Foundation of China (NSFC No. 61331013). We sincerely thank Shiqi Shen and Lei Xu for their insightful discussions, and thank Ayana, Yu Zhao, Ruobing Xie, Jiacheng Zhang and Meng Zhang in Tsinghua University Natural Language Processing group for their constructive comments. We also thank all anonymous reviewers for their insightful suggestions.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of EMNLP*.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Proceedings of CVPR*, pages 3642–3649. IEEE.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio, Speech, and Language Processing*, 20(1):30–42.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of WSDM*, pages 231–240. ACM.

Gottlob Frege. 1892. On sense and reference. In *Ludlow*.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *Proceedings of IJCNLP*, pages 1107–1111.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML*, pages 513–520.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161.

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

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, page 1642.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *CL*, 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of ACL*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings NAACL*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification

Jiacheng Xu[†] **Danlu Chen**[‡] **Xipeng Qiu**[*‡] **Xuanjing Huang**[‡]
Software School, Fudan University [†]
School of Computer Science, Fudan University [‡]
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University[†‡]
825 Zhangheng Road, Shanghai, China[†‡]
{jcxu13,dlchen13,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Recently, neural networks have achieved great success on sentiment classification due to their ability to alleviate feature engineering. However, one of the remaining challenges is to model long texts in document-level sentiment classification under a recurrent architecture because of the deficiency of the memory unit. To address this problem, we present a Cached Long Short-Term Memory neural networks (CLSTM) to capture the overall semantic information in long texts. CLSTM introduces a cache mechanism, which divides memory into several groups with different forgetting rates and thus enables the network to keep sentiment information better within a recurrent unit. The proposed CLSTM outperforms the state-of-the-art models on three publicly available document-level sentiment analysis datasets.

1 Introduction

Sentiment classification is one of the most widely used natural language processing techniques in many areas, such as E-commerce websites, online social networks, political orientation analyses (Wilson et al., 2009; O'Connor et al., 2010), etc.

Recently, deep learning approaches (Socher et al., 2013; Kim, 2014; Chen et al., 2015; Liu et al., 2016) have gained encouraging results on sentiment classification, which frees researchers from handcrafted feature engineering. Among these methods, Recurrent Neural Networks (RNNs) are one of the most prevalent architectures because of the ability to handle variable-length texts.

Sentence- or paragraph-level sentiment analysis expects the model to extract features from limited source of information, while document-level sentiment analysis demands more on selecting and storing global sentiment message from long texts with noises and redundant local pattern. Simple RNNs are not powerful enough to handle the overflow and to pick up key sentiment messages from relatively far time-steps .

Efforts have been made to solve such a scalability problem on long texts by extracting semantic information hierarchically (Tang et al., 2015a; Tai et al., 2015), which first obtain sentence representations and then combine them to generate high-level document embeddings. However, some of these solutions either rely on explicit *a priori* structural assumptions or discard the order information within a sentence, which are vulnerable to sudden change or twists in texts especially a long-range one (McDonald et al., 2007; Mikolov et al., 2013). Recurrent models match people's intuition of reading word by word and are capable to model the intrinsic relations between sentences. By keeping the word order, RNNs could extract the sentence representation implicitly and meanwhile analyze the semantic meaning of a whole document without any explicit boundary.

Partially inspired by neural structure of human brain and computer system architecture, we present the Cached Long Short-Term Memory neural networks (CLSTM) to capture the long-range sentiment information. In the dual store memory model

* Corresponding author.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1660–1669,
Austin, Texas, November 1-5, 2016.

proposed by Atkinson and Shiffrin (1968), memories can reside in the short-term "buffer" for a limited time while they are simultaneously strengthening their associations in long-term memory. Accordingly, CLSTM equips a standard LSTM with a similar cache mechanism, whose internal memory is divided into several groups with different forgetting rates. A group with high forgetting rate plays a role as a cache in our model, bridging and transiting the information to groups with relatively lower forgetting rates. With different forgetting rates, CLSTM learns to capture, remember and forget semantics information through a very long distance.

Our main contributions are as follows:

- We introduce a cache mechanism to diversify the internal memory into several distinct groups with different memory cycles by squashing their forgetting rates. As a result, our model can capture the local and global emotional information, thereby better summarizing and analyzing sentiment on long texts in an RNN fashion.
- Benefiting from long-term memory unit with a low forgetting rate, we could keep the gradient stable in the long back-propagation process. Hence, our model could converge faster than a standard LSTM.
- Our model outperforms state-of-the-art methods by a large margin on three document-level datasets (Yelp 2013, Yelp 2014 and IMDB). It worth noticing that some of the previous methods have utilized extra user and product information.

2 Related Work

In this section, we briefly introduce related work in two areas: First, we discuss the existing document-level sentiment classification approaches; Second, we discuss some variants of LSTM which address the problem on storing the long-term information.

2.1 Document-level Sentiment Classification

Document-level sentiment classification is a sticky task in sentiment analysis (Pang and Lee, 2008), which is to infer the sentiment polarity or intensity of a whole document. The most challenging part is that not every part of the document is equally informative for inferring the sentiment of the whole document (Pang and Lee, 2004; Yessenalina et al., 2010). Various methods have been investigated and explored over years (Wilson et al., 2005; Pang and Lee, 2008; Pak and Paroubek, 2010; Yessenalina et al., 2010; Moraes et al., 2013). Most of these methods depend on traditional machine learning algorithms, and are in need of effective handcrafted features.

Recently, neural network based methods are prevalent due to their ability of learning discriminative features from data (Socher et al., 2013; Le and Mikolov, 2014; Tang et al., 2015a). Zhu et al. (2015) and Tai et al. (2015) integrate a tree-structured model into LSTM for better semantic composition; Bhatia et al. (2015) enhances document-level sentiment analysis by using extra discourse paring results. Most of these models work well on sentence-level or paragraph-level sentiment classification. When it comes to the document-level sentiment classification, a bottom-up hierarchical strategy is often adopted to alleviate the model complexity (Denil et al., 2014; Tang et al., 2015b; Li et al., 2015).

2.2 Memory Augmented Recurrent Models

Although it is widely accepted that LSTM has more long-lasting memory units than RNNs, it still suffers from "forgetting" information which is too far away from the current point (Le et al., 2015; Karpathy et al., 2015). Such a scalability problem of LSTMs is crucial to extend some previous sentence-level work to document-level sentiment analysis.

Various models have been proposed to increase the ability of LSTMs to store long-range information (Le et al., 2015; Salehinejad, 2016) and two kinds of approaches gain attraction. One is to augment LSTM with an external memory (Sukhbaatar et al., 2015; Monz, 2016), but they are of poor performance on time because of the huge external memory matrix. Unlike these methods, we fully exploit the potential of internal memory of LSTM by adjusting its forgetting rates.

The other one tries to use multiple time-scales to distinguish different states (El Hihi and Bengio, 1995; Koutnik et al., 2014; Liu et al., 2015). They partition the hidden states into several groups and each group is activated and updated at different frequencies (e.g. one group updates every 2 time-step,

the other updates every 4 time-step). In these methods, different memory groups are not fully interconnected, and the information is transmitted from faster groups to slower ones, or vice versa.

However, the memory of slower groups are not updated at every step, which may lead to sentiment information loss and semantic inconsistency. In our proposed CLSTM, we assign different forgetting rates to memory groups. This novel strategy enable each memory group to be updated at every time-step, and every bit of the long-term and short-term memories in previous time-step to be taken into account when updating.

3 Long Short-Term Memory Networks

Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a typical recurrent neural network, which alleviates the problem of gradient diffusion and explosion. LSTM can capture the long dependencies in a sequence by introducing a memory unit and a gate mechanism which aims to decide how to utilize and update the information kept in memory cell.

Formally, the update of each LSTM component can be formalized as:

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i\mathbf{x}^{(t)} + \mathbf{U}_i\mathbf{h}^{(t-1)}), \quad (1)$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f\mathbf{x}^{(t)} + \mathbf{U}_f\mathbf{h}^{(t-1)}), \quad (2)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_o\mathbf{x}^{(t)} + \mathbf{U}_o\mathbf{h}^{(t-1)}), \quad (3)$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c\mathbf{x}^{(t)} + \mathbf{U}_c\mathbf{h}^{(t-1)}), \quad (4)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \quad (5)$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}), \quad (6)$$

where σ is the logistic sigmoid function. Operator $\odot$ is the element-wise multiplication operation. $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$, $\mathbf{o}^{(t)}$ and $\mathbf{c}^{(t)}$ are the input gate, forget gate, output gate, and memory cell activation vector at time-step t respectively, all of which have the same size as the hidden vector $\mathbf{h}^{(t)} \in \mathbb{R}^H$. $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o \in \mathbb{R}^{H\times d}$ and $\mathbf{U}_i$, $\mathbf{U}_f$, $\mathbf{U}_o \in \mathbb{R}^{H\times H}$ are trainable parameters. Here, H and d are the dimensionality of hidden layer and input respectively.

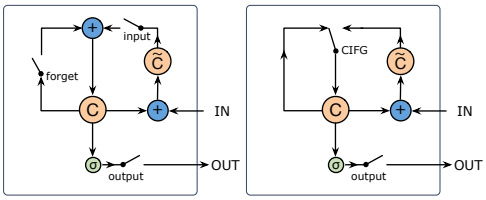


Figure 1: (a) A standard LSTM unit and (b) a CIFG-LSTM unit. There are three gates in (a), the input gate, forget gate and output gates, while in (b), there are only two gates, the CIFG gate and output gate.

4 Cached Long Short-Term Memory Neural Network

LSTM is supposed to capture the long-term and short-term dependencies simultaneously, but when dealing with considerably long texts, LSTM also fails on capturing and understanding significant sentiment message (Le et al., 2015). Specifically, the error signal would nevertheless suffer from gradient vanishing in modeling long texts with hundreds of words and thus the network is difficult to train.

Since the standard LSTM inevitably loses valuable features, we propose a cached long short-term memory neural networks (CLSTM) to capture information in a longer steps by introducing a cache mechanism. Moreover, in order to better control and balance the historical message and the incoming information, we adopt one particular variant of LSTM proposed by Greff et al. (2015), the Coupled Input and Forget Gate LSTM (CIFG-LSTM).

Coupled Input and Forget Gate LSTM Previous studies show that the merged version gives performance comparable to a standard LSTM on language modeling and classification tasks because using the input gate and forget gate simultaneously incurs redundant information (Chung et al., 2014; Greff et al., 2015).

In the CIFG-LSTM, the input gate and forget gate are coupled as one uniform gate, that is, let $\mathbf{i}^{(t)} = \mathbf{1} - \mathbf{f}^{(t)}$. We use $\mathbf{f}^{(t)}$ to denote the coupled gate. Formally, we will replace Eq. 5 as below:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + (\mathbf{1} - \mathbf{f}^{(t)}) \odot \tilde{\mathbf{c}}^{(t)} \quad (7)$$

Figure 1 gives an illustrative comparison of a standard LSTM and the CIFG-LSTM.

Cached LSTM Cached long short-term memory neural networks (CLSTM) aims at capturing the long-range information by a cache mechanism, which divides memory into several groups, and different forgetting rates, regarded as filters, are assigned to different groups.

Different groups capture different-scale dependencies by squashing the scales of forgetting rates. The groups with high forgetting rates are short-term memories, while the groups with low forgetting rates are long-term memories.

Specially, we divide the memory cells into K groups $\{G_1, \cdots, G_K\}$. Each group includes a internal memory $\mathbf{c}_k$, output gate $\mathbf{o}_k$ and forgetting rate $\mathbf{r}_k$. The forgetting rate of different groups are squashed in distinct ranges.

We modify the update of a LSTM as follows.

$$\mathbf{r}_k^{(t)} = \psi_k \left(\sigma(\mathbf{W}_r^k \mathbf{x}^{(t)} + \sum_{j=1}^{K} \mathbf{U}_f^{j \to k} \mathbf{h}_j^{(t-1)}) \right), \tag{8}$$

$$\mathbf{o}_k^{(t)} = \sigma(\mathbf{W}_o^k \mathbf{x}^{(t)} + \sum_{j=1}^{K} \mathbf{U}_o^{j \to k} \mathbf{h}_j^{(t-1)}), \tag{9}$$

$$\tilde{\mathbf{c}}_k^{(t)} = \tanh(\mathbf{W}_c^k \mathbf{x}^{(t)} + \sum_{j=1}^{K} \mathbf{U}_c^{j \to k} \mathbf{h}_j^{(t-1)}), \tag{10}$$

$$\mathbf{c}_k^{(t)} = (1 - \mathbf{r}_k^{(t)}) \odot \mathbf{c}_k^{(t-1)} + (\mathbf{r}_k^{(t)}) \odot \tilde{\mathbf{c}}_k^{(t)}, \tag{11}$$

$$\mathbf{h}_k^{(t)} = \mathbf{o}_k^{(t)} \odot \tanh(\mathbf{c}_k^{(t)}), \tag{12}$$

where $\mathbf{r}_k^{(t)}$ represents forgetting rate of the k-th memory group at step t; ψ_k is a squash function, which constrains the value of forgetting rate $\mathbf{r}_k$ within a range. To better distinguish the different role of each group, its forgetting rate is squashed into a distinct area. The squash function $\psi_k(\mathbf{z})$ could be formalized as:

$$\mathbf{r}_k = \psi_k(\mathbf{z}) = \frac{1}{K} \cdot \mathbf{z} + \frac{k-1}{K}, \tag{13}$$

where $\mathbf{z} \in (0, 1)$ is computed by logistic sigmoid function. Therefore, $\mathbf{r}_k$ can constrain the forgetting rate in the range of $(\frac{k-1}{K}, \frac{k}{K})$.

Intuitively, if a forgetting rate $\mathbf{r}_k$ approaches to $\mathbf{0}$, the group k tends to be the long-term memory; if a $\mathbf{r}_k$ approaches to $\mathbf{1}$, the group k tends to be the short-term memory. Therefore, group G_1 is the slowest, while group G_K is the fastest one. The faster groups are supposed to play a role as a cache, transiting information from faster groups to slower groups.

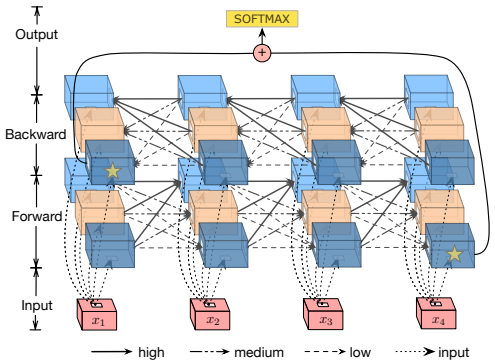


Figure 2: An overview of the proposed architecture. Different styles of arrows indicate different forgetting rates. Groups with stars are fed to a fully connected layers for softmax classification. Here is an instance of B-CLSTM with text length equal to 4 and the number of memory groups is 3.

Bidirectional CLSTM Graves and Schmidhuber (2005) proposed a Bidirectional LSTM (B-LSTM) model, which utilizes additional backward information and thus enhances the memory capability.

We also employ the bi-directional mechanism on CLSTM and words in a text will receive information from both sides of the context. Formally, the outputs of forward LSTM for the k-th group is $[\overrightarrow{\mathbf{h}}_k^{(1)}, \overrightarrow{\mathbf{h}}_k^{(2)}, \ldots, \overrightarrow{\mathbf{h}}_k^{(T)}]$. The outputs of backward LSTM for the k-th group is $[\overleftarrow{\mathbf{h}}_k^{(1)}, \overleftarrow{\mathbf{h}}_k^{(2)}, \ldots, \overleftarrow{\mathbf{h}}_k^{(T)}]$.

Hence, we encode each word w_t in a given text $w_{1:T}$ as $\mathbf{h}_k^{(t)}$:

$$\mathbf{h}_k^{(t)} = \overrightarrow{\mathbf{h}}_k^{(t)} \oplus \overleftarrow{\mathbf{h}}_k^{(t)}, \tag{14}$$

where the $\oplus$ indicates concatenation operation.

Task-specific Output Layer for Document-level Sentiment Classification With the capability of modeling long text, we can use our proposed model to analyze sentiment in a document. Figure 2 gives an overview of the architecture.

Since the first group, the slowest group, is supposed to keep the long-term information and can better represent a whole document, we only utilize the

Dataset	Type	Train Size	Dev. Size	Test Size	Class	Words/Doc	Sents/Doc
IMDB	Document	67426	8381	9112	10	394.6	16.08
Yelp 2013	Document	62522	7773	8671	5	189.3	10.89
Yelp 2014	Document	183019	22745	25399	5	196.9	11.41

Table 1: Statistics of the three datasets used in this paper. The rating scale (Class) of Yelp2013 and Yelp2014 range from 1 to 5 and that of IMDB ranges from 1 to 10. Words/Doc is the average length of a sample and Sents/Doc is the average number of sentences in a document.

final state of this group to represent a document. As for the B-CLSTM, we concatenate the state of the first group in the forward LSTM at T-th time-step and the first group in the backward LSTM at first time-step.

Then, a fully connected layer followed by a softmax function is used to predict the probability distribution over classes for a given input. Formally, the probability distribution $\mathbf{p}$ is:

$$\mathbf{p} = \text{softmax}(\mathbf{W}_p \times \mathbf{z} + \mathbf{b}_p), \tag{15}$$

where $\mathbf{W}_p$ and $\mathbf{b}_p$ are model's parameters. Here $\mathbf{z}$ is $\overrightarrow{\mathbf{h}}_1^{(T)}$ in CLSTM, and $\mathbf{z}$ is $[\overrightarrow{\mathbf{h}}_1^{(T)} \oplus \overleftarrow{\mathbf{h}}_1^{(1)}]$ in B-CLSTM.

5 Training

The objective of our model is to minimize the cross-entropy error of the predicted and true distributions. Besides, the objective includes an L_2 regularization term over all parameters. Formally, suppose we have m train sentence and label pairs $(w_{1:T_i}^{(i)}, y^{(i)})_{i=1}^m$, the object is to minimize the objective function $J(\theta)$:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \log \mathbf{p}_{y^{(i)}}^{(i)} + \frac{\lambda}{2}||\theta||^2, \tag{16}$$

where θ denote all the trainable parameters of our model.

6 Experiment

In this section, we study the empirical result of our model on three datasets for document-level sentiment classification. Results show that the proposed model outperforms competitor models from several aspects when modelling long texts.

6.1 Datasets

Most existing datasets for sentiment classification such as Stanford Sentiment Treebank (Socher et al., 2013) are composed of short paragraphs with several sentences, which cannot evaluate the effectiveness of the model under the circumstance of encoding long texts. We evaluate our model on three popular real-world datasets, Yelp 2013, Yelp 2014 and IMDB. Table 1 shows the statistical information of the three datasets. All these datasets can be publicly accessed[1]. We pre-process and split the datasets in the same way as Tang et al. (2015b) did.

- **Yelp 2013** and **Yelp 2014** are review datasets derived from Yelp Dataset Challenge[2] of year 2013 and 2014 respectively. The sentiment polarity of each review is 1 star to 5 stars, which reveals the consumers' attitude and opinion towards the restaurants.
- **IMDB** is a popular movie review dataset consists of 84919 movie reviews ranging from 1 to 10 (Diao et al., 2014). Average length of each review is 394.6 words, which is much larger than the length of two Yelp review datasets.

6.2 Evaluation Metrics

We use Accuracy (Acc.) and MSE as evaluation metrics for sentiment classification. Accuracy is a standard metric to measure the overall classification result and Mean Squared Error (MSE) is used to figure out the divergences between predicted sentiment labels and the ground truth ones.

6.3 Baseline Models

We compare our model, CLSTM and B-CLSTM with the following baseline methods.

- **CBOW** sums the word vectors and applies a non-linearity followed by a softmax classification layer.

[1] http://ir.hit.edu.cn/~dytang/paper/acl2015/dataset.7z

[2] http://www.yelp.com/dataset_challenge

Model	IMDB		Yelp 2014		Yelp 2013	
	Acc. (%)	MSE	Acc. (%)	MSE	Acc. (%)	MSE
CBOW	34.8	2.867	56.8	0.620	54.5	0.706
PV (Tang et al., 2015b)	34.1	3.291	56.4	0.643	55.4	0.692
RNTN+Recurrent (Tang et al., 2015b)	40.0	3.112	58.2	0.674	57.4	0.646
UPNN (CNN) (Tang et al., 2015b)	40.5	2.654	58.5	0.653	57.7	0.659
JMARS* (Diao et al., 2014)	-	3.143	-	0.998	-	0.970
UPNN (CNN)* (Tang et al., 2015b)	**43.5**	**2.566**	**60.8**	**0.584**	**59.6**	**0.615**
RNN	20.5	6.163	41.0	1.203	42.8	1.144
LSTM	37.8	2.597	56.3	0.592	53.9	0.656
CIFG-LSTM	39.1	2.467	55.2	0.598	57.3	0.558
CLSTM	**42.1**	**2.399**	**59.2**	**0.539**	**59.4**	**0.587**
BLSTM	43.3	2.231	59.2	0.538	58.4	0.583
CIFG-BLSTM	44.5	2.283	60.1	0.527	59.2	0.554
B-CLSTM	**46.2**	**2.112**	**61.9**	**0.496**	**59.8**	**0.549**

Table 2: Sentiment classification results of our model against competitor models on IMDB, Yelp 2014 and Yelp 2013. Evaluation metrics are classification accuracy (Acc.) and MSE. Models with * use user and product information as additional features. Best results in each group are in bold.

Dataset	IMDB	Yelp13	Yelp14
Hidden layer units	120	120	120
Number of groups	3	4	4
Weight Decay	$1e{-4}$	$1e{-4}$	$5e{-4}$
Batch size	128	64	64

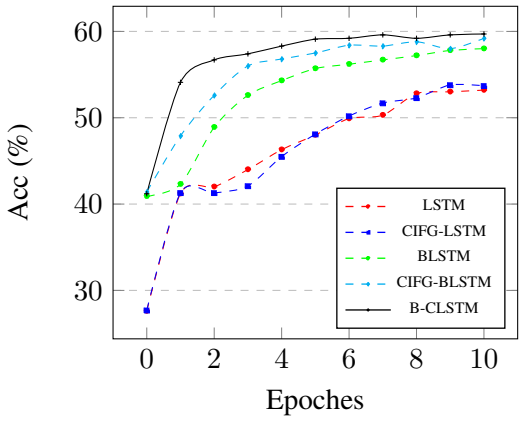
Table 3: Optimal hyper-parameter configuration for three datasets.

- **JMARS** is one of the state-of-the-art recommendation algorithm (Diao et al., 2014), which leverages user and aspects of a review with collaborative filtering and topic modeling.
- **CNN** UPNN (CNN) (Tang et al., 2015b) can be regarded as a CNN (Kim, 2014). Multiple filters are sensitive to capture different semantic features during generating a representation in a bottom-up fashion.
- **RNN** is a basic sequential model to model texts (Elman, 1991).
- **LSTM** is a recurrent neural network with memory cells and gating mechanism (Hochreiter and Schmidhuber, 1997).
- **BLSTM** is the bidirectional version of LSTM, and can capture more structural information and longer distance during looking forward and back (Graves et al., 2013).
- **CIFG-LSTM & CIFG-BLSTM** are Coupled Input Forget Gate LSTM and BLSTM, denoted as CIFG-LSTM and CIFG-BLSTM respectively (Greff et al., 2015). They combine the input and forget gate of LSTM and require smaller number of parameters in comparison with the standard LSTM.

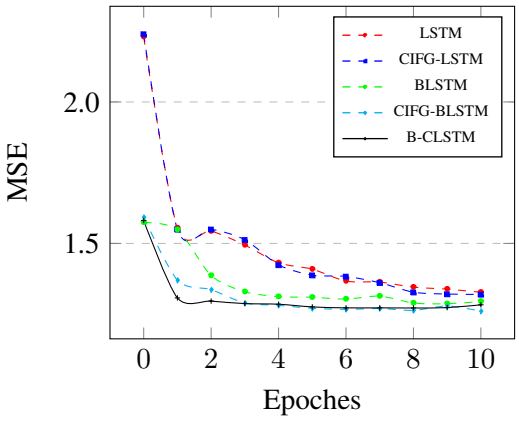
6.4 Hyper-parameters and Initialization

For parameter configuration, we choose parameters on validation set mainly according to classification accuracy for convenience because MSE always has strong correlation with accuracy. The dimension of pre-trained word vectors is 50. We use 120 as the dimension of hidden units, and choose weight decay among $\{ 5e{-4}, 1e{-4}, 1e{-5} \}$. We use Adagrad (Duchi et al., 2011) as optimizer and its initial learning rate is 0.01. Batch size is chosen among $\{$ 32, 64, 128 $\}$ for efficiency. For CLSTM, the number of memory groups is chosen upon each dataset, which will be discussed later. We remain the total number of the hidden units unchanged. Given 120 neurons in all for instance, there are four memory groups and each of them has 30 neurons. This makes model comparable to (B)LSTM. Table 3 shows the optimal hyper-parameter configurations for each dataset.

For model initialization, we initialize all recurrent matrices with randomly sampling from uniform distribution in [-0.1, 0.1]. Besides, we use GloVe(Pennington et al., 2014) as pre-trained word vectors. The word embeddings are fine-tuned during training. Hyper-parameters achieving best results on



(a) Accuracy on Yelp 2013



(b) MSE on Yelp 2013

Figure 3: Convergence speed experiment on Yelp 2013. X-axis is the iteration epoches and Y-axis is the classifcication accuracy(%) achieved.

the validation set are chosen for final evaluation on test set.

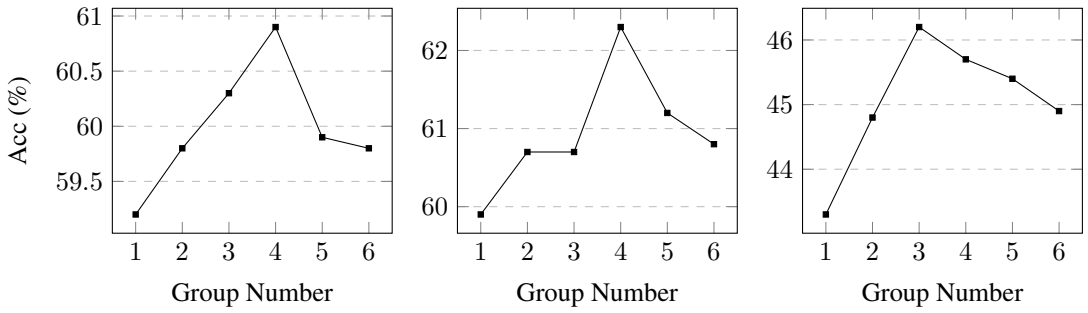
6.5 Results

The classification accuracy and mean square error (MSE) of our models compared with other competitive models are shown in Table 2. When comparing our models to other neural network models, we have several meaningful findings.

1. Among all unidirectional sequential models, RNN fails to capture and store semantic features while vanilla LSTM preserves sentimental messages much longer than RNN. It shows that internal memory plays a key role in text modeling. CIFG-LSTM gives performance comparable to vanilla LSTM.
2. With the help of bidirectional architecture, models could look backward and forward to capture features in long-range from global perspective. In sentiment analysis, if users show their opinion at the beginning of their review, single directional models will possibly forget these hints.
3. The proposed CLSTM beats the CIFG-LSTM and vanilla LSTM and even surpasses the bidirectional models. In Yelp 2013, CLSTM achieves 59.4% in accuracy, which is only 0.4 percent worse than B-CLSTM, which reveals that the cache mechanism has successfully and effectively stored valuable information without the support from bidirectional structure.
4. Compared with existing best methods, our model has achieved new state-of-the-art results by a large margin on all document-level datasets in terms of classification accuracy. Moreover, B-CLSTM even has surpassed JMARS and CNN (UPNN) methods which utilized extra user and product information.
5. In terms of time complexity and numbers of parameters, our model keeps almost the same as its counterpart models while models of hierarchically composition may require more computational resources and time.

6.6 Rate of Convergence

We compare the convergence rates of our models, including CIFG-LSTM, CIFG-BLSTM and B-CLSTM, and the baseline models (LSTM and BLSTM). We configure the hyper-parameter to make sure every competing model has approximately the same numbers of parameters, and various models have shown different convergence rates in Figure 3. In terms of convergence rate, B-CLSTM beats other competing models. The reason why B-CLSTM converges faster is that the splitting memory groups can be seen as a better initialization and constraints during the training process.



(a) Acc on Yelp 2013 (b) Acc on Yelp 2014 (c) Acc on IMDB

Figure 4: Classification accuracy on different number of memory group on three datasets. X-axis is the number of memory group(s).

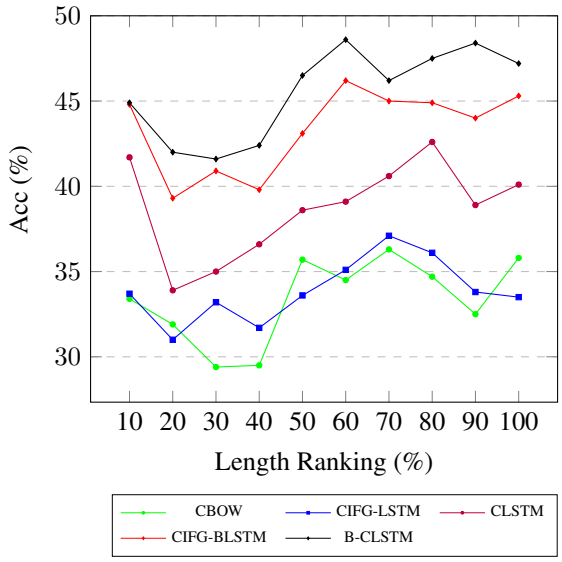


Figure 5: Study of model sensitivity on document length on IMDB. All test samples are sorted by their length and divided into 10 parts. Left most dot means classification accuracy on the shortest 10% samples. X-axis is length ranking from 0% to 100%.

6.7 Effectiveness on Grouping Memory

For the proposed model, the number of memory groups is a highlight. In Figure 4, we plot the best prediction accuracy (Y-axis) achieved in validation set with different number of memory groups on all datasets. From the diagram, we can find that our model outperforms the baseline method. In Yelp 2013, when we split the memory into 4 groups, it achieves the best result among all tested memory group numbers. We can observe the dropping trends when we choose more than 5 groups.

For fair comparisons, we set the total amount of neurons in our model to be same with vanilla LSTM. Therefore, the more groups we split, the less the neurons belongs to each group, which leads to a worse capacity than those who have sufficient neurons for each group.

6.8 Sensitivity on Document Length

We also investigate the performance of our model on IMDB when it encodes documents of different lengths. Test samples are divided into 10 groups with regard to the length. From Figure 5, we can draw several thoughtful conclusions.

1. Bidirectional models have much better performance than the counterpart models.
2. The overall performance of B-CLSTM is better than CIFG-BLSTM. This means that our model is adaptive to both short texts and long documents. Besides, our model shows power in dealing with very long texts in comparison with CIFG-BLSTM.
3. CBOW is slightly better than CIFG-LSTM due to LSTM forgets a large amount of information during the unidirectional propagation.

7 Conclusion

In this paper, we address the problem of effectively analyzing the sentiment of document-level texts in an RNN architecture. Similar to the memory structure of human, memory with low forgetting rate captures the global semantic features while memory with high forgetting rate captures the local semantic features. Empirical results on three real-world

document-level review datasets show that our model outperforms state-of-the-art models by a large margin.

For future work, we are going to design a strategy to dynamically adjust the forgetting rates for fine-grained document-level sentiment analysis.

Acknowledgments

We appreciate the constructive work from Xinchi Chen. Besides, we would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Richard C Atkinson and Richard M Shiffrin. 1968. Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, 2:89–195.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing,(EMNLP)*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. 2015. Sentence modeling with gated recursive neural network. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS Deep Learning Workshop*.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 193–202.

John C Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, pages 493–499.

Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225.

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

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A Search Space Odyssey. *arXiv.org*, March.

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

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *International Conference on Learning Representations (ICLR), Workshop Track*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746-1751*.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. pages 1863–1871.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 2304–2314. The Association for Computational Linguistics.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 432. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv.org*.

Ke Tran Arianna Bisazza Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of NAACL-HLT*, pages 321–331.

Rodrigo Moraes, Joao Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. *ICWSM 2010*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. *EMNLP*, pages 1532–1543.

Hojjat Salehinejad. 2016. Learning over long time lags. *arXiv preprint arXiv:1602.04335*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *ACL*, pages 1556–1566.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. *EMNLP*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning Semantic Representations of Users and Products for Document Level Sentiment Classification. *ACL*, pages 1014–1023.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1056. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612.

Deep Neural Networks with Massive Learned Knowledge

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Eric P. Xing
School of Computer Science
Carnegie Mellon University
{zhitingh,zichaoy,rsalakhu,epxing}@cs.cmu.edu

Abstract

Regulating deep neural networks (DNNs) with human structured knowledge has shown to be of great benefit for improved accuracy and interpretability. We develop a general framework that enables learning knowledge and its confidence jointly with the DNNs, so that the vast amount of fuzzy knowledge can be incorporated and automatically optimized with little manual efforts. We apply the framework to sentence sentiment analysis, augmenting a DNN with massive linguistic constraints on discourse and polarity structures. Our model substantially enhances the performance using less training data, and shows improved interpretability. The principled framework can also be applied to posterior regularization for regulating other statistical models.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in a large variety of application domains (Krizhevsky et al., 2012; Hinton et al., 2012; Bahdanau et al., 2014). However, the powerful end-to-end learning comes with limitations, including the requirement on massive amount of labeled data, uninterpretability of prediction results, and difficulty of incorporating human intentions and domain knowledge.

To alleviate these drawbacks, recent work has focused on training DNNs with extra domain-specific features (Collobert et al., 2011), combining oracle similarity constraints (Karaletsos et al., 2016), modeling output correlations (Deng et al., 2014), and others. Recently, Hu et al. (2016) proposed a general distillation framework that transfers knowledge expressed as first-order logic (FOL) rules into neural networks, where FOL constraints are integrated via posterior regularization (Ganchev et al., 2010). Despite the intuitiveness of FOL rules and the impressive performance in various tasks, the approach, as with the previous posterior constraint methods (Ganchev et al., 2010; Liang et al., 2009; Zhu et al., 2014), has been limited to simple *a priori* fixed constraints with manually selected weights, lacking the ability of inducing and adapting abstract knowledge from data. This issue is further exacerbated in the context of regulating DNNs that map raw data directly into the label space, leaving a huge semantic gap in between, and making it unfeasible to express rich human knowledge built on the intermediate abstract concepts.

In this paper, we introduce a generalized framework which enables a learning procedure for knowledge representations and their weights jointly with the regulated DNN models. This greatly extends the applicability to massive structures in diverse forms, such as structured models and soft logic rules, facilitating practitioners to incorporate rich domain expertise and fuzzy constraints. Specifically, we propose a *mutual* distillation method that iteratively transfers information between DNN and structured knowledge, resulting in effective integration of the representation learning capacity of DNN and the generalization power of structured knowledge. Our method does not require additional supervision beyond raw data-labels for knowledge learning.

We present an instantiation of our method in the task of sentence sentiment analysis. We aug-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1670–1679,
Austin, Texas, November 1-5, 2016.

ment a base convolutional network with linguistic knowledge that encourages coherent sentiment transitions across the clauses in terms of discourse relations. All uncertain modules, such as clause relation and polarity identification, are automatically learned from data, freeing practitioners from exhaustive specification. We further improve the model by integrating thousands of soft word polarity and negation rules, with their confidence directly induced from the data.

Trained with only sentence level supervisions, our model substantially outperforms plain neural networks learned from both sentence and clause labels. Our method also shows enhanced generalization on limited data size, and improved interpretability of predictions.

Our work enjoys general versatility on diverse types of structured knowledge and neural architectures. The principled knowledge and weight learning approach can also be applied to the posterior constraint frameworks (Ganchev et al., 2010; Liang et al., 2009) for regulating other statistical models.

2 Related Work

Deep Networks with Structured Knowledge Combining the powerful deep neural models with structured knowledge has been of increasing interest to enhance generalization and improve interpretability (Li et al., 2015; Deng et al., 2014; Johnson et al., 2016). Recently, Hu et al. (2016) proposed to transfer logical knowledge information into neural networks with diverse architectures (e.g., convolutional networks and recurrent networks). They developed an iterative distillation framework that trains the neural network to emulate the predictions of a "teacher" model which is iteratively constructed by imposing posterior constraints on the network. The framework has shown to be effective in regulating different neural models. However, the method has required fixed constraints and manually specified weights, making it unsuitable to incorporate large amount of fuzzy human intuitions where adaptation to data is necessary to obtain meaningful knowledge representations.

The limitation is in fact shared with the general-purpose posterior regularization methods (Ganchev et al., 2010; Liang et al., 2009; Zhu et al., 2014). Though attempts have been made to learn the constraint weights from additional supervisions (Mei et al., 2014) or for tractability purposes (Steinhardt and Liang, 2015), learning and optimizing knowledge expressions jointly with the regulated models from data is still unsolved, and critically restricting the application scope.

Sentiment Analysis Sentence level sentiment classification is to identify the sentiment polarity (e.g., positive or negative) of a sentence (Pang and Lee, 2008). Recently, a number of neural models have been developed and achieved new levels of performance (Kim, 2014; Socher et al., 2013; Lei et al., 2015). Despite the impressive success, most of the existing neural network approaches require large amount of labeled data while encoding very limited linguistic knowledge, making them inefficient to handle sophisticated linguistic phenomena, such as contrastive transitions and negations (Choi and Cardie, 2008; Bhatia et al., 2015).

Hu et al. (2016) combines a neural network with a logic rule that captures contrastive sense by observing the word "but" in a sentence. However, such simple deterministic rules suffer from limited generality and robustness. This paper develops a new sentiment neural model that combines a large diverse set of linguistic knowledge through our enhanced framework. Our method efficiently captures complex linguistic patterns from limited data, and yields highly interpretable predictions.

3 Mutual Distillation

This section introduces the proposed framework that enables joint learning of knowledge components and their weights with the neural network models. In particular, we generalize the one-sided distillation method of (Hu et al., 2016) (section 3.1), and propose to mutually transfer information between the neural network and the structured constraints for effective knowledge learning (section 3.2), and optimize the weights by considering jointly all components (section 3.3).

We consider input variable $\boldsymbol{x} \in \mathcal{X}$ and target variable $\boldsymbol{y} \in \mathcal{Y}$. For clarity we focus on classification where $\boldsymbol{y}$ is a one-hot encoding of the class labels, though our method also applies to other contexts. Let $(\boldsymbol{X}, \boldsymbol{Y})$ denote a set of instances of $(\boldsymbol{x}, \boldsymbol{y})$.

A neural network defines a conditional probability $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ parameterized by $\boldsymbol{\theta}$. We will omit the subscript θ when there is no ambiguity.

3.1 Network Learning with Knowledge Distillation

We first review the iterative distillation method (Hu et al., 2016) that transfers structured knowledge into neural networks. Consider constraint functions $f_l \in \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, indexed by l, that encode the knowledge and we want to satisfy (i.e., maximize by optimizing the predictions $\boldsymbol{y}$) with confidence weights $\lambda_l \in \mathbb{R}$. Given the current state of the neural network parameters $\boldsymbol{\theta}$ at each iteration, a structure-enriched teacher network q is obtained by solving

$$\min_{q \in \mathcal{P}} \mathrm{KL}(q(\boldsymbol{Y}) \| p_\theta(\boldsymbol{Y}|\boldsymbol{X})) - C \sum\nolimits_l \lambda_l \mathbb{E}_q[f_l(\boldsymbol{X}, \boldsymbol{Y})], \quad (1)$$

where $\mathcal{P}$ denotes the appropriate distribution space; and C is the regularization parameter. Problem (1) is convex and has a closed-form solution

$$q^*(\boldsymbol{Y}) \propto p_\theta(\boldsymbol{Y}|\boldsymbol{X}) \exp\left\{C \sum\nolimits_l \lambda_l f_l(\boldsymbol{X}, \boldsymbol{Y})\right\}, \quad (2)$$

whose normalization term can be calculated efficiently according to how the constraints factorize (Hu et al., 2016). The neural network p_θ at iteration t is then updated with a distillation objective (Hinton et al., 2015) that balances between imitating soft predictions of teacher q and predicting true hard labels:

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} (1-\pi)\ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)) \\ + \pi \ell(\boldsymbol{s}_n^{(t)}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)),\end{aligned} \quad (3)$$

where ℓ denotes the loss function (e.g., cross entropy loss for classification); $\boldsymbol{\sigma}_\theta(\boldsymbol{x})$ is the softmax output of p_θ on $\boldsymbol{x}$; $\boldsymbol{s}_n^{(t)}$ is the soft prediction vector of q on training point $\boldsymbol{x}_n$ at iteration t; N is the training size; and π is the imitation parameter calibrating the relative importance of the two objectives. The training procedure iterates between Eq.(2) and Eq.(3), resulting in the richly structured teacher model q and the knowledge distilled student network p. While q generally provides better accuracy, p is more lightweight and applicable to many different contexts (Hu et al., 2016; Liang et al., 2008).

In (Hu et al., 2016), the constraint $f_l(\boldsymbol{X}, \boldsymbol{Y})$ has been limited to be of the form $r_l(\boldsymbol{X}, \boldsymbol{Y}) - 1$, where r_l is an FOL function yielding truth values in $[0, 1]$, and is required to be fully-specified *a priori* and fixed throughout the training. Besides, the constraint weight λ_l has to be manually selected. This severely deviates from the characters of human knowledge which is usually abstract, fuzzy, built on high-level concepts (e.g., discourse relations, visual attributes) as opposed to low-level observations (e.g., word sequences, image pixels), and thus incomplete in the sense of end-to-end learning that maps raw input directly into target space of interest. This necessitates expressing structured knowledge allowing some modules unknown and induced automatically from observations.

3.2 Knowledge Learning

To substantially extend the scope of knowledge used in the framework, we introduce learnable modules $\boldsymbol{\phi}$ in the knowledge expression denoted as f_ϕ. The module $\boldsymbol{\phi}$ is general, and can be, e.g., free parameters of structured metrics, or dependency structures over semantic units. We assume f_ϕ can be optimized in terms of $\boldsymbol{\phi}$ against a given objective (e.g., through gradient descent for parameter updating). We aim to learn the knowledge by determining $\boldsymbol{\phi}$ from data.

For clarity we consider one knowledge constraint and omit the index l. We further assume the constraint factorizes over data instances. Note that our method can straightforwardly be applied to the case of multiple constraints and constraints spanning multiple instances. As any meaningful knowledge is expected to be consistent with the observations, a straightforward way is then to directly optimize against the training data: $\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \frac{1}{N} \sum_n f_\phi(\boldsymbol{x}_n, \boldsymbol{y}_n)$, and insert the resulting f_{ϕ^*} in Eq.(1) for subsequent steps. However, such a pipelined method fails to establish interactions between the knowledge and network learning, and can lead to a sub-optimal system, as shown in our experiments.

To address this, we inspect the posterior regularization objective in Eq.(1), and write it in an analogous form to the variational free energy of some model evidence. Specifically, let $\log h_\phi(\boldsymbol{X}, \boldsymbol{Y}) \triangleq C\lambda f_\phi(\boldsymbol{X}, \boldsymbol{Y})$, then the objective can be written as

$$-\sum_{\boldsymbol{Y}} q(\boldsymbol{Y}) \log \frac{p(\boldsymbol{Y}|\boldsymbol{X}) h_\phi(\boldsymbol{X}, \boldsymbol{Y})}{q(\boldsymbol{Y})}. \quad (4)$$

Intuitively, we can view the output distribution of the neural network $p(\boldsymbol{Y}|\boldsymbol{X})$ as a prior distribution over the labels, while considering $h_\phi(\boldsymbol{X}, \boldsymbol{Y})$ as defining a "likelihood" metric w.r.t the observations, making the objective analogous to a (negative) variational lower bound of the respective "model". This naturally inspires an EM-type algorithm (Neal and Hinton, 1998) to optimize relevant parameters and improve the "evidence": the E-step optimizes over q, yielding Eq.(2); and the M-step optimizes over $\boldsymbol{\phi}$. Further incorporating the true training labels with balancing parameter π', we obtain the update for $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}^{(t+1)} = \arg\max_{\phi\in\Phi} \frac{1}{N}\sum_{n=1}^{N}(1-\pi')h_\phi(\boldsymbol{x}_n, \boldsymbol{y}_n) + \pi'\mathbb{E}_{q^{(t)}(\boldsymbol{y})}[h_\phi(\boldsymbol{x}_n, \boldsymbol{y})] \tag{5}$$

The update rule resembles the distillation objective for learning parameters $\boldsymbol{\theta}$ in Eq.(3). Indeed, the expectation term in Eq.(5) in effect optimizes h_ϕ on examples labeled by $q(\boldsymbol{y})$, i.e., forcing the knowledge function to mimic the predictions of the teacher model and distill encoded information. Thus, besides transferring from structured knowledge to a neural model by Eq.(3), we now further bridge from the neural network to the knowledge constraints for joint learning and better integrating the best of both worlds. We call our framework with the symmetric objectives as *mutual distillation*. In fact, we can view Eq.(4) as a single joint objective and we are alternating optimization of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, resulting in the update rules in Eq.(3) and Eq.(5) with the supervised loss terms included, respectively (and with the loss function in Eq.(3) being cross-entropy loss).

Additionally, the resemblance of the two objectives indicates that we can readily translate the successful neural learning method to knowledge learning. For instance, the expectation term in Eq.(5), as the second loss term in Eq.(3), can be evaluated on rich unlabeled data in addition to labeled examples, enabling semi-supervised learning which has shown to be useful (Hu et al., 2016). Empirical studies show superiority of the proposed method over several potential alternatives (section 5).

3.3 Weight Learning

Besides optimizing the knowledge representations, we also aim to automate the selection of constraint weights by learning from data. This would enable us to incorporate massive amount of noisy knowledge, without the need to worry about the confidence which is usually unfeasible to set manually.

As the constraint weights serve to balance between the different components of the whole framework, we learn the weights by optimizing the regularized joint model q (see Eq.(2)):

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda}\geq 0} \frac{1}{N}\sum_{n=1}^{N} q_\lambda(\boldsymbol{y}_n) \tag{6}$$

This is also validated in the view of regularized Bayes (Zhu et al., 2014) where q is a generalized posterior function by regularizing the standard posterior p (see Eq.(1)). Although here, we omit the Bayesian treatment of the weights $\boldsymbol{\lambda}$ and instead optimize them directly to find the posterior. It is straightforward to impose priors over $\boldsymbol{\lambda}$ to encode preferences. In practice, Eq. (6) can be carried out through gradient descent.

The training procedure of the proposed mutual distillation is summarized in Algorithm 1.

Algorithm 1 Mutual Distillation

Input: Training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$,
Initial knowledge constraints $\mathcal{F} = \{f_{\phi,l}\}_{l=1}^L$,
Initial neural network p_θ,
Parameters: π, π' – imitation parameters
C – regularization parameters

1: Initialize neural network parameters $\boldsymbol{\theta}$
2: Initialize knowledge parameters $\boldsymbol{\phi}$ and weights $\boldsymbol{\lambda}$
3: **while** not converged **do**
4: Sample a minibatch $(\boldsymbol{X}, \boldsymbol{Y}) \subset \mathcal{D}$
5: Build the teacher model q with Eq.(2) and Eq.(6)
6: Update p_θ with distillation objective Eq.(3)
7: Update f_l $(l = 1, \ldots, L)$ with distillation objective Eq.(5)
8: **end while**

Output: Learned network p, knowledge modules $\mathcal{F}$, and the joint teacher network q

4 Sentiment Classification

This section provides a concrete instance of our general framework in the task of sentence sentiment analysis. We augment a base convolutional network with a large diverse set of linguistic knowledge, including 1) sentiment transition structure for coherent multi-level prediction, 2) conjunction word rules

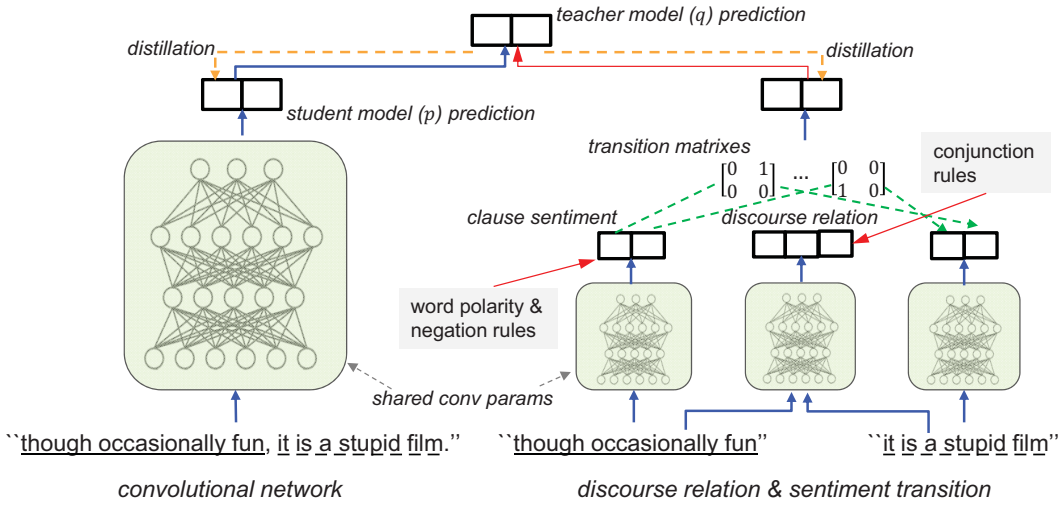


Figure 1: Our sentiment classification model. The left part is the base convolutional network over sentences, and the right part is the knowledge component over clauses. Blue arrows denote neural feed-forwards; red arrows denote knowledge incorporation steps; and the orange dashed arrows denote the distillation processes. The convolutional parameters are shared across all the networks.

for improving discourse relation identification, and 3) word polarity rules for tackling negations. These knowledge structures are fulfilled with neural network modules that are learned jointly within our framework. The resulting model efficiently captures sophisticated linguistic patterns from limited data, and produces interpretable predictions.

Figure 1 shows an overview of our model. We assume binary sentiment labels (i.e., positive-1 and negative-0). The left part of the figure is the base neural network for sentence classification. Since our framework is agnostic to the neural architecture, we can use any off-the-shelf neural models such as convolutional network and recurrent network. Here we choose the simple yet effective convolutional network proposed in (Kim, 2014). The network takes as input the word embedding vectors of a given sentence, and extracts feature maps with a convolutional layer followed by max-over-time pooling. A final fully-connected layer with softmax activation transforms the extracted features into a prediction vector.

We next introduce the three types of domain knowledge, which leverage rich fine-grained level structures, from clauses to words, to guide sentence level prediction. The clause segmentation of sentences is obtained using the public Stanford parser [1].

Sentiment transition by discourse relation Discourse structures characterize how the clauses (i.e., discourse units) of a sentence are connected with each other and thereby provide clues for coherent sentence and clause labeling. Instead of using standard general-purpose discourse relation system, we define three types of relations between adjacent clauses (denoted as c_i and c_{i+1}) specific to sentiment change, namely, *consistent* (c_i and c_{i+1} have the same polarity), *contrastive* (c_{i+1} opposes c_i and is the main part), and *concessive* (c_{i+1} opposes c_i and is secondary). The relations also indicate the connections between clauses and the whole sentence. For instance, a contrastive relation typically indicates c_{i+1} has the same polarity with the full sentence (we reasonably assume a sentence has contrastive sense in at most one position). To encode these dependencies we define sentiment *transition matrices* conditioned on discourse relation r and sentence polarity y, denoted as $M_{r,y}$. For instance, given $r = \textit{contrastive}$ and $y = 0$, we expect the sentiment change between two adjacent clauses to follow

$$M_{r=\textit{contrastive},y=0} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \tag{7}$$

i.e., transiting from positive polarity of c_i to negative of c_{i+1}. We list all transition matrices in supplement.

We now design a constraint on sentence predictions leveraging the above knowledge. Using the identification modules presented shortly, we first get the discourse relation probabilities $\boldsymbol{p}^r_{i,i+1}$ as well as

[1] http://nlp.stanford.edu/software/openie.html

the sentiment polarity probabilities $\boldsymbol{p}_i^c$ and $\boldsymbol{p}_{i+1}^c$ of adjacent clauses (c_i, c_{i+1}). For a given sentence label y_s, we then compute the expected transition matrix at each position by $\bar{M}_{i,y_s} = \mathbb{E}_{\boldsymbol{p}_{i,i+1}^r}[M_{r,y_s}]$. The value of the constraint function on $y = y_s$ is then defined as the probability of the most likely clause polarity configuration according to the clause predictions $\boldsymbol{p}_\cdot^c$ and the averaged transitions $\bar{M}_{\cdot,y_s}$:

$$f^{st}(\boldsymbol{x}, y_s) = \max_{\boldsymbol{a}\in\{0,1\}^m} \prod_i \boldsymbol{p}_{i,a_i}^r \cdot \bar{M}_{i,y_s,a_i a_{i+1}}, \quad (8)$$

where $\boldsymbol{a}$ is the polarity configuration and m is the number of clauses. We use the Viterbi algorithm for efficient computation.

We need the clause relation and polarity probabilities $\boldsymbol{p}^r$ and $\boldsymbol{p}^c$, which are unfeasible to identify from raw text with only simple deterministic rules. We apply a convolutional network for each module, with similar network architectures to the base network (we describe details in the supplement). For efficiency, we tie the convolutional parameters across all the networks, while leaving the parameters of the fully-connected layers to be learned individually.

Conjunction word rules We enhance the discourse relation neural network with robust clues from explicit discourse connectives (e.g., "but", "and", etc.) that occur in the sentence. In particular, we collect a set of conjunction words (listed in the supplement) and specify a rule constraint for each of them. For instance, the conjunction "and" results in the following constraint function:

$$f^{rel}(c_i, c_{i+1}, r) = \left(\mathbf{1}_{\text{and}}(c_i, c_{i+1}) \Rightarrow r = \textit{consistent}\right),$$

where $\mathbf{1}_{\text{and}}(c_i, c_{i+1})$ is an indicator function that takes 1 if the two clauses are connected by "and", and 0 otherwise. Note that these rules are soft, with the confidence weights learned from data. We use the regularized joint model over the base discourse network for predicting the relations.

Negation and word polarity rules Negations reverse the polarity of relevant statements. Identifying negation sense has been a challenging problem for accurate sentiment prediction. We address this by incorporating rich lexicon rules at the clause level. That is, if a polarity-carrying word (e.g., "good") occurs in the scope of a negator (e.g., "not"), then the sentiment prediction of the clause is encouraged to be the opposite polarity. We specify one separate rule for each polarity-carrying word from public lexicons (see the supplement), e.g.,

$$f^{lex}(c_i, y_c) = \left(\mathbf{1}_{\text{good}}(c_i) \Rightarrow y_c = \textit{negative}\right), \quad (9)$$

where $\mathbf{1}_{\text{good}}(c_i)$ is an indicator function that takes 1 if word "good" occurs in a negation scope in the clause text, and 0 otherwise. This results in over 3,000 rules, and our automated weight optimization frees us from manually selecting the weights exhaustively. We define the negation scope to be the 4 words following a negator (Choi and Cardie, 2008).

Though polarities of single words can be brittle features for determining the sentiment of a long statement due to complex semantic compositions, they are more robust and effective at the level of clauses which are generally short and simple. Moreover, inaccurate rules will be downplayed through the weight learning procedure.

We have presented our neural sentiment model. We tackle several long-standing challenges by directly incorporating linguistic knowledge. Comparing to previous work that designs various neural architectures and relies on substantial annotations for specific issues (Socher et al., 2013; Bhatia et al., 2015), our knowledge framework is more straightforward, interpretable, and general, while still preserving the power of neural methods.

Notably, even with several additional components to be learned for knowledge representation, our method does not require extra supervision signals beyond the raw sentence-labels, making our framework generally applicable to many different tasks (Neelakantan et al., 2016).

The sentiment transition knowledge is expressed in the form of structured model with features extracted using neural networks. Though apparently similar to recent deep structured models such as neural-CRFs (Durrett and Klein, 2015; Ammar et al., 2014; Do et al., 2010), ours is different since we parsimoniously extract features that are necessary for precise and efficient knowledge expression, as opposed to neural-CRFs that learn as rich representations as possible for final prediction.

5 Experiments

We evaluate our method on the widely-used sentiment classification benchmarks. Our knowledge

		Model	Accuracy (%)
sentences	1	CNN (Kim, 2014)	86.6
	2	CNN+REL	q: 87.8; p: 87.1
	3	CNN+REL+LEX	q: **88.0**; p: **87.2**
sentences	4	MC-CNN (Kim, 2014)	86.8
	5	Tensor-CNN (Lei et al., 2015)	87.0
	6	CNN+But-q (Hu et al., 2016)	87.1
+phrases	7	CNN (Kim, 2014)	87.2
	8	Tree-LSTM (Tai et al., 2015)	88.0
	9	MC-CNN (Kim, 2014)	88.1
	10	CNN+But-q (Hu et al., 2016)	89.2
	11	MVCNN (Yin and Schutze, 2015)	89.4

Table 1: Classification performance on SST2. The top and second blocks use only sentence-level annotations for training, while the bottom block uses both sentence- and phrases-level annotations. We report the accuracy of both the regularized teacher model q and the student model p after distillation.

enriched model significantly outperforms plain neural networks. We obtain even higher improvements with limited data sizes. Comparison with extensive other potential knowledge learning methods shows the effectiveness of our framework. Our model also shows improved interpretability.

5.1 Setup

Datasets Two classification benchmarks are used: 1) Stanford Sentiment Treebank-2 (**SST2**) (Socher et al., 2013) is a binary classification dataset that consists of 6920/872/1821 moview review sentences in the train/dev/test sets, respectively. Besides sentence-level annotations, the dataset also provides exhaustive gold-standard labels at fine-grained levels, from clauses to phrases. The resulting full training set includes 76,961 labeled instances. We train our model using only the *sentence-level* annotations, and compare to baselines learned from either training set. 2) Customer Reviews (**CR**) (Hu and Liu, 2004) consists of 3,775 product reviews with positive and negative polarities. Following previous work we use 10-fold cross-validation.

Model configurations We evaluate two variants of our model: CNN+REL leverages the knowledge of sentiment transition and discourse conjunctions, and CNN+REL+LEX additionally incorporates the negation lexicon rules.

Throughout the experiments we set the regularization parameter to $C = 10$. The imitation parameters π and π' decay as $\pi^{(t)} = \pi'^{(t)} = 0.9^t$ where t is the iteration number (Bengio et al., 2015; Hu et al., 2016). For the base neural network, we choose the "non-static" version from (Kim, 2014) and use the same configurations.

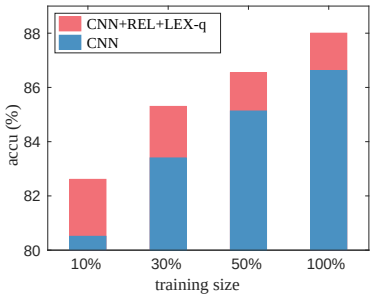


Figure 2: Performance with varying sizes of training examples.

5.2 Classification Results

Table 1 shows the classification performance on the SST2 dataset. From rows 1-3 we see that our proposed sentiment model that integrates the diverse set of knowledge (section 4) significantly outperforms the base CNN (Kim, 2014). The improvement of the student network p validates the effectiveness of the iterative mutual distillation process. Consistent with the observations in (Hu et al., 2016), the regularized teacher model q provides further performance boost, though it imposes additional computational overhead for explicit knowledge representations. Note that our models are trained with only sentence-level annotations. Compared with the baselines trained in the same setting (rows 4-6), our model with the full knowledge, CNN+REL+LEX, performs the best. CNN+But-q (row 6) is the base CNN augmented with a logic rule that identifies contrastive sense through explicit occurrence of word "but" (section 3.1) (Hu et al., 2016). Our enhanced framework enables richer knowledge and achieves much better performance.

Our method further outperforms the base CNN that is additionally trained with dense phrase-level annotations (row 7), showing improved generalization of the knowledge-enhanced model from limited data. Figure 2 further studies the performance with varying training sizes. We can clearly observe that the incorporated knowledge tends to offer higher improvement with less training data. This property can be particularly desirable in applications of structured predictions where manual annotations are expensive while rich human knowledge is available.

	Model	Accuracy (%)
1	CNN (Kim, 2014)	84.1±0.2
2	CNN+REL	q: 85.0±0.2; p: 84.7±0.2
3	CNN+REL+LEX	q: **85.3**±**0.3**; p: **85.0**±**0.2**
4	MC-CNN (Kim, 2014)	85.0
5	Bi-RNN (Lai et al., 2015)	82.6
6	CRF-PR (Yang and Cardie, 2014)	82.7
7	AdaSent (Zhao et al., 2015)	**86.3**

Table 2: Classification performance on the CR dataset. We report the average accuracy±one standard deviation with 10-fold CV. The top block compares the base CNN (row 1) with the knowledge-enhanced CNNs by our framework.

Table 2 shows model performance on the CR dataset. Our model again surpasses the base network and several other competitive neural methods by a large margin. Though falling behind AdaSent (row 7) which has a more specialized and complex architecture than standard convolutional networks, the proposed framework indeed is general enough to apply on top of it for further enhancement.

To further evaluate the proposed mutual distillation framework for learning knowledge, we compare to an extensive set of other possible knowledge optimization approaches. Table 3 shows the results. In row 2, the "opt-joint" method optimizes the regularized joint model of Eq.(2) directly in terms of both the neural network and knowledge parameters. Row 3, "opt-knwl-pipeline", is an approach that first optimizes the standalone knowledge component and then inserts it into the previous framework of (Hu et al., 2016) as a fixed constraint. Without interaction between the knowledge and neural network learning, the pipelined method yields inferior results. Finally, rows 4-5 display a method that adapts the knowledge component at each iteration by optimizing the joint model q in terms of the knowledge parameters. We report the accuracy of both the student network p (row 4) and the joint teacher network q (row 5), and compare with our method in row 6 and 7, respectively. We can see that both models performs poorly, achieving the accuracy of only 68.6% for the knowledge component, similar to the accuracy achieved by the "opt-joint" method.

In contrast, our mutual distillation framework offers the best performance. Table 3 shows that the knowledge component as a standalone classifier does not achieve high accuracy (the numbers in parentheses). As discussed in section 4, this is because of the parsimonious formulation for the precise knowledge expression, while leaving the expressive base NN to extract rich representations. The enhanced performance of the combination indicates complementary effects of the two parts.

	Model	Accuracy (%)
1	CNN (Kim, 2014)	86.6
2	opt-joint	86.9 (68.8)
3	opt-knwl-pipeline	86.7 (70.4)
4	opt-joint-iterative-p	86.9
5	opt-joint-iterative-q	87.6 (68.6)
6	mutual-p	87.2
7	mutual-q	**88.0** (72.5)

Table 3: Comparisons between our mutual distillation (rows 4-5) and other knowledge optimization methods, on SST2. See the text for details. The numbers in parentheses are the accuracy of the learned knowledge component (Figure 1, right part) if we take it as a standalone classifier. All knowledge is used.

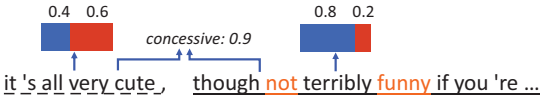


Figure 3: An example sentence and the results of the learned knowledge modules applied on it. Red denotes *positive*, and blue denotes *negative*. The snippet "not ... funny" triggers the negation rule.

enough, good, strong, engaging, great	awful, loses, fake doubt, bad

Table 4: The top 5 positive (left) and negative (right) words with the largest weights of the negation rules.

5.3 Qualitative Analysis

Our model not only provides better classification performance, but also shows improved interpretability due to the learned structured knowledge representation. Figure 3 illustrates an example sentence from test set. We see that the clause sentiments as well as the discourse relation are correctly captured. The negation rule of "not ... funny" (Eq.(9)) also helps to identify the right polarity.

Table 4 lists the top-5 positive and negative words that are most confident for the negation rules, providing insights into the linguistic norms in the movie review context.

6 Conclusion

In this paper we have developed a framework that learns structured knowledge and its weights for regulating deep neural networks through mutual distillation. We instantiated our framework for the sentiment classification task. Using massive learned linguistic knowledge, our neural model provides substantial improvements over many of the existing approaches, especially in the limited data setting. In the future work, we plan to apply our framework to other text and vision applications.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by NSF IIS1218282, NSF IIS1447676, Air Force FA8721-05-C-0003.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Proc. of NIPS*, pages 3311–3319.

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. of NIPS*, pages 1171–1179.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proc. of EMNLP*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proc. of EMNLP*, pages 793–801. Association for Computational Linguistics.

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

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. 2014. Large-scale object classification using label relation graphs. In *ECCV 2014*, pages 48–64. Springer.

Trinh Do, Thierry Arti, et al. 2010. Neural conditional random fields. In *Proc. of AISTATS*, pages 177–184.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of KDD*, pages 168–177. ACM.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proc. of ACL*.

Matthew J. Johnson, David K. Duvenaud, Alex B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. 2016. Composing graphical models with neural networks for structured representations and fast inference. *Arxiv preprint arXiv:1603.06277*.

Theofanis Karaletsos, Serge Belongie, Cornell Tech, and Gunnar Rätsch. 2016. Bayesian representation learning with oracle constraints. In *Proc. of ICLR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proc. of EMNLP*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*, pages 1097–1105.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proc. of EMNLP*.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations?

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proc. of ICML*, pages 592–599. ACM.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proc. of ICML*, pages 641–648. ACM.

Shike Mei, Jun Zhu, and Jerry Zhu. 2014. Robust RegBayes: Selectively incorporating first-order logic domain knowledge into Bayesian models. In *Proc. of ICML*, pages 253–261.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proc. of ICLR*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, volume 1631, page 1642. Citeseer.

Jacob Steinhardt and Percy S Liang. 2015. Learning with relaxed supervision. In *Proc. of NIPS*, pages 2809–2817.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proc. of ACL*, pages 325–335.

Wenpeng Yin and Hinrich Schutze. 2015. Multichannel variable-size convolution for sentence classification. *Proc. of CONLL*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent svms. *JMLR*, 15(1):1799–1847.

De-Conflated Semantic Representations

Mohammad Taher Pilehvar and **Nigel Collier**
Language Technology Lab
Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
{mp792,nhc30}@cam.ac.uk

Abstract

One major deficiency of most semantic representation techniques is that they usually model a word type as a single point in the semantic space, hence conflating all the meanings that the word can have. Addressing this issue by learning distinct representations for individual meanings of words has been the subject of several research studies in the past few years. However, the generated sense representations are either not linked to any sense inventory or are unreliable for infrequent word senses. We propose a technique that tackles these problems by de-conflating the representations of words based on the deep knowledge that can be derived from a semantic network. Our approach provides multiple advantages in comparison to the previous approaches, including its high coverage and the ability to generate accurate representations even for infrequent word senses. We carry out evaluations on six datasets across two semantic similarity tasks and report state-of-the-art results on most of them.

1 Introduction

Modeling the meanings of linguistic items in a machine-interpretable form, i.e., semantic representation, is one of the oldest, yet most active, areas of research in Natural Language Processing (NLP). The field has recently experienced a resurgence of interest with neural network-based models that view the representation task as a language modeling problem and learn dense representations (usually referred to as *embeddings*) by efficiently processing massive amounts of texts. However, either in its conventional count-based form (Turney and Pantel, 2010) or the recent predictive approach, the prevailing objective of representing each word type as a single point in the semantic space has a major limitation: it ignores the fact that words can have multiple meanings and conflates all these meanings into a single representation. This objective can have negative impacts on accurate semantic modeling, e.g., semantically unrelated words that are synonymous to different senses of a word are pulled towards each other in the semantic space (Neelakantan et al., 2014). For example, the two semantically-unrelated words *squirrel* and *keyboard* are pulled towards each other in the semantic space for their similarities to two different senses of *mouse*, i.e., rodent and computer input device.

Recently, there has been a growing interest in addressing the meaning conflation deficiency of word representations. A series of techniques have been developed to associate a word to multiple points in the semantic space by clustering its contexts in a given text corpus and learning distinct representations for individual clusters (Reisinger and Mooney, 2010; Huang et al., 2012). However, these techniques usually assume a fixed number of word senses per word type, disregarding the fact that the number of senses per word can range from one (monosemy) to dozens. Neelakantan et al. (2014) tackled this issue by allowing the number to be dynamically adjusted for each word during training. However, the approach and all the other clustering-based techniques still suffer from the fact that the computed sense representations are not linked to

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1680–1690,
Austin, Texas, November 1-5, 2016.

any sense inventory, a linking which would require large amounts of sense-annotated data (Agirre et al., 2006). In addition, because of their dependence on knowledge derived from a text corpus, these techniques are generally unable to learn accurate representations for word senses that are infrequent in the underlying corpus.

Knowledge-based techniques tackle these issues by deriving sense-specific knowledge from external sense inventories, such as WordNet (Fellbaum, 1998), and learning representations that are linked to the sense inventory. These approaches either use sense definitions and employ Word Sense Disambiguation (WSD) to gather sense-specific contexts (Chen et al., 2014; Iacobacci et al., 2015) or take advantage of the properties of WordNet, such as synonymy and direct semantic relations (Rothe and Schütze, 2015). However, the non-optimal WSD techniques and the shallow utilization of knowledge from WordNet do not allow these techniques to learn accurate and high-coverage semantic representations for all senses in the inventory.

We propose a technique that de-conflates a given word representation into its constituent sense representations by exploiting deep knowledge from the semantic network of WordNet. Our approach provides the following three main advantages in comparison to the past work: (1) our representations are linked to the WordNet sense inventory and, accordingly, the number of senses for a word is a dynamic parameter which matches that defined by WordNet; (2) the deep exploitation of WordNet's semantic network allows us to obtain accurate semantic representations, even for word senses that are infrequent in generic text corpora; and (3) our methodology involves only minimal parameter tuning and can be effectively applied to any sense inventory that is viewable as a semantic network and to any word representation technique. We evaluate our sense representations in two tasks: word similarity (both in-context and in-isolation) and cross-level semantic similarity. Experimental results show that the proposed technique can provide consistently high performance across six datasets, outperforming the recent state of the art on most of them.

2 De-Conflated Representations

Preliminaries. Our proposed approach takes a set of pre-trained word representations and uses the graph structure of a semantic lexical resource in order to de-conflate the representations into those of word senses. Therefore, our approach firstly requires a set of pre-trained word representations (e.g., word embeddings). Any model that maps a given word to a fixed-size vector representation (i.e., vector space model) can be used by our approach. In our experiments, we opted for a set of publicly available word embeddings (cf. §3.1).

Secondly, we require a lexical resource whose semantic relations allow us to view it as a graph $G = (V, E)$ where each vertex in the set of vertices V corresponds to a concept and edges in E denote lexico-semantic relationships among these vertices. Each concept $c \in V$ is mapped to a set of word senses by a mapping function $\mu(c) : c \rightarrow \{s_1, \ldots, s_l\}$. WordNet, the *de facto* community standard sense inventory, is a suitable resource that satisfies these properties. WordNet can be readily represented as a semantic graph in which vertices are synsets and edges are the semantic relations that connect these synsets (e.g., hypernymy and meronymy). The mapping function in WordNet maps each synset to the set of synonymous words it contains (i.e., word senses).

2.1 Overview of the approach

Our goal is to compute a semantic representation that places a given word sense in an existing semantic space of words. We achieve this by leveraging word representations as well as the knowledge derived from WordNet. The gist of our approach lies in its computation of a list of *sense biasing words* for a given word sense. To this end, we first analyze the semantic network of WordNet and extract a list of most representative words that can effectively pinpoint the semantics of individual synsets (§2.2). We then leverage an effective technique which learns semantic representations for individual word senses by placing the senses in the proximity of their corresponding sense biasing words (§2.3).

2.2 Determining sense biasing words

Algorithm 1 shows the procedure we use to extract from WordNet a list of sense biasing words for a

Algorithm 1 Get sense biasing words for synset y_t

Require: Graph $G = (V, E)$ of vertices $V = \{y_i\}_{i=1}^{m}$ (of m synsets) and edges E (semantic relationships between synsets)
Require: Function $\mu(y_i)$ that returns for a given synset y_i the words it contains
Require: Target synset $y_t \in V$ for which a sense biasing word sequence is required
Ensure: The sequence $\mathcal{B}_t$ of sense biasing words for synset y_t

1: $\mathcal{B}_t \leftarrow ()$
2: **for all** word w in $\mu(y_t)$ **do**
3: $\quad \mathcal{B}_t \leftarrow \mathcal{B}_t \cup (w)$
4: **for** $y_i \in V : y_i \neq y_t$ **do**
5: $\quad p_i \leftarrow \text{PERSONALIZEDPAGERANK}(y_i, y_t, G)$
6: $(y_h^*)_{h=1}^{m-1} \leftarrow \text{SORT}(V \setminus \{y_t\})$ according to scores p_i
7: **for** $h : 1$ to $m - 1$ **do**
8: $\quad$ **for all** word w in $\mu(y_h^*)$ **do**
9: $\quad\quad$ **if** $w \notin \mathcal{B}_t$ **then**
10: $\quad\quad\quad \mathcal{B}_t \leftarrow \mathcal{B}_t \cup (w)$
11: **return** sequence $\mathcal{B}_t$

given target synset y_t. The algorithm receives as its inputs the semantic graph of WordNet and the mapping function $\mu(\cdot)$, and outputs an ordered list of biasing words $\mathcal{B}_t$ for y_t. The list comprises the most semantically-related words to synset y_t which can best represent and pinpoint its meaning. We leverage a graph-based algorithm for the computation of the sense biasing words.

Specifically, we use the Personalized PageRank (Haveliwala, 2002, PPR) algorithm which has been extensively used by several NLP applications (Yeh et al., 2009; Niemann and Gurevych, 2011; Agirre et al., 2014). To this end, we first represent the semantic network of WordNet as a row-stochastic transition matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ where m is the number of synsets in WordNet ($|V|$). The cell M_{ij} of $\mathbf{M}$ is set to the inverse of the degree of i if there is a semantic relationship between synsets i and j and to zero otherwise. We compute the PPR distribution for a target synset y_t by using the power iteration method $\mathcal{P}^{t+1} = (1 - \sigma)\mathcal{P}^0 + \sigma\mathbf{M}\mathcal{P}^t$, where σ is the damping factor (usually set to 0.85) and $\mathcal{P}^0$ is a one-hot initialization vector with the corresponding dimension of y_t being set to 1.0. The weight p_i in line 5 is the value of the i^{th} dimension of the PPR vector $\mathcal{P}$ computed for the synset y_t. This weight can be seen as the importance of the corresponding synset of the i^{th} dimension (i.e., y_i) to y_t. When applied to a semantic network, such as the WordNet graph, this importance can be interpreted as semantic relevance. Hence, the value of p_i denotes the extent of semantic relatedness between y_i and y_t. We use this notion and retrieve a list of most semantically-related words to y_t.

To achieve this, we sort the synsets $\{y^* \in V : y^* \neq y_t\}$ according to their PPR values $\{p_i\}_{i=1}^{m-1}$ (line 6). We then iterate (lines 7-10) the sorted list (y^*) and for each synset y_h^* append the list $\mathcal{B}_t$ with all the words in y_h^* (i.e., $\mu(y_h^*)$). However, in order to ensure that the words in the target synset y_t appear as the most representative words in $\mathcal{B}_t$, we first assign these words to the list (line 3). Finally, the algorithm returns the ordered list $\mathcal{B}_t$ of sense biasing words for the target synset y_t.

#	Sense biasing words
1	dactyl, finger, toe, thumb, pollex, body_part, nail, minimus, tarsier, webbed, extremity, appendage
2	figure, cardinal_number, cardinal, integer, whole_number, numeration_system, number_system, system_of_numeration, large_integer, constituent, element, digital

Table 1: The top sense biasing words for the synsets containing the anatomical (#1) and numerical (#2) senses of the noun *digit*.

Table 1 shows a sample of top biasing words extracted for the two senses of the noun *digit*: the numerical and the anatomical senses.[1] We explain in §2.3 how we use the sense biasing lists to learn sense-specific representations. Note that the size of the list is equal to the total number of strings in WordNet. However, we observed that taking a very small portion of the top-ranking elements in the lists is enough to generate representations that perform very similarly to those generated when using the full-sized lists (please see §3.1).

2.3 Learning sense representations

Let $\mathcal{V}$ be the set of pre-trained d-dimensional word representations. Our objective here is to compute a set $\mathcal{V}^* = \{v_{s_1}^*, \ldots, v_{s_n}^*\}$ of representations for n word senses $\{s_1, \ldots, s_n\}$ in the same d-dimensional semantic space of words. We achieve this for each sense s_i by de-conflating the representation v_{s_i} of its corresponding lemma and biasing it towards the

[1] The first and third senses of the noun *digit* in WordNet 3.0.

representations of the words in $\mathcal{B}_i$. Specifically, we obtain a representation $v^*_{s_i}$ for a word sense s_i by solving:

$$\arg\min_{v^*_{s_i}} \alpha\ d(v^*_{s_i}, v_{s_i}) + \sum_{b_{ij} \in \mathcal{B}_i} \delta_{ij}\ d(v^*_{s_i}, v_{b_{ij}}) \quad (1)$$

where v_{s_i} and $v_{b_{ij}}$ are the respective word representations ($\in\ \mathcal{V}$) of the lemma of s_i and the j^{th} biasing word in the list of biasing words for s_i, i.e, $\mathcal{B}_i$. The distance $d(v, v')$ between vectors v and v' is measured by squared Euclidean distance $\|v - v'\|^2 = \sum_k (v_k - v'_k)^2$. The first term in Formula 1 requires the representation of the word sense s_i (i.e., $v^*_{s_i}$) to be similar to that of its corresponding lemma, i.e., v_{s_i}, whereas the second term encourages $v^*_{s_i}$ to be in the proximity of its biasing words in the semantic space. The above criterion is similar to the frameworks of Das and Smith (2011) and Faruqui et al. (2015) which, though being convex, is usually solved for efficiency reasons by an iterative method proposed by Bengio et al. (2007). Following these works, we obtain the below equation for computing the representation of a word sense s_i:

$$v^*_{s_i} = \frac{\alpha v_{s_i} + \sum_{b_{ij} \in \mathcal{B}_i} \delta_{ij} v_{b_{ij}}}{\alpha + \sum_j \delta_{ij}}. \quad (2)$$

We define δ_{ij} as $\frac{e^{-\lambda r(i,j)}}{|\mathcal{B}_i|}$ where $r(i, j)$ denotes the rank of the word b_{ij} in the list $\mathcal{B}_i$. This is essentially an exponential decay function that gives more importance to the top-ranking biasing words for s_i. The hyperparameter α denotes the extent to which $v^*_{s_i}$ is kept close to its corresponding lemma representation v_{s_i}. Following Faruqui et al. (2015), we set α to 1. The only parameter to be tuned in our experiments is λ. We discuss the tuning of this parameter in §3.1. The representation of a synset y_i can be accordingly calculated as the centroid of the vectors of its associated word senses, i.e.,

$$\{\frac{v_{y_i}}{\|v_{y_i}\|} : v_{y_i} = \sum_{s \in \mu(y_i)} \hat{v}^*_s, \hat{v}^*_s = \frac{v^*_s}{\|v^*_s\|}\}. \quad (3)$$

As a result of this procedure, we obtain the set $\mathcal{V}^*$ of n sense representations in the same semantic space as word representations in $\mathcal{V}$. In fact, we now have a unified semantic space which enables a direct comparison of the two types of linguistic items. In §3.3 we evaluate our approach in the word to sense similarity measurement framework.

#	Closest words
1	crappie, trout, guitar, shad, walleye, bassist, angler, catfish, trombone, percussion, piano, drummer, saxophone, jigs, fish
2	baritone, piano, guitar, trombone, saxophone, cello, percussion, tenor, saxophonist, clarinet, pianist, vocals, solos, harmonica
3	fish, trout, shrimp, anglers, fishing, bait, guitar, salmon, shark, fisherman, lakes, seafood, drummer, whale, fisheries

Table 2: Ten most similar words to the word *bass* (#1) and two of its senses: music (#2) and fish (#3).

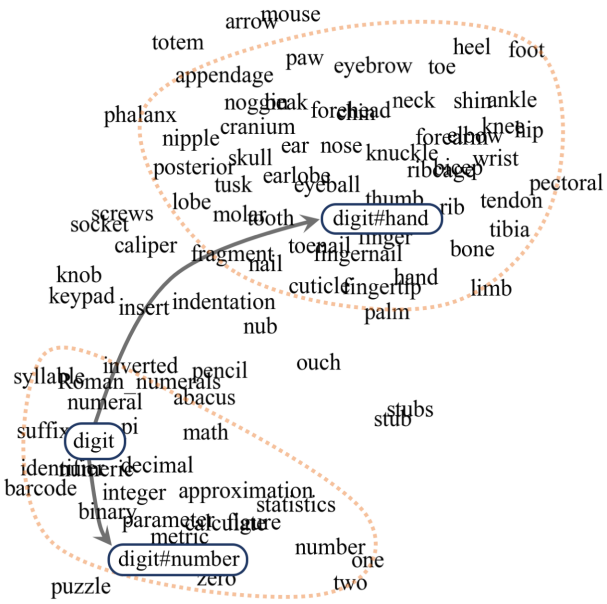


Figure 1: The illustration of the word *digit* and two of its computed senses in our unified 2-d semantic space.

We show in Table 2 the closest words to the word *bass* and two of its senses, music and fish,[2] in our unified semantic space. We can see in row #1 a mixture of both meanings when the word representation is used whereas the closest words to the senses (rows #2 and #3) are mostly in-domain and specific to the corresponding sense.

To exhibit another interesting property of our sense representation approach, we depict in Figure 1 the word *digit* and its numerical and anatomical senses (from the example in Table 1) in a 2-d semantic space, along with a sample set of words in their

[2]The first and fourth senses in WordNet 3.0, respectively defined as "the lowest part of the musical range" and "the lean flesh of a saltwater fish of the family Serranidae."

proximity.[3] We can see that the word *digit* is placed in the semantic space in the neighbourhood of words from the numerical domain (lower left of the figure), mainly due the dominance (Sanderson and Van Rijsbergen, 1999) of this sense in the general-domain corpus on which the word embeddings in our experiments were trained (cf. §3.1). However, upon de-conflation, the emerging anatomical sense of the word is shifted towards the region in the semantic space which is occupied by anatomical words (upper right of the figure). A clustering-based sense representation technique would have failed in accurately representing the infrequent anatomical meaning of *digit* by analyzing a general domain corpus (such as the one used here). But our sense representation technique, thanks to its proper usage of knowledge from a sense inventory, is effective in unveiling and accurately modeling less frequent or domain-specific senses of a given word.

Please note that any vector space model representation technique can be used for the pre-training of word representations in $\mathcal{V}$. Also, the list of sense biasing words can be obtained for larger sense inventories, such as FreeBase (Bollacker et al., 2008) or BabelNet (Navigli and Ponzetto, 2012). We leave the exploration of further ways of computing sense biasing words to future work.

3 Experiments

We benchmarked our sense representation approach against several recent techniques on two standard tasks: word similarity (§3.2), for which we evaluate on both in-isolation and in-context similarity datasets, and cross-level semantic similarity (§3.3).

3.1 Experimental setup

Pre-trained word representations. As our word representations, we used the 300-d Word2vec (Mikolov et al., 2013) word embeddings trained on the Google News dataset[4] mainly for their popularity across different NLP applications. However, our approach is equally applicable to any count-based representation technique (Baroni and Lenci, 2010; Turney and Pantel, 2010) or any other embedding approach (Pennington et al., 2014; LeCun et al., 2015). We leave the evaluation and comparison of various word representation techniques with different training approaches, objectives, and dimensionalities to future work.

Parameter tuning. Recall from §2.3 that our procedure for learning sense representations needs only one parameter to be tuned, i.e., λ. We did not perform an extensive tuning on the value of this parameter and set its value to 1/5 after trying four different values (1, 1/2, 1/5, and 1/10) on a small validation dataset. We leave the more systematic tuning of the parameter and the choice of alternative decay functions (cf. §2.3) to future work.

The size of the sense biasing words lists. Also recall from §2.2 that the extracted lists of sense biasing words were originally as large as the total number of unique strings in WordNet (around 150K in ver. 3.0). But, given that we use an exponential decay function in our learning algorithm (cf. §2.3), the impact of the low-ranking words in the list is negligible. In fact, we observed that taking a very small portion of the top-ranking words, i.e., the top 25, produces similarity scores that are on par with those generated when the full lists were considered. Therefore, we experimented with the down-sized lists which enabled us to generate very quickly sense representations for all word senses in WordNet.

3.2 Word similarity

Comparison systems. We compared our results against nine other sense representation techniques: the WordNet-based approaches of Pilehvar and Navigli (2015), Chen et al. (2014), Rothe and Schütze (2015), Jauhar et al. (2015), and Iacobacci et al. (2015) and the clustering-based approaches of Huang et al. (2012), Tian et al. (2014), Neelakantan et al. (2014), and Liu et al. (2015) (please see §4 for more details). We also compared against the approach of Faruqui et al. (2015) which uses knowledge derived from WordNet for improving word representations. From the different configurations presented in (Faruqui et al., 2015) we chose the system that uses GloVe (Pennington et al., 2014) with *all* WordNet relations which is their best performing monolingual system. As for the approach of Jauhar et al. (2015), we show the results of the

[3] We used the t-SNE algorithm (van der Maaten and Hinton, 2008) for dimensionality reduction.

[4] https://code.google.com/archive/p/word2vec/

EM+RETERO system which performs most consistently across different datasets.

Benchmarks. As our word similarity benchmark, we considered five datasets: RG-65 (Rubenstein and Goodenough, 1965), YP-130 (Yang and Powers, 2005), MEN-3K (Bruni et al., 2014), SimLex-999 (Hill et al., 2015, SL-999), and Stanford Contextual Word Similarity (Huang et al., 2012, SCWS). The latter benchmark provides for each word a context that triggers a specific meaning of it, making the dataset very suitable for the evaluation of sense representation. For each datasets, we list the results that are reported by any of our comparison systems.

Similarity measurement. For the SCWS dataset, we follow the past works (Reisinger and Mooney, 2010; Huang et al., 2012) and report the results according to two system configurations: (1) AvgSim: where the similarity between two words is computed as the average of all the pairwise similarities between their senses, and (2) AvgSimC: where each pairwise sense similarity is weighted by the relevance of each sense to its corresponding context. For all the other datasets, since words are not provided with any context (they are in isolation), we measure the similarity between two words as that between their most similar senses. In all the experiments, we use the cosine distance as our similarity measure.

3.2.1 Experimental results

Tables 4 and 3 show the results of our system, DECONF, and the comparison systems on the SCWS and the other four similarity datasets, respectively. In both tables we also report the word vectors baseline, whenever they are available, which is computed by directly comparing the corresponding word representations of the two words ($\in \mathcal{V}$). Note that the word-based baseline does not apply to the approach of Pilehvar and Navigli (2015) as it is purely based on the semantic network of WordNet and does not use any pre-trained word embeddings.

We can see from the tables that our sense representations obtain considerable improvements over those of words across the five datasets. This highlights the fact that the de-conflation of word representations into those of their individual meanings has been highly beneficial. On the SCWS dataset, DECONF outperforms all the recent state-of-the-art sense representation techniques (in their best settings) which proves the effectiveness of our approach in capturing the semantics of specific meanings of the words. The improvement is consistent across both system configurations (i.e., AvgSim and AvgSimC). Moreover, the state-of-the-art WordNet-based approach of Rothe and Schütze (2015) uses the same initial word vectors as DECONF does (cf. §3.1). Hence, the improvement we obtain indicates that our approach has made better use of the sense-specific knowledge encoded in WordNet.

As seen in Table 3 our approach shows competitive performance on the other four datasets. The YP-130 dataset focuses on verb similarity, whereas SimLex-999 contains verbs and adjectives and MEN-3K has word pairs with different parts of speech (e.g., a noun compared to a verb). The results we obtain on these datasets exhibit the reliability of our approach in modeling non-nominal word senses.

3.2.2 Discussion

The similarity scale of the SimLex-999 dataset is different from our other word similarity benchmarks in that it assigns relatively low scores to antonymous pairs. For instance, *sunset-sunrise* and *man-woman* in this dataset are assigned the respective similarities of 2.47 and 3.33 (in a $[0, 10]$ similarity scale) which is in the same range as the similarity between word pairs with slight domain relatedness, such as *head-nail* (2.47), *air-molecule* (3.05), or *succeed-try* (3.98). In fact, we observed that tweaking the similarity scale of our system in a way that it diminishes the similarity scores between antonyms can result in a significant performance improvement on this dataset. To this end, we performed an experiment in which the similarity of a word pair was simply divided by 3 whenever the two words belonged to synsets that were linked by an antonymy relation in WordNet.[5] We observed that the performance on the SimLex-999 dataset increased to 61.6 (from 54.2) and 59.1 (from 51.7) according to Pearson ($r \times 100$) and Spearman ($\rho \times 100$) correlation scores, respectively.

[5] We chose 3 so as to transform a pair with high similarity score (around 9.0) to one with slight semantic similarity (around 3.0) in the $[0, 10]$ similarity scale of SimLex-999. We also tested for other values in $[2, 6]$ an observed similar performance gains.

Dataset	Approach	Sense-based score		Word-based score	
		r	ρ	r	ρ
MEN-3K	Iacobacci et al. (2015)	–	**80.5**	–	66.5
	DECONF	**78.0**	78.6	72.3	73.2
	Faruqui et al. (2015)	–	75.9	–	73.7
	Pilehvar and Navigli (2015)	61.7	66.6	–	–
RG-65	DECONF	**90.5**	**89.6**	77.2	76.1
	Iacobacci et al. (2015)	–	87.1	–	73.2
	Faruqui et al. (2015)	–	84.2	–	76.7
	Pilehvar and Navigli (2015)	80.2	84.3	–	–
YP-130	DECONF	**81.6**	**75.2**	58.0	55.9
	Pilehvar and Navigli (2015)	79.0	71.0	–	–
	Iacobacci et al. (2015)	–	63.9	–	34.3
SimLex-999	DECONF	**54.2**	**51.7**	45.4	44.2
	Pilehvar and Navigli (2015)	43.4	43.6	–	–

Table 3: Pearson ($r \times 100$) and Spearman ($\rho \times 100$) correlation scores on four standard word similarity benchmarks. For each benchmark, we show the results reported by any of the comparison systems along with the scores for their corresponding initial word representations (word-based).

Approach	Score	
	AvgSim	AvgSimC
DECONF	**70.8**	**71.5**
Rothe and Schütze (2015) (best)	68.9	69.8
Neelakantan et al. (2014) (best)	67.3	69.3
Chen et al. (2014)	66.2	68.9
Liu et al. (2015) (best)	–	68.1
Huang et al. (2012)	62.8	65.7
Tian et al. (2014) (best)	–	65.7
Iacobacci et al. (2015)	62.4	–
Jauhar et al. (2015)	–	58.7
Initial word vectors	65.1	

Table 4: Spearman correlation scores ($\rho \times 100$) on the Stanford Contextual Word Similarity (SCWS) dataset. We report the AvgSim and AvgSimC scores (cf. §3.2) for each system, where available.

3.3 Cross-Level semantic similarity

In addition to the word similarity benchmark, we evaluated the performance of our representations in the cross-level semantic similarity measurement framework. For this, we opted for the SemEval-2014 task on Cross-Level Semantic Similarity (Jurgens et al., 2014, CLSS). The word to sense similarity subtask of this task, with 500 instances in its test set, provides a suitable benchmark for the evaluation of sense representation techniques.

For a word sense s and a word w, we compute the similarity score according to four different strategies: the similarity of s to the most similar sense of w (**MaxSim**), the average similarity of s to individual senses of w (**AvgSim**), the direct similarity of s to w when the latter is modeled as its word representation (Sense-to-Word or **S2W**) or as the centroid of its senses' representations (Sense to aggregated word senses or **S2A**). For this task, we can only compare against the publicly-available sense representations of Iacobacci et al. (2015), Rothe and Schütze (2015), Pilehvar and Navigli (2015) and Chen et al. (2014) which are linked to the WordNet sense inventory.

3.3.1 Experimental results

Table 5 shows the results on the word to sense dataset of the SemEval-2014 CLSS task, according to Pearson ($r \times 100$) and Spearman ($\rho \times 100$) correlation scores and for the four strategies. As can be seen from the low overall performance, the task is a very challenging benchmark with many WordNet out-of-vocabulary or slang terms and rare usages. Despite this, DECONF provides consistent improvement over the comparison sense representation techniques according to both measures and for all the strategies.

Across the four strategies, S2A proves to be the most effective for DECONF and the representations of Rothe and Schütze (2015). The representations of Chen et al. (2014) perform best with the S2W strat-

System	MaxSim		AvgSim		S2W		S2A	
	r	ρ	r	ρ	r	ρ	r	ρ
DECONF*	**36.4**	**37.6**	**36.8**	**38.8**	**34.9**	**35.6**	**37.5**	**39.3**
Rothe and Schütze (2015)*	34.0	33.8	34.1	33.6	33.4	32.0	35.4	34.9
Iacobacci et al. (2015)*	19.1	21.5	21.3	24.2	22.7	21.7	19.5	21.1
Chen et al. (2014)*	17.7	18.0	17.2	16.8	27.7	26.7	17.9	18.8
DECONF	35.5	36.4	36.2	38.0	34.9	35.6	36.8	38.4
Pilehvar and Navigli (2015)	19.4	23.8	21.2	26.0	–	–	–	–
Iacobacci et al. (2015)	19.0	21.5	20.9	23.2	22.3	20.6	19.2	20.4

Table 5: Evaluation results on the word to sense similarity test dataset of the SemEval-14 task on Cross-Level Semantic Similarity, according to Pearson ($r \times 100$) and Spearman ($\rho \times 100$) correlations. We show results for four similarity computation strategies (see §3.3). The best results per strategy are shown in bold whereas they are underlined for the best strategies per system. Systems marked with $*$ are evaluated on a slightly smaller dataset (474 of the original 500 pairs) so as to have a fair comparison with Rothe and Schütze (2015) and Chen et al. (2014) that use older versions of WordNet (1.7.1 and 1.7, respectively).

egy whereas those of Iacobacci et al. (2015) do not show a consistent trend with relatively low performance across the four strategies. Also, a comparison of our results across the S2W and S2A strategies reveals that a word's aggregated representation, i.e., the centroid of the representations of its senses, is more accurate than its original word representation.

Our analysis showed that the performance of the approaches of Rothe and Schütze (2015) and Iacobacci et al. (2015) were hampered partly due to their limited coverage. In fact, the former was unable to model around 35% of the synsets in WordNet 1.7.1, mainly for its shallow exploitation of knowledge from WordNet, whereas the latter approach did not cover around 15% of synsets in WordNet 3.0. Chen et al. (2014) provide near-full coverage for word senses in WordNet. However, the relatively low performance of their system shows that the usage of glosses in WordNet and the automated disambiguation have not resulted in accurate sense representations. Thanks to its deep exploitation of the underlying resource, our approach provides more reliable representations and full coverage for all word senses and synsets in WordNet.

The three best-performing systems in the task are Meerkat_Mafia (Kashyap et al., 2014) ($r = 37.5$, $\rho = 39.3$), SimCompass (Banea et al., 2014) ($r = 35.4$, $\rho = 34.9$), and SemantiKLUE (Proisl et al., 2014) ($r = 17.9$, $\rho = 18.8$). Note that these systems are specifically designed for the cross-level similarity measurement task. For instance, the best-ranking system in the task leverages a compilation of several dictionaries, including The American Heritage Dictionary, Wiktionary and WordNet, in order to handle slang terms and rare usages, which leads to its competitive performance (Kashyap et al., 2014).

4 Related Work

Learning semantic representations for individual senses of words has been an active area of research for the past few years. Based on the way they view the problem, the recent techniques can be classified into two main branches: (1) those that, similarly to our work, extract knowledge from external sense inventories for learning sense representations; and (2) those techniques that cluster the contexts in which a word appears in a given text corpus and learn distinct representations for individual clusters.

Examples for the first branch include the approaches of Chen et al. (2014), Jauhar et al. (2015) and Rothe and Schütze (2015), all of which use WordNet as an external resource and obtain sense representations for this sense inventory. Chen et al. (2014) uses the content words in the definition of a word sense and WSD. However, the sole usage of glosses as sense-distinguishing contexts and the non-optimal WSD make the approach inaccurate, particularly for highly polysemous words with similar senses and for word senses with short definitions. Similarly, Rothe and Schütze (2015) use only polysemy and synonymy properties of words in WordNet along with a small set of semantic re-

lations. This significantly hampers the reliability of the technique in providing high coverage (discussed further in §3.3.1). Our approach improves over these works by exploiting deep knowledge from the semantic network of WordNet, coupled with an effective training approach. ADW (Pilehvar and Navigli, 2015) is another WordNet-based approach which exploits only the semantic network of this resource and obtains interpretable sense representations. Other work in this branch include SensEmbed (Iacobacci et al., 2015) and Nasari (Camacho-Collados et al., 2015; Camacho-Collados et al., 2016) which are based on the BabelNet sense inventory (Navigli and Ponzetto, 2012). The former technique first disambiguates words in a given corpus with the help of a knowledge-based WSD system and then uses the generated sense-annotated corpus as training data for Word2vec. Nasari combines structural knowledge from the semantic network of BabelNet with corpus statistics derived from Wikipedia for representing BabelNet synsets. However, the approach falls short of modeling non-nominal senses as Wikipedia, due to its very encyclopedic nature, does not cover verbs, adjectives, or adverbs.

The second branch, which is usually referred to as *multi-prototype* representation, is often associated with clustering. Reisinger and Mooney (2010) proposed one of the recent pioneering techniques in this branch. Other prominent work in the category include topical word embeddings (Liu et al., 2015) which use latent topic models for assigning topics to each word in a corpus and learn topic-specific word representations, and the technique proposed by Huang et al. (2012) which incorporates "global document context." Tian et al. (2014) modified the Skip-gram model in order to learn multiple embeddings for each word type. Despite the fact that these techniques do not usually take advantage of the knowledge encoded in structured knowledge resource, they generally suffer from two disadvantages. The first limitation is that they usually make an assumption that a given word has a fixed number of senses, ignoring the fact that polysemy is highly dynamic across words that can range from monosemous to highly ambiguous with dozens of associated meanings (McCarthy et al., 2016). Neelakantan et al. (2014) tackled this issue by estimating the number of senses for a word type during the learning process. However, all techniques in the second branch suffer from another disadvantage that their computed sense representations are not linked to any sense inventory, a linking which itself would require the existence of high coverage sense-annotated data (Agirre et al., 2006).

Another notable line of research incorporates knowledge from external resources, such as PPDB (Ganitkevitch et al., 2013) and WordNet, to improve word embeddings (Yu and Dredze, 2014; Faruqui et al., 2015). Neither of the two techniques however, provide representations for word senses.

5 Conclusions

We put forward a sense representation technique, namely DECONF, that provides multiple advantages in comparison to the recent state of the art: (1) the number of word senses in our technique is flexible and the computed representations are linked to word senses in WordNet; (2) DECONF is effective in providing accurate representation of word senses, even for those senses that do not usually appear frequently in generic text corpora; and (3) our approach is general in that it can be readily applied to any set of word representations and any semantic network without the need for extensive parameter tuning. Our experimental results showed that DECONF can outperform recent state of the art on several datasets across two tasks. The computed representations for word senses in WordNet 3.0 are released at `https://pilehvar.github.io/deconf/`. We intend to apply our technique to the task of harmonizing biomedical terms in the PheneBank project. As future work, we plan to investigate the possibility of using larger semantic networks, such as FreeBase and BabelNet, which would also allow us to apply the technique to languages other than English. We also plan to evaluate the performance of our approach with other decay functions as well as with other initial word representations.

Acknowledgments

The authors gratefully acknowledge the support of the MRC grant No. MR/M025160/1 for PheneBank.

References

[Agirre et al.2006] Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 89–96.

[Agirre et al.2014] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based Word Sense Disambiguation. *Computational Linguistics*, 40(1):57–84.

[Banea et al.2014] Carmen Banea, Di Chen, Rada Mihalcea, Claire Cardie, and Janyce Wiebe. 2014. Simcompass: Using deep learning word embeddings to assess cross-level similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 560–565, Dublin, Ireland.

[Baroni and Lenci2010] Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

[Bengio et al.2007] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2007. *Semi-Supervised Learning*. MIT Press. chapter Label Propagation and Quadratic Criterion.

[Bollacker et al.2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, Vancouver, Canada.

[Bruni et al.2014] Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(1):1–47.

[Camacho-Collados et al.2015] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a Novel Approach to a Semantically-Aware Representation of Items. In *Proceedings of NAACL*, pages 567–577, Denver, USA.

[Camacho-Collados et al.2016] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. NASARI: Integrating explicit knowledge and corpus statistics for amultilingual representation of concepts and entities. *Artificial Intelligence*, 240:36 – 64.

[Chen et al.2014] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP 2014*, pages 1025–1035, Doha, Qatar.

[Das and Smith2011] Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444, Portland, Oregon, USA.

[Faruqui et al.2015] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado.

[Fellbaum1998] Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia.

[Haveliwala2002] Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web*, pages 517–526, Honolulu, Hawaii, USA.

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

[Huang et al.2012] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

[Iacobacci et al.2015] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, Beijing, China.

[Jauhar et al.2015] Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 683–693, Denver, Colorado.

[Jurgens et al.2014] David Jurgens, Mohammad Taher Pilehvar, and Roberto Navigli. 2014. SemEval-2014 task 3: Cross-level semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 17–26, Dublin, Ireland.

[Kashyap et al.2014] Abhay L. Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya W. Satyapanich, Sunil R Gandhi, and Tim Finin. 2014. Meerkat Mafia: Multilingual and Cross-Level Semantic Textual Similarity systems. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 416–423.

[LeCun et al.2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

[Liu et al.2015] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

[McCarthy et al.2016] Diana McCarthy, Marianna Apidianaki, and Katrin Erk. 2016. Word sense clustering and clusterability. *Computational Linguistics*, in press.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations*, Scottsdale, Arizona.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Neelakantan et al.2014] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP 2014*, pages 1059–1069, Doha, Qatar.

[Niemann and Gurevych2011] Elisabeth Niemann and Iryna Gurevych. 2011. The people's Web meets linguistic knowledge: Automatic sense alignment of Wikipedia and WordNet. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 205–214, Oxford, United Kingdom.

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

[Pilehvar and Navigli2015] Mohammad Taher Pilehvar and Roberto Navigli. 2015. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128.

[Proisl et al.2014] Thomas Proisl, Stefan Evert, Paul Greiner, and Besim Kabashi. 2014. SemantiKLUE: Robust semantic similarity at multiple levels using maximum weight matching. In *Proceedings of SemEval-2014*, pages 532–540, Dublin, Ireland.

[Reisinger and Mooney2010] Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, Los Angeles, California.

[Rothe and Schütze2015] Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*, pages 1793–1803, Beijing, China.

[Rubenstein and Goodenough1965] Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

[Sanderson and Van Rijsbergen1999] Mark Sanderson and C. J. Van Rijsbergen. 1999. The impact on retrieval effectiveness of skewed frequency distributions. *ACM Transactions of Infmation Systems*, 17(4):440–465.

[Tian et al.2014] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160, Dublin, Ireland.

[Turney and Pantel2010] Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

[van der Maaten and Hinton2008] L.J.P van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9: 25792605.

[Yang and Powers2005] Dongqiang Yang and David M. W. Powers. 2005. Measuring semantic similarity in the taxonomy of WordNet. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science*, volume 38, pages 315–322, Newcastle, Australia.

[Yeh et al.2009] Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, and Aitor Soroa. 2009. WikiWalk: Random walks on Wikipedia for semantic relatedness. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 41–49, Suntec, Singapore.

[Yu and Dredze2014] Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550, Baltimore, Maryland.

Improving Sparse Word Representations with Distributional Inference for Semantic Composition

Thomas Kober, Julie Weeds, Jeremy Reffin and **David Weir**
TAG laboratory, Department of Informatics, University of Sussex
Brighton, BN1 9RH, UK
{t.kober, j.e.weeds, j.p.reffin, d.j.weir}@sussex.ac.uk

Abstract

Distributional models are derived from co-occurrences in a corpus, where only a small proportion of all possible plausible co-occurrences will be observed. This results in a very sparse vector space, requiring a mechanism for inferring missing knowledge. Most methods face this challenge in ways that render the resulting word representations uninterpretable, with the consequence that semantic composition becomes hard to model. In this paper we explore an alternative which involves explicitly inferring unobserved co-occurrences using the distributional neighbourhood. We show that distributional inference improves sparse word representations on several word similarity benchmarks and demonstrate that our model is competitive with the state-of-the-art for adjective-noun, noun-noun and verb-object compositions while being fully interpretable.

1 Introduction

The aim of distributional semantics is to derive meaning representations based on observing co-occurrences of words in large text corpora. However not all plausible co-occurrences will be observed in any given corpus, resulting in word representations that only capture a fragment of the meaning of a word. For example the verbs "walking" and "strolling" may occur in many different and possibly disjoint contexts, although both verbs would be equally plausible in numerous cases. This subsequently results in incomplete representations for both lexemes. In addition, models based on counting co-occurrences face the general problem of sparsity in a very high-dimensional vector space. The most common approaches to these challenges have involved the use of various techniques for dimensionality reduction (Bullinaria and Levy, 2012; Lapesa and Evert, 2014) or the use of low-dimensional and dense neural word embeddings (Mikolov et al., 2013; Pennington et al., 2014). The common problem in both of these approaches is that composition becomes a black-box process due to the lack of interpretability of the representations. Count-based models are therefore a very attractive line of work with regards to a number of important long-term research challenges, most notably the development of an adequate model of distributional compositional semantics. In this paper we propose the use of distributional inference (DI) to inject unobserved but plausible distributional semantic knowledge into the vector space by leveraging the intrinsic structure of the distributional neighbourhood. This results in richer word representations and furthermore mitigates the sparsity effect common in high-dimensional vector spaces, while remaining fully interpretable.

Our contributions are as follows: we show that typed and untyped sparse word representations, enriched by distributional inference, lead to performance improvements on several word similarity benchmarks, and that a higher-order dependency-typed vector space model, based on "Anchored Packed Dependency Trees (APTs)" (Weir et al., 2016), is competitive with the state-of-the-art for adjective-noun, noun-noun and verb-object compositions. Using our method, we are able to bridge the gap in performance between high dimensional interpretable mod-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1691–1702,
Austin, Texas, November 1-5, 2016.

els and low dimensional non-interpretable models and offer evidence to support a possible explanation of why high-dimensional models usually perform worse, together with a simple, practical method for over-coming this problem. We furthermore demonstrate that *intersective* approaches to composition benefit more from distributional inference than composition by *union* and highlight the ability of composition by *intersection* to disambiguate the meaning of a phrase in a local context.

The remainder of this paper is structured as follows: we discuss related work in section 2, followed by an introduction of the APT framework for semantic composition in section 3. We describe distributional inference in section 4 and present our experimental work, together with our results in section 5. We conclude this paper and outline future work in section 6.

2 Related Work

Our method follows the distributional smoothing approach of Dagan et al. (1994) and Dagan et al. (1997). In these works the authors are concerned with smoothing the probability estimate for unseen words in bigrams. This is achieved by measuring which unobserved bigrams are more likely than others on the basis of the Kullback-Leibler divergence between bigram distributions. This has led to significantly improved performance on a language modelling for speech recognition task, as well as for word-sense disambiguation in machine translation (Dagan et al., 1994; Dagan et al., 1997). More recently Padó et al. (2013) used a distributional approach for smoothing derivationally related words, such as *oldish – old*, as a back-off strategy in case of data sparsity. However, none of these approaches have used distributional inference as a general technique for directly enriching sparse distributional vector representations, or have explored its behaviour for semantic composition.

Compositional models of distributional semantics have become an increasingly popular topic in the research community. Starting from simple pointwise additive and multiplicative approaches to composition, such as Mitchell and Lapata (2008; 2010), and Blacoe and Lapata (2012), to tensor based models, such as Baroni and Zamparelli (2010), Coecke et al. (2010), Grefenstette et al. (2013) and Paperno et al. (2014), and neural network based approaches, such as Socher et al. (2012), Le and Zuidema (2015), Mou et al. (2015) and Tai et al. (2015). Zanzotto et al. (2015) provide a decompositional analysis of how similarity is affected by distributional composition, and link compositional models to convolution kernels. Most closely related to our approach of composition are the works of Thater et al. (2010), Thater et al. (2011) and Weeds et al. (2014), which aim to provide a general model of compositionality in a typed distributional vector space. In this paper we adopt the approach to distributional composition introduced by Weir et al. (2016), whose APT framework is based on a higher-order dependency-typed vector space, however they do not address the issue of sparsity in their work.

3 Background

Distributional vector space models can broadly be categorised into untyped proximity based models and typed models (Baroni and Lenci, 2010). Examples of the former include Deerwester et al. (1990); Lund and Burgess (1996); Curran (2004); Sahlgren (2006); Bullinaria and Levy (2007) and Turney and Pantel (2010). These models count the number of times every word in a large corpus co-occurs with other words within a specified spatial context window, without leveraging the structural information of the text. Typed models on the other hand, take the grammatical relation between two words for a co-occurrence event into account. Early proponents of that approach are Grefenstette (1994) and Lin (1998). More recent work by Padó and Lapata (2007), Erk and Padó (2008) and Weir et al. (2016) uses dependency paths to build a structured vector space model. In both kinds of models, the raw counts are usually transformed by Positive Pointwise Mutual Information (PPMI) or a variant of it (Church and Hanks, 1990; Niwa and Nitta, 1994; Scheible et al., 2013; Levy and Goldberg, 2014).

In the following we will give an explanation of the theory of composition with APTs as introduced by Weir et al. (2016), which we adopt in this paper. In addition to direct relations between two words, the APT model also considers *inverse* and *higher*

order relations. Inverse relations are denoted with a horizontal bar above the dependency relation, such as $\overline{\texttt{amod}}$ for an inverse adjectival modifier. Higher order dependencies are separated by a colon as in the second order distributional feature $\overline{\texttt{dobj}}\texttt{:nsubj}$. The example below illustrates how raw text is processed to retrieve elementary representations in our APT model. As an example we consider a lowercased corpus consisting of the sentences:

we folded the clean clothes
i like your clothes
we bought white shoes yesterday
he folded the white sheets

We dependency parse the raw sentences and, following Weir et al. (2016), align and aggregate the resulting parse trees according to their dependency type as shown in Figure 1. For example the lexeme *clothes* has the distributional features $\texttt{amod:}dry$ and $\overline{\texttt{dobj}}\texttt{:nsubj:}we$ among others. Over a large corpus, this results in a very high-dimensional and sparse vector space, which due to its typed nature is much sparser than for untyped models.

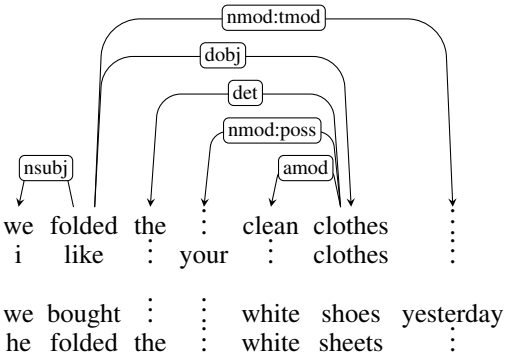


Figure 1: Aligned Packed Dependency Tree representation of the example sentences.

Composition with APTs

Composition is linguistically motivated by the principle of compositionality, which states that the meaning of a complex expression is fully determined by its structure and the meanings of its constituents (Frege, 1884). Many simple approaches to semantic composition neglect the structure and lose information in the composition process. For example, the phrases *house boat* and *boat house* have the exact same representation when composition is done via a pointwise arithmetic operation. Despite performing well in a number of studies, this commutativity is not desirable for a fine grained understanding of the semantics of natural language. When performing composition with APTs, we adopt the method introduced by Weir et al. (2016) which views distributional composition as a process of contextualisation. For composing the adjective *white* with the noun *clothes* via the dependency relation $\texttt{amod}$ we need to consider how the adjective interacts with the noun in the vector space. The distributional features of white describe things that are white via their first order relations such as $\overline{\texttt{amod}}$, and things that can be done to white things, such as *bought* via $\overline{\texttt{amod}}\texttt{:}\overline{\texttt{dobj}}$ in the example above.

Table 1 shows a number of features extracted from the aligned dependency trees in Figure 1 and highlights that adjectives and nouns do not share many features if only first order dependencies would be considered. However through the inclusion of inverse and higher order dependency paths we can observe that the second order features of the adjective align with the first order features of the noun. For composition, the adjective *white* needs to be offset by its *inverse* relation to *clothes*[1] making it distributionally similar to a noun that has been modified by *white*. Offsetting can be seen as shifting the current viewpoint in the APT data structure and is necessary for aligning the feature spaces for composition (Weir et al., 2016). We are then in a position to compose the offset representation of *white* with the vector for *clothes* by the *union* or the *intersection* of their features.

Table 2 shows the resulting feature spaces of the composed vectors. It is worth noting that any arithmetic operation can be used to combine the counts of the aligned features, however for this paper we use pointwise addition for both composition functions. One of the advantages of this approach to composition is that the inherent interpretability of count-based models naturally expands beyond the word level, allowing us to study the distributional semantics of phrases in the same space as words. Due to offsetting one of the constituents, the composition operation is not commutative and hence avoids identical representations for *house boat* and *boat house*. However, the typed nature of our vector space re-

[1] The inverse of $\overline{\texttt{amod}}$ is just $\texttt{amod}$.

white			*clothes*	
Distributional Features	**Offset Features (by amod)**	**Co-occurrence Count**	**Distributional Features**	**Co-occurrence Count**
$\overline{\texttt{amod}}$:*shoes*	:*shoes*	1	amod:*clean*	1
$\overline{\texttt{amod}}$:$\overline{\texttt{dobj}}$:*bought*	$\overline{\texttt{dobj}}$:*bought*	1	$\overline{\texttt{dobj}}$:*like*	1
$\overline{\texttt{amod}}$:$\overline{\texttt{dobj}}$:*folded*	$\overline{\texttt{dobj}}$:*folded*	1	$\overline{\texttt{dobj}}$:*folded*	1
$\overline{\texttt{amod}}$:$\overline{\texttt{dobj}}$:nsubj:*we*	$\overline{\texttt{dobj}}$:nsubj:*we*	1	$\overline{\texttt{dobj}}$:nsubj:*we*	1

Table 1: Example feature spaces for the lexemes *white* and *clothes* extracted from the dependency tree of Figure 1. Not all features are displayed for space reasons. Offsetting $\overline{\texttt{amod}}$:*shoes* by amod results in an empty dependency path, leaving just the word co-occurrence :*shoes* as feature.

Composition by *union*		Composition by *intersection*	
Distributional Features	**Co-occurrence Count**	**Distributional Features**	**Co-occurrence Count**
:*shoes*	1		
amod:*clean*	1		
$\overline{\texttt{dobj}}$:*bought*	1		
$\overline{\texttt{dobj}}$:*folded*	2	$\overline{\texttt{dobj}}$:*folded*	2
$\overline{\texttt{dobj}}$:*like*	1		
$\overline{\texttt{dobj}}$:nsubj:*we*	2	$\overline{\texttt{dobj}}$:nsubj:*we*	2

Table 2: Comparison of composition by *union* and composition by *intersection*. Not all features are displayed for space reasons.

sults in extreme sparsity, for example while the untyped VSM has 130k dimensions, our APT model can have more than 3m dimensions. We therefore need to enrich the elementary vector representations with the distributional information of their nearest neighbours to ease the sparsity effect and infer missing information. Due to the syntactic nature of our composition operation it is not straightforward to apply common dimensionality reduction techniques such as SVD, as the type information needs to be preserved.

4 Distributional Inference

Following Dagan et al. (1994) and Dagan et al. (1997), we propose a simple unsupervised algorithm for enriching sparse vector representations with their nearest neighbours. We show that our distributional inference algorithm improves performance for untyped and typed models on several word similarity benchmarks, as well as being competitive with the state-of-the-art on semantic composition. As shown in algorithm 1 below, we iterate over all word vectors w in a given distributional model M, and add the vector representations of the nearest neighbours n, determined by cosine similarity, to the representation of the enriched word vector w'. The parameter α in line 4 scales the contribution of the original word vector to the resulting enriched representation. In this work we always chose α to be identical to the number of neighbours used for distributional inference. For example, if we used 10 neighbours for DI, we would set $\alpha = 10$, which we found sufficient to prevent the neighbours from dominating the vector representation. In our experiments we kept the input distributional model fixed, however it is equally possible to update the given model in an online fashion, adding some amount of stochasticity to the enriched word vector representations. There is a number of possibilities for the neighbour retrieval function $neighbours()$ and we explore several options in this paper. The algorithm furthermore is agnostic to the input distributional model, for example it is possible to use completely different vector space models for querying neighbours and enrichment.

Algorithm 1 Distributional Inference

```
1:  procedure DIST_INFERENCE(M)
2:      init M'
3:      for all w in M do
4:          w' ← w × α
5:          for all n in neighbours(M, w) do
6:              w' ← w' + n
7:              add w' to M'
8:          end for
9:      end for
10:     return M'
11: end procedure
```

Static Top n Neighbour Retrieval

The perhaps simplest way is to choose the top n most similar neighbours for each word in the vector space and enrich the respective vector representations with them.

Density based Neighbour Retrieval

This approach has its roots in kernel density estimation (Parzen, 1962), however instead of defining a static global parzen window, we set the window size for every word individually, depending on the distance to its nearest neighbour, plus a threshold. For example if the cosine distance between the target vector and its top neighbour is 0.5, we use a window size of $0.5 + \epsilon$ for that word. In our experiments we typically define ϵ to be proportional to the distance of the nearest neighbour (e.g. $\epsilon = 0.5 \times 0.1$).

WordNet based Neighbour Retrieval

Instead of leveraging the intrinsic structure of our distributional vector space, we retrieve neighbours by querying WordNet (Fellbaum, 1998), and treat synsets with agreeing PoS tags as the nearest neighbours of any target vector. This restricts the retrieved neighbours to synonyms only.

5 Experiments

Our model is based on a cleaned October 2013 Wikipedia dump, which excludes all pages with fewer than 20 page views, resulting in a corpus of approximately 0.6 billion tokens (Wilson, 2015). The corpus is lowercased, tokenised, lemmatised, PoS tagged and dependency parsed with the Stanford NLP tools, using universal dependencies (Manning et al., 2014; de Marneffe et al., 2014). We then build our APT model with first, second and third order relations. We remove distributional features with a count of less than 10, and vectors containing fewer than 50 non-zero entries. The raw counts are subsequently transformed to PPMI weights. The untyped vector space model is built from the same lowercased, tokenised and lemmatised Wikipedia corpus. We discard terms with a frequency of less than 50 and apply PPMI to the raw co-occurrence counts.

Shifted PPMI

We explore a range of different values for shifting the PPMI scores as these have a significant impact on the performance of the APT model. The effect of shifting PPMI scores for untyped vector space models has already been explored in Levy and Goldberg (2014), and Levy et al. (2015), thus we only present results for the APT model. As shown in equation 1, PMI is defined as the log of the ratio of the joint probability of observing a word w and a context c together, and the product of the respective marginals of observing them separately. In our APT model, a context c is defined as a dependency relation together with a word.

$$\begin{aligned} PMI(w,c) &= \log \frac{P(w,c)}{P(w)P(c)} \\ SPPMI(w,c) &= \max(PMI(w,c) - \log k, 0) \end{aligned} \tag{1}$$

As PMI is negatively unbounded, PPMI is used to ensure that all values are greater than or equal to 0. Shifted PPMI (SPPMI) subtracts a constant from any PMI score before applying the PPMI threshold. We experiment with values of 1, 5, 10, 40 and 100 for the shift parameter k.

5.1 Word Similarity Experiments

We first evaluate our models on 3 word similarity benchmarks, MEN (Bruni et al., 2014), which is testing for *relatedness* (e.g. meronymy or holonymy) between terms, SimLex-999 (Hill et al., 2015), which is testing for *substitutability* (e.g. synonymy, antonymy, hyponymy and hypernymy), and WordSim-353 (Finkelstein et al., 2001), where we use the version of Agirre et al. (2009), who split the dataset into a *relatedness* and a *substitutability* subset. Baroni and Lenci (2011) have shown that untyped models are typically better at capturing *relatedness*, whereas typed models are better at encoding *substitutability*. Performance is measured by computing Spearman's ρ between the cosine similarities of the vector representations and the corresponding aggregated human similarity judgements. For these experiments we keep the number of neighbours that a word vector can consume fixed at 30. This value is based on preliminary experiments on WordSim-353 (see Figure 2) using the static top n neighbour retrieval function and a PPMI shift of $k = 40$. Figure 2 shows that distributional inference improves performance for any number of neighbours over a model without DI (marked as horizontal dashed lines for each WordSim-353 subset) and peaks at a value of

30. Performance slightly degrades with more neighbours. For the untyped VSM we use a symmetric window of 5 on either side of the target word.

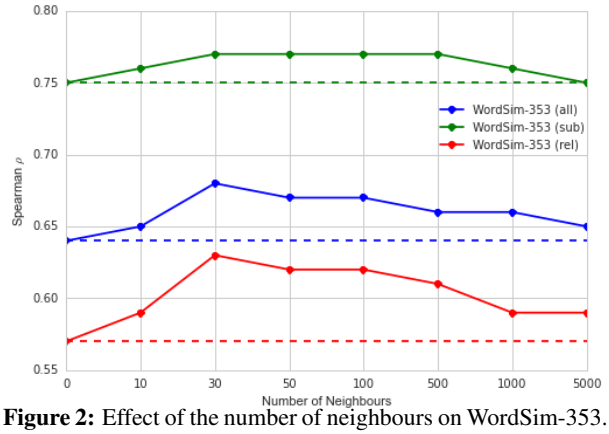


Figure 2: Effect of the number of neighbours on WordSim-353.

Table 3 highlights the effect of the SPPMI shift parameter k, while keeping the number of neighbours fixed at 30 and using the static top n neighbour retrieval function. For the APT model, a value of $k = 40$ performs best (except for SimLex-999, where smaller shifts give better results), with a performance drop-off for larger shifts. In our experiments we find that a shift of $k = 1$ results in top performance for the untyped vector space model. It appears that shifting the PPMI scores in the APT model has the effect of cleaning the vectors from noisy PPMI artefacts, which reinforces the predominant sense, while other senses get suppressed. Subsequently, this results in a cleaner neighbourhood around the word vector, dominated by a single sense. This explains why distributional inference slightly degrades performance for smaller values of k.

Table 4 shows that distributional inference successfully infers missing information for both model types, resulting in improved performance over models without the use of DI on all datasets. The improvements are typically larger for the APT model, suggesting that it is missing more distributional knowledge in its elementary representations than untyped models. The density window and static top n neighbour retrieval functions perform very similar, however the static approach is more consistent and never underperforms the baseline for either model type on any dataset. The WordNet based neighbour retrieval function performs particularly well on SimLex-999. This can be explained by the fact that antonyms, which frequently happen to be among the nearest neighbours in distributional vector spaces, are regarded as dissimilar in SimLex-999, whereas the WordNet neighbour retrieval function only returns synonyms. The results furthermore confirm the effect that untyped models perform better on datasets modelling *relatedness*, whereas typed models work better for *substitutability* tasks (Baroni and Lenci, 2011).

5.2 Composition Experiments

Our approach to semantic composition as described in section 3 requires the dimensions of our vector space models to be meaningful and interpretable. However, the problem of missing information is amplified in compositional settings as many compatible dimensions between words are not observed in the source corpus. It is therefore crucial that distributional inference is able to inject some of the missing information in order to improve the composition process. For the experiments involving semantic composition, we enrich the elementary representations of the phrase constituents before composition.

We first conduct a qualitative analysis for our APT model and observe the effect of distributional inference on the nearest neighbours of composed adjective-noun, noun-noun and verb-object compounds. In these experiments, we show how distributional inference changes the neighbourhood in which composed phrases are embedded, and highlight the difference between composition by *union* and composition by *intersection*. For this experiment we use the static top n neighbour retrieval function with 30 neighbours and $k = 40$.

Table 5 shows a small number of example phrases together with their top 3 nearest neighbours, computed from the union of all words in the Wikipedia corpus and all phrase pairs in the Mitchell and Lapata (2010) dataset. As can be seen, nearest neighbours of phrases can be either single words or other composed phrases. Words or phrases marked with "*" in Table 5 mean that DI introduced, or failed to downrank, a spurious neighbour, while boldface means that performing distributional inference resulted in a neighbourhood more coherent with the query phrase than without DI.

Table 5 shows that composition by *union* is unable to

APTs	MEN		SimLex-999		WordSim-353 (rel)		WordSim-353 (sub)	
	without DI	*with DI*	*without DI*	*with DI*	*without DI*	*with DI*	*without DI*	*with DI*
$k = 1$	0.54	0.52	0.31	0.30	0.34	0.27	0.62	0.60
$k = 5$	0.64	0.65	0.35	**0.36**	0.56	0.51	0.74	0.73
$k = 10$	0.63	0.66	0.35	**0.36**	0.56	0.55	0.75	0.74
$k = 40$	0.63	**0.68**	0.30	0.32	0.55	**0.61**	0.75	**0.76**
$k = 100$	0.61	0.67	0.26	0.29	0.47	0.60	0.71	0.72

Table 3: Effect of the magnitude of the shift parameter k in SPPMI on the word similarity tasks. Boldface means best performance per dateset.

APTs ($k = 40$)	**No Distributional Inference**	**Density Window**	**Static Top** n	**WordNet**
MEN	0.63	0.67	**0.68**	0.63
SimLex-999	0.30	0.32	0.32	**0.38**
WordSim-353 (rel)	0.55	**0.62**	0.61	0.56
WordSim-353 (sub)	0.75	**0.78**	0.76	0.77
Untyped VSM ($k = 1$)	**No Distributional Inference**	**Density Window**	**Static Top** n	**WordNet**
*MEN**	**0.71**	**0.71**	**0.71**	**0.71**
SimLex-999	0.30	0.29	0.30	**0.36**
WordSim-353 (rel)	0.60	**0.64**	**0.64**	0.52
WordSim-353 (sub)	0.70	**0.73**	0.72	0.67

Table 4: Neighbour retrieval function comparison. Boldface means best performance on a dataset *per* VSM type. *) With 3 significant figures, the density window approach (0.713) is slightly better than the baseline without DI (0.708), static top n (0.710) and WordNet (0.710).

downrank unrelated neighbours introduced by distributional inference. For example *large quantity* is incorrectly introduced as a top ranked neighbour for the phrase *small house*, due to the proximity of *small* and *large* in the vector space. The phrases *market leader* and *television programme* are two examples of incoherent neighbours, which the composition function was unable to downrank and where DI could not improve the neighbourhood. Composition by *intersection* on the other hand vastly benefits from distributional inference. Due to the increased sparsity induced by the composition process, a neighbourhood without DI produces numerous spurious neighbours as in the case of the verb *have* as a neighbour for *win battle*. Distributional inference introduces qualitatively better neighbours for almost all phrases. For example, *government leader* and *opposition member* are introduced as top ranked neighbours for the phrase *party leader*, and *stress importance* and *underline* are introduced as new top neighbours for the phrase *emphasise need*. These results show that composition by *union* does not have the ability to disambiguate the meaning of a word in a given phrasal context, whereas composition by *intersection* has that ability but requires distributional inference to unleash its full potential.

For a quantitative analysis of distributional inference for semantic composition, we evaluate our model on the composition dataset of Mitchell and Lapata (2010), consisting of 108 adjective-noun, 108 noun-noun, and 108 verb-object pairs. The task is to compare the model's similarity estimates with the human judgements by computing Spearman's ρ. For comparing the performance of the different neighbour retrieval functions, we choose the same parameter settings as in the word similarity experiments ($k = 40$ and using 30 neighbours for DI).

Table 6 shows that the static top n and density window neighbour retrieval functions perform very similar again. The density window retrieval function outperforms static top n for composition by *intersection* and *vice versa* for composition by *union*. The WordNet approach is competitive for composition by *union*, but underperfoms the other approaches for composition by *intersection* significantly. For further experiments we use the static top n approach as it is computationally cheap and easy to interpret due to the fixed number of neighbours. Table 6 also shows that while composition by *intersection* is significantly improved by distributional

Phrase	Comp.	Union	Union (with DI)	Intersection	Intersection (with DI)
national government	AN	government, regime, ministry	government, regime*, european state*	federal assembly, government, monopoly	federal assembly, government, local office
small house	AN	house, public building, building	house, public building, large quantity*	apartment, cottage, cabin	cottage, apartment, cabin
party leader	NN	leader, market leader, government leader	leader, government leader, market leader*	party official, NDP, leader	**government leader**, party official, **opposition member**
training programme	NN	programme, action programme, television programme	programme, action programme*, television programme*	training college, trainee, education course	training college, education course, **seminar**
win battle	VO	win, win match, ties	win, win match, **fight war**	win match, win, have	**fight war**, **fight**, win match
emphasise need	VO	emphasise, underline, underscore	emphasise, underline, underscore	emphasise, prioritize, negate	emphasise, **stress importance**, **underline**

Table 5: Nearest neighbours AN, NN and VO pairs in the Mitchell and Lapata (2010) dataset, with and without distributional inference. Words and phrases marked with * denote spurious neighbours, boldfaced words and phrases mark improved neighbours.

inference, composition by *union* does not appear to benefit from it.

Composition by *Union* or *Intersection*

Both model types in this study support composition by *union* as well as composition by *intersection*. In untyped models, composition by *union* and composition by *intersection* can be achieved by pointwise addition and pointwise multiplication respectively. The major difference between composition in the APT model and the untyped model is that in the former, composition is not commutative due to offsetting the modifier in a dependency relation (see section 3). Blacoe and Lapata (2012) showed that an intersective composition function such as pointwise multiplication represents a competitive and robust approach in comparison to more sophisticated composition methods. For the final set of experiments on the Mitchell and Lapata (2010) dataset, we present results the APT model and the untyped model, using composition by *union* and composition by *intersection*, with and without distributional inference. We compare our models with the best performing untyped VSMs of Mitchell and Lapata (2010), and Blacoe and Lapata (2012), the best performing APT model of Weir et al. (2016), as well as with the recently published state-of-the-art methods by Hashimoto et al. (2014), and Wieting et al. (2015), who are using neural network based approaches. For our models, we use the static top n approach as neighbour retrieval function and tune the remaining parameters, the SPPMI shift k (1, 5, 10, 40, 100) and the number of neighbours (10, 30, 50, 100, 500, 1000, 5000), for both model types, and the sliding window size for the untyped VSM (1, 2, 5), on the development portion of the Mitchell and Lapata (2010) dataset. We keep the vector configuration (k and window size) fixed for all phrase types and only tune the number of neighbours used for DI individually. The best vector configuration for the APT model is achieved with $k = 10$ and for the untyped VSM with $k = 1$. For composition by *intersection* best performance on the dev set was achieved with 1000 neighbours for ANs, 10 for NNs and 50 for VOs with DI. For composition by *union*, top performance was obtained with 100 neighbours for ANs, 30 neighbours for NNs and 50 for VOs. The best results for the untyped model on the dev set are achieved with a symmetric window size of 1 and using 5000 neighbours for ANs, 10 for NNs and 1000 for VOs with composition by pointwise multiplication, and 30 neighbours for ANs, 5000 for NNs and 5000 for VOs for composition by pointwise addition. The validated numbers of neighbours on the development set show that the problem of missing information appears to be more severe for semantic composition than for word similarity tasks. Even though a neighbour at rank 1000 or lower does not appear to have a close relationship to the target word, it still can contribute useful co-occurrence information not observed in the original vector.

Table 7 shows that composition by *intersection* with distributional inference considerably improves upon the best results for APT models without distributional inference and for untyped count-based models, and is competitive with the state-of-the-art neural network based models of Hashimoto et al. (2014) and Wieting et al. (2015). Distributional inference also improves upon the performance of an untyped VSM where composition by pointwise multiplication is outperforming the models of Mitchell and Lapata (2010), and Blacoe and Lapata (2012). Table 7 furthermore shows that DI has a smaller effect on

APTs	No Distributional Inference		Density Window		Static Top n		WordNet	
	intersection	*union*	*intersection*	*union*	*intersection*	*union*	*intersection*	*union*
Adjective-Noun	0.10	0.41	0.31	0.39	0.25	0.40	0.12	0.41
Noun-Noun	0.18	0.42	0.34	0.38	0.37	0.45	0.24	0.36
Verb-Object	0.17	0.36	0.36	0.36	0.34	0.35	0.25	0.36
Average	0.15	**0.40**	0.34	0.38	0.32	**0.40**	0.20	0.38

Table 6: Neighbour retrieval function. Underlined means best performance per phrase type, boldface means best average performance overall.

Model	Adjective-Noun	Noun-Noun	Verb-Object	Average
APT – *union*	0.45 (0.45)	0.45 (0.43)	0.38 (0.37)	0.43 (0.42)
APT – *intersect*	0.50 (0.38)	0.49 (0.44)	0.43 (0.36)	0.47 (0.39)
Untyped VSM – *addition*	0.46 (0.46)	0.40 (0.41)	0.38 (0.33)	0.41 (0.40)
Untyped VSM – *multiplication*	0.46 (0.42)	0.48 (0.45)	0.40 (0.39)	0.45 (0.42)
Mitchell and Lapata (2010) (untyped VSM & *multiplication*)	0.46	0.49	0.37	0.44
Blacoe and Lapata (2012) (untyped VSM & *multiplication*)	0.48	**0.50**	0.35	0.44
Hashimoto et al. (2014) (PAS-CLBLM & Add_{nl})	**0.52**	0.46	0.45	**0.48**
Wieting et al. (2015) (Paragram word embeddings & *RNN*)	0.51	0.40	**0.50**	0.47
Weir et al. (2016) (APT & *union*)	0.45	0.42	0.42	0.43

Table 7: Results for the Mitchell and Lapata (2010) dataset. Results in brackets denote the performance of the respective models without the use of distributional inference. Underlined means best within group, boldfaced means best overall.

the APT model based on composition by *union* and the untyped model based on composition by pointwise addition. The reason, as pointed out in the discussion for Table 5, is that the composition function has no disambiguating effect and thus cannot eliminate unrelated neighbours introduced by distributional inference. An intersective composition function on the other hand is able to perform the disambiguation locally in any given phrasal context. This furthermore suggests that for the APT model it is not necessary to explicitly model different word senses in separate vectors, as composition by *intersection* is able to disambiguate any word in context individually. Unlike the models of Hashimoto et al. (2014) and Wieting et al. (2015), the elementary word representations, as well as the representations for composed phrases and the composition process in our models are fully interpretable[2].

6 Conclusion and Future Work

One of the major challenges in count-based models is dealing with sparsity and missing information. To address this challenge, we contribute an unsupervised algorithm for enriching sparse word representations by exploiting the distributional neighbourhood. We have demonstrated its benefit to typed and untyped vector space models on a range of tasks and have shown that with distributional inference our APT model is competitive with the state-of-the-art for adjective-noun, noun-noun and verb-object compositions while being fully interpretable. With our method, we are able to bridge the gap in performance between low-dimensional non-interpretable and high-dimensional interpretable representations. Lastly, we have investigated the different behaviour of composition by *union* and composition by *intersection* and have shown that an *intersective* composition function, together with distributional inference, has the ability to locally disambiguate the meaning of a phrase.

In future work we aim to scale our approach to semantic composition with distributional inference to longer phrases and full sentences. We furthermore plan to investigate whether the number of neighbours required for improving elementary vector representations remains as high for other compositional tasks and longer phrases as in this study.

Acknowledgments

This work was funded by UK EPSRC project EP/IO37458/1 *"A Unified Model of Compositional and Distributional Compositional Semantics: Theory and Applications"*. We would like to thank our anonymous reviewers for their helpful comments.

[2]We release the APT vectors and our code at `https://github.com/tttthomasssss/apt-toolkit`.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL-HLT*, pages 19–27, Boulder, Colorado, June. Association for Computational Linguistics.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, December.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of GEMS Workshop*, GEMS '11, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Cambridge, MA, October. Association for Computational Linguistics.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea, July. Association for Computational Linguistics.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47, January.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, pages 510–526.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior Research Methods*, 44(3):890–907.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, March.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Ido Dagan, Fernando Pereira, and Lillian Lee. 1994. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of ACL*, pages 272–278, Las Cruces, New Mexico, USA, June. Association for Computational Linguistics.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1997. Similarity-based methods for word sense disambiguation. In *Proceedings of ACL*, pages 56–63, Madrid, Spain, July. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592, Reykjavik, Iceland, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1045.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *J. Amer. Soc. Inf. Sci.*, 41(6):391–407.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, Hawaii, October. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of WWW*, WWW '01, pages 406–414, New York, NY, USA. ACM.

Gottlob Frege. 1884. *Die Grundlagen der Arithmetik: Eine logisch mathematische Untersuchung über den Begriff der Zahl*. W. Koebner.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. *Proceedings of IWCS*.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of EMNLP*, pages 1544–1555, Doha, Qatar, October. Association for Computational Linguistics.

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

Gabriella Lapesa and Stefan Evert. 2014. A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *TACL*, 2:531–545.

Phong Le and Willem Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of EMNLP*, pages 1155–1164, Lisbon, Portugal, September. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, pages 2177–2185.

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

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of ACL*, pages 768–774, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL - System Demonstrations*, pages 55–60.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *In Proceedings of ACL-08: HLT*, pages 236–244.

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

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of EMNLP*, pages 2315–2325, Lisbon, Portugal, September. Association for Computational Linguistics.

Yoshiki Niwa and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of Coling*, COLING '94, pages 304–309, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Sebastian Padó, Jan Šnajder, and Britta Zeller. 2013. Derivational smoothing for syntactic distributional semantics. In *Proceedings of ACL*, pages 731–735, Sofia, Bulgaria, August. Association for Computational Linguistics.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*, pages 90–99, Baltimore, Maryland, June. Association for Computational Linguistics.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09.

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

Magnus Sahlgren. 2006. *The Word-space model*. Ph.D. thesis, University of Stockholm (Sweden).

Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of IJCNLP*, pages 489–497, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL*, pages 948–957, Uppsala, Sweden, July. Association for Computational Linguistics.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of IJCNLP*, pages 1134–1143, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

Julie Weeds, David Weir, and Jeremy Reffin. 2014. Distributional composition using higher-order dependency vectors. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, pages 11–20, Gothenburg, Sweden, April. Association for Computational Linguistics.

David Weir, Julie Weeds, Jeremy Reffin, and Thomas Kober. 2016. Aligning packed dependency trees:

a theory of composition for distributional semantics. *Computational Linguistics*, in press (http://arxiv.org/abs/1608.07115).

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

Benjamin Wilson. 2015. The unknown perils of mining wikipedia. https://blog.lateral.io/2015/06/the-unknown-perils-of-mining-wikipedia/, June.

Fabio Massimo Zanzotto, Lorenzo Ferrone, and Marco Baroni. 2015. Squibs: When the whole is not greater than the combination of its parts: A "decompositional" look at compositional distributional semantics. *Computational Linguistics*, 41(1):165–173.

Modelling Interaction of Sentence Pair with Coupled-LSTMs

Pengfei Liu Xipeng Qiu* Yaqian Zhou Jifan Chen Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,zhouyaqian,jfchen14, xjhuang}@fudan.edu.cn

Abstract

Recently, there is rising interest in modelling the interactions of two sentences with deep neural networks. However, most of the existing methods encode two sequences with separate encoders, in which a sentence is encoded with little or no information from the other sentence. In this paper, we propose a deep architecture to model the strong interaction of sentence pair with two coupled-LSTMs. Specifically, we introduce two coupled ways to model the interdependences of two LSTMs, coupling the local contextualized interactions of two sentences. We then aggregate these interactions and use a dynamic pooling to select the most informative features. Experiments on two very large datasets demonstrate the efficacy of our proposed architectures.

1 Introduction

Distributed representations of words or sentences have been widely used in many natural language processing (NLP) tasks, such as text classification (Kalchbrenner et al., 2014; Liu et al., 2015), question answering and machine translation (Sutskever et al., 2014) and so on. Among these tasks, a common problem is modelling the relevance/similarity of the sentence pair, which is also called text semantic matching.

Recently, deep learning based models is rising a substantial interest in text semantic matching and have achieved some great progresses (Hu et al., 2014; Qiu and Huang, 2015; Wan et al., 2016).

*Corresponding author.

According to the phases of interaction between two sentences, previous models can be classified into three categories.

Weak interaction Models Some early works focus on sentence level interactions, such as ARC-I(Hu et al., 2014), CNTN(Qiu and Huang, 2015) and so on. These models first encode two sequences with some basic (Neural Bag-of-words, BOW) or advanced (RNN, CNN) components of neural networks separately, and then compute the matching score based on the distributed vectors of two sentences. In this paradigm, two sentences have no interaction until arriving final phase.

Semi-interaction Models Some improved methods focus on utilizing multi-granularity representation (word, phrase and sentence level), such as MultiGranCNN (Yin and Schütze, 2015) and Multi-Perspective CNN (He et al., 2015). Another kind of models use soft attention mechanism to obtain the representation of one sentence by depending on representation of another sentence, such as ABCNN (Yin et al., 2015), Attention LSTM(Rocktäschel et al., 2015; Hermann et al., 2015). These models can alleviate the weak interaction problem, but are still insufficient to model the contextualized interaction on the word as well as phrase level.

Strong Interaction Models These models directly build an interaction space between two sentences and model the interaction at different positions, such as ARC-II (Hu et al., 2014), MV-LSTM (Wan et al., 2016) and DF-LSTMs(Liu et al., 2016). These models can easily capture the difference between semantic capacity of two sentences.

In this paper, we propose a new deep neural network architecture to model the strong interactions

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1703–1712,
Austin, Texas, November 1-5, 2016.

of two sentences. Different with modelling two sentences with separated LSTMs, we utilize two interdependent LSTMs, called coupled-LSTMs, to fully affect each other at different time steps. The output of coupled-LSTMs at each step depends on both sentences. Specifically, we propose two interdependent ways for the coupled-LSTMs: loosely coupled model (LC-LSTMs) and tightly coupled model (TC-LSTMs). Similar to bidirectional LSTM for single sentence (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005), there are four directions can be used in coupled-LSTMs. To utilize all the information of four directions of coupled-LSTMs, we aggregate them and adopt a dynamic pooling strategy to automatically select the most informative interaction signals. Finally, we feed them into a fully connected layer, followed by an output layer to compute the matching score.

The contributions of this paper can be summarized as follows.

1. Different with the architectures of using similarity matrix, our proposed architecture directly model the strong interactions of two sentences with coupled-LSTMs, which can capture the useful local semantic relevances of two sentences. Our architecture can also capture the multiple granular interactions by several stacked coupled-LSTMs layers.

2. Compared to previous works on text matching, we perform extensive empirical studies on two very large datasets. The massive scale of the datasets allows us to train a very deep neural network and present an elaborate qualitative analysis of our models, which gives an intuitive understanding how our model worked.

2 Sentence Modelling with LSTM

Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies.

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The elements of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$.

The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}, \quad (1)$$

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where $\mathbf{x}_t$ is the input at the current time step; $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

The update of each LSTM unit can be written precisely as follows

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t). \quad (4)$$

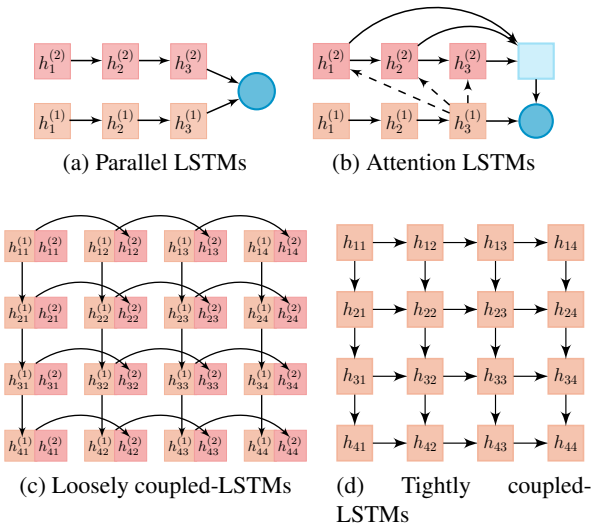
Here, the function $\mathbf{LSTM}(\cdot, \cdot, \cdot)$ is a shorthand for Eq. (1-3).

3 Coupled-LSTMs for Strong Sentence Interaction

To deal with two sentences, one straightforward method is to model them with two separate LSTMs. However, this method is difficult to model local interactions of two sentences. An improved way is to introduce attention mechanism, which has been used in many tasks, such as machine translation (Bahdanau et al., 2014) and question answering (Hermann et al., 2015).

Inspired by the multi-dimensional recurrent neural network (Graves et al., 2007; Graves and Schmidhuber, 2009; Byeon et al., 2015) and grid LSTM (Kalchbrenner et al., 2015) in computer vision community, we propose two models to capture the interdependences between two parallel LSTMs, called **coupled-LSTMs** (C-LSTMs).

To facilitate our models, we firstly give some definitions. Given two sequences $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_m$, we let $\mathbf{x}_i \in \mathbb{R}^d$ denote the embedded representation of the word x_i. The standard LSTM have one temporal dimension. When dealing with a sentence, LSTM regards the position as time step. At position i of sentence $x_{1:n}$,



(a) Parallel LSTMs (b) Attention LSTMs

(c) Loosely coupled-LSTMs (d) Tightly coupled-LSTMs

Figure 1: Four different coupled-LSTMs.

the output $\mathbf{h}_i$ reflects the meaning of subsequence $x_{0:i} = x_0, \cdots, x_i$.

To model the interaction of two sentences as early as possible, we define $\mathbf{h}_{i,j}$ to represent the interaction of the subsequences $x_{0:i}$ and $y_{0:j}$.

Figure 1(c) and 1(d) illustrate our two propose models. For intuitive comparison of weak interaction parallel LSTMs, we also give parallel LSTMs and attention LSTMs in Figure 1(a) and 1(b)[1].

We describe our two proposed models as follows.

3.1 Loosely Coupled-LSTMs (LC-LSTMs)

To model the local contextual interactions of two sentences, we enable two LSTMs to be interdependent at different positions. Inspired by Grid LSTM (Kalchbrenner et al., 2015) and word-by-word attention LSTMs (Rocktäschel et al., 2015), we propose a loosely coupling model for two interdependent LSTMs.

More concretely, we refer to $\mathbf{h}^{(1)}_{i,j}$ as the encoding of subsequence $x_{0:i}$ in the first LSTM influenced by the output of the second LSTM on subsequence $y_{0:j}$. Meanwhile, $\mathbf{h}^{(2)}_{i,j}$ is the encoding of subsequence $y_{0:j}$ in the second LSTM influenced by the output of the first LSTM on subsequence $x_{0:i}$

[1]In Rocktäschel et al. (2015) model, conditioned LSTM was used, meaning that $\mathbf{h}^{(1)}_1$ is produced conditioned on $\mathbf{h}^{(2)}_3$

$\mathbf{h}^{(1)}_{i,j}$ and $\mathbf{h}^{(2)}_{i,j}$ are computed as

$$\mathbf{h}^{(1)}_{i,j} = \mathbf{LSTM}^1(\mathbf{H}^{(1)}_{i-1}, \mathbf{c}^{(1)}_{i-1,j}, \mathbf{x}_i), \quad (5)$$

$$\mathbf{h}^{(2)}_{i,j} = \mathbf{LSTM}^2(\mathbf{H}^{(2)}_{j-1}, \mathbf{c}^{(2)}_{i,j-1}, \mathbf{y}_j), \quad (6)$$

where

$$\mathbf{H}^{(1)}_{i-1} = [\mathbf{h}^{(1)}_{i-1,j}, \mathbf{h}^{(2)}_{i-1,j}], \quad (7)$$

$$\mathbf{H}^{(2)}_{j-1} = [\mathbf{h}^{(1)}_{i,j-1}, \mathbf{h}^{(2)}_{i,j-1}]. \quad (8)$$

3.2 Tightly Coupled-LSTMs (TC-LSTMs)

The hidden states of LC-LSTMs are the combination of the hidden states of two interdependent LSTMs, whose memory cells are separated. Inspired by the configuration of the multi-dimensional LSTM (Byeon et al., 2015), we further conflate both the hidden states and the memory cells of two LSTMs. We assume that $\mathbf{h}_{i,j}$ directly model the interaction of the subsequences $x_{0:i}$ and $y_{0:j}$, which depends on two previous interaction $\mathbf{h}_{i-1,j}$ and $\mathbf{h}_{i,j-1}$, where i, j are the positions in sentence X and Y.

We define a tightly coupled-LSTMs *units* as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_{i,j} \\ \mathbf{o}_{i,j} \\ \mathbf{i}_{i,j} \\ \mathbf{f}^1_{i,j} \\ \mathbf{f}^2_{i,j} \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_j \\ \mathbf{h}_{i,j-1} \\ \mathbf{h}_{i-1,j} \end{bmatrix}, \quad (9)$$

$$\mathbf{c}_{i,j} = \tilde{\mathbf{c}}_{i,j} \odot \mathbf{i}_{i,j} + [\mathbf{c}_{i,j-1}, \mathbf{c}_{i-1,j}]^{\mathrm{T}} \begin{bmatrix} \mathbf{f}^1_{i,j} \\ \mathbf{f}^2_{i,j} \end{bmatrix} \quad (10)$$

$$\mathbf{h}_{i,j} = \mathbf{o}_t \odot \tanh(\mathbf{c}_{i,j}) \quad (11)$$

where the gating units $\mathbf{i}_{i,j}$ and $\mathbf{o}_{i,j}$ determine which memory units are affected by the inputs through $\tilde{\mathbf{c}}_{i,j}$, and which memory cells are written to the hidden units $\mathbf{h}_{i,j}$. $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. In contrast to the standard LSTM defined over time, each memory unit $\mathbf{c}_{i,j}$ of a tightly coupled-LSTMs has two preceding states $\mathbf{c}_{i,j-1}$ and $\mathbf{c}_{i-1,j}$ and two corresponding forget gates $\mathbf{f}^1_{i,j}$ and $\mathbf{f}^2_{i,j}$.

3.3 Analysis of Two Proposed Models

Our two proposed coupled-LSTMs can be formulated as

$$(\mathbf{h}_{i,j}, \mathbf{c}_{i,j}) = \mathbf{C\text{-}LSTMs}(\mathbf{h}_{i-1,j}, \mathbf{h}_{i,j-1}, \mathbf{c}_{i-1,j}, \mathbf{c}_{i,j-1}, \mathbf{x}_i, \mathbf{y}_j), \quad (12)$$

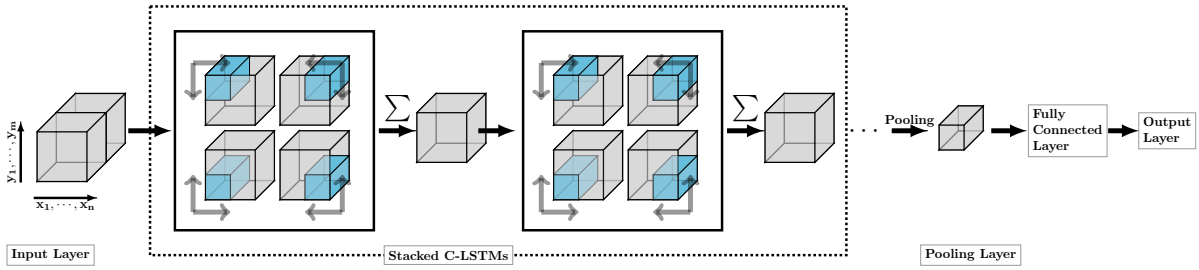


Figure 2: Architecture of coupled-LSTMs for sentence-pair encoding. Inputs are fed to four C-LSTMs followed by an aggregation layer. Blue cuboids represent different contextual information from four directions.

where **C-LSTMs** can be either **TC-LSTMs** or **LC-LSTMs**.

The input consists of two type of information at step (i, j) in coupled-LSTMs: temporal dimension $\mathbf{h}_{i-1,j}, \mathbf{h}_{i,j-1}, \mathbf{c}_{i-1,j}, \mathbf{c}_{i,j-1}$ and depth dimension $\mathbf{x}_i, \mathbf{y}_j$. The difference between TC-LSTMs and LC-LSTMs is the dependence of information from temporal and depth dimension.

Interaction Between Temporal Dimensions The TC-LSTMs model the interactions at position (i, j) by merging the internal memory $\mathbf{c}_{i-1,j}$ $\mathbf{c}_{i,j-1}$ and hidden state $\mathbf{h}_{i-1,j}$ $\mathbf{h}_{i,j-1}$ along row and column dimensions. In contrast with TC-LSTMs, LC-LSTMs firstly use two standard LSTMs in parallel, producing hidden states $\mathbf{h}^1_{i,j}$ and $\mathbf{h}^2_{i,j}$ along row and column dimensions respectively, which are then merged together flowing next step.

Interaction Between Depth Dimension In TC-LSTMs, each hidden state $\mathbf{h}_{i,j}$ at higher layer receives a fusion of information $\mathbf{x}_i$ and $\mathbf{y}_j$, flowed from lower layer. However, in LC-LSTMs, the information $\mathbf{x}_i$ and $\mathbf{y}_j$ are accepted by two corresponding LSTMs at the higher layer separately.

The two architectures have their own characteristics, TC-LSTMs give more strong interactions among different dimensions while LC-LSTMs ensures the two sequences interact closely without being conflated using two separated LSTMs.

Comparison of LC-LSTMs and word-by-word Attention LSTMs The characteristic of attention LSTMs is that they obtain the attention weighted representation of one sentence considering he alignment between the two sentences, which is asymmetric unidirectional encoding. Nevertheless, in LC-LSTM, each hidden state of each step is obtained with the consideration of interaction between two sequences with symmetrical encoding fashion.

4 End-to-End Architecture for Sentence Matching

In this section, we present an end-to-end deep architecture for matching two sentences, as shown in Figure 2.

4.1 Embedding Layer

To model the sentences with neural model, we firstly need transform the one-hot representation of word into the distributed representation. All words of two sequences $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_m$ will be mapped into low dimensional vector representations, which are taken as input of the network.

4.2 Stacked Coupled-LSTMs Layers

A basic block consists of five layers. We firstly use four directional coupled-LSTMs to model the local interactions with different information flows. And then we sum the outputs of these LSTMs by aggregation layer. To increase the learning capabilities of the coupled-LSTMs, we stack the basic block on top of each other.

4.2.1 Four Directional Coupled-LSTMs Layers

The C-LSTMs is defined along a certain predefined direction, we can extend them to access to the surrounding context in all directions. Similar to bi-directional LSTM, there are four directions in coupled-LSTMs.

$$(\mathbf{h}^1_{i,j}, \mathbf{c}^1_{i,j}) = \textbf{C-LSTMs}(\mathbf{h}_{i-1,j}, \mathbf{h}_{i,j-1}, \mathbf{c}_{i-1,j}, \mathbf{c}_{i,j-1}, \mathbf{x}_i, \mathbf{y}_j),$$
$$(\mathbf{h}^2_{i,j}, \mathbf{c}^2_{i,j}) = \textbf{C-LSTMs}(\mathbf{h}_{i-1,j}, \mathbf{h}_{i,j+1}, \mathbf{c}_{i-1,j}, \mathbf{c}_{i,j+1}, \mathbf{x}_i, \mathbf{y}_j),$$
$$(\mathbf{h}^3_{i,j}, \mathbf{c}^3_{i,j}) = \textbf{C-LSTMs}(\mathbf{h}_{i+1,j}, \mathbf{h}_{i,j+1}, \mathbf{c}_{i+1,j}, \mathbf{c}_{i,j+1}, \mathbf{x}_i, \mathbf{y}_j),$$

$$(\mathbf{h}^4_{i,j}, \mathbf{c}^4_{i,j}) = \textbf{C-LSTMs}(\mathbf{h}_{i+1,j}, \mathbf{h}_{i,j-1}, \mathbf{c}_{i+1,j}, \mathbf{c}_{i,j-1}, \mathbf{x}_i, \mathbf{y}_j).$$

4.2.2 Aggregation Layer

The aggregation layer sums the outputs of four directional coupled-LSTMs into a vector.

$$\hat{\mathbf{h}}_{i,j} = \sum_{d=1}^{4} \mathbf{h}^d_{i,j}, \tag{13}$$

where the superscript t of $\mathbf{h}_{i,j}$ denotes the different directions.

4.2.3 Stacking C-LSTMs Blocks

To increase the capabilities of network of learning multiple granularities of interactions, we stack several blocks (four C-LSTMs layers and one aggregation layer) to form deep architectures.

4.3 Pooling Layer

The output of stacked coupled-LSTMs layers is a tensor $\mathbf{H} \in \mathbb{R}^{n \times m \times d}$, where n and m are the lengths of sentences, and d is the number of hidden neurons. We apply dynamic pooling to automatically extract $\mathbb{R}^{p \times q}$ subsampling matrix in each slice $\mathbf{H}_i \in \mathbb{R}^{n \times m}$, similar to (Socher et al., 2011).

More formally, for each slice matrix $\mathbf{H}_i$, we partition the rows and columns of $\mathbf{H}_i$ into $p \times q$ roughly equal grids. These grid are non-overlapping. Then we select the maximum value within each grid thereby obtaining a $p \times q \times d$ tensor.

4.4 Fully-Connected Layer

The vector obtained by pooling layer is fed into a full connection layer to obtain a final more abstractive representation.

4.5 Output Layer

The output layer depends on the types of the tasks, we choose the corresponding form of output layer. There are two popular types of text matching tasks in NLP. One is ranking task, such as community question answering. Another is classification task, such as textual entailment.

1. For ranking task, the output is a scalar matching score, which is obtained by a linear transformation after the last fully-connected layer.

	MQA	RTE
Embedding size	100	100
Hidden layer size	50	50
Initial learning rate	0.05	0.005
Regularization	$5E{-}5$	$1E{-}5$
Pooling (p, q)	(2,1)	(1,1)

Table 1: Hyper-parameters for our model on two tasks.

2. For classification task, the outputs are the probabilities of the different classes, which is computed by a softmax function after the last fully-connected layer.

5 Training

Our proposed architecture can deal with different sentence matching tasks. The loss functions varies with different tasks. More concretely, we use max-margin loss (Bordes et al., 2013; Socher et al., 2013) for ranking task and cross-entropy loss for classification task.

To minimize the objective, we use stochastic gradient descent with the diagonal variant of AdaGrad (Duchi et al., 2011). To prevent exploding gradients, we perform gradient clipping by scaling the gradient when the norm exceeds a threshold (Graves, 2013).

6 Experiment

In this section, we investigate the empirical performances of our proposed model on two different text matching tasks: classification task (recognizing textual entailment) and ranking task (matching of question and answer).

6.1 Hyperparameters and Training

The word embeddings for all of the models are initialized with the 100d GloVe vectors (840B token version, (Pennington et al., 2014)) and fine-tuned during training to improve the performance. The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$.

For each task, we take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.05, 0.0005, 0.0001]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5, 1E{-}6]$ and the threshold

value of gradient norm [5, 10, 100]. The final hyper-parameters are set as Table 1.

6.2 Competitor Methods

- Neural bag-of-words (NBOW): Each sequence as the sum of the embeddings of the words it contains, then they are concatenated and fed to a MLP.
- Single LSTM: A single LSTM to encode the two sequences, which is used in (Rocktäschel et al., 2015).
- Parallel LSTMs: Two sequences are encoded by two LSTMs separately, then they are concatenated and fed to a MLP.
- Attention LSTMs: An attentive LSTM to encode two sentences into a semantic space, which used in (Hermann et al., 2015; Rocktäschel et al., 2015).
- Word-by-word Attention LSTMs: An improvement of attention LSTM by introducing word-by-word attention mechanism, which used in (Hermann et al., 2015; Rocktäschel et al., 2015).

6.3 Experiment-I: Recognizing Textual Entailment

Recognizing textual entailment (RTE) is a task to determine the semantic relationship between two sentences. We use the Stanford Natural Language Inference Corpus (SNLI) (Bowman et al., 2015). This corpus contains 570K sentence pairs, and all of the sentences and labels stem from human annotators. SNLI is two orders of magnitude larger than all other existing RTE corpora. Therefore, the massive scale of SNLI allows us to train powerful neural networks such as our proposed architecture in this paper.

6.3.1 Results

Table 2 shows the evaluation results on SNLI. The 3rd column of the table gives the number of parameters of different models without the word embeddings.

Our proposed two C-LSTMs models with four stacked blocks outperform all the competitor models, which indicates that our thinner and deeper network does work effectively.

Model	k	$\|\theta\|_M$	Test
NBOW	100	80K	75.1
single LSTM (Rocktäschel et al., 2015)	100	111K	80.9
parallel LSTMs (Bowman et al., 2015)	100	221K	77.6
Attention LSTMs (Rocktäschel et al., 2015)	100	252K	82.3
Attention(w-by-w) LSTMs (Rocktäschel et al., 2015)	100	252K	83.5
LC-LSTMs (Single Direction)	50	45K	80.5
LC-LSTMs	50	45K	80.9
four stacked LC-LSTMs	50	135K	84.3
TC-LSTMs (Single Direction)	50	77.5K	80.1
TC-LSTMs	50	77.5K	81.6
four stacked TC-LSTMs	50	190K	**85.1**

Table 2: Results on SNLI corpus.

Besides, we can see both LC-LSTMs and TC-LSTMs benefit from multi-directional layer, while the latter obtains more gains than the former. We attribute this discrepancy between two models to their different mechanisms of controlling the information flow from depth dimension.

Compared with attention LSTMs, our two models achieve comparable results to them using much fewer parameters (nearly $1/5$). By stacking C-LSTMs[2] , the performance of them are improved significantly, and the four stacked TC-LSTMs achieve 85.1% accuracy on this dataset.

Moreover, we can see TC-LSTMs achieve better performance than LC-LSTMs on this task, which need fine-grained reasoning over pairs of words as well as phrases.

6.3.2 Understanding Behaviors of Neurons in C-LSTMs

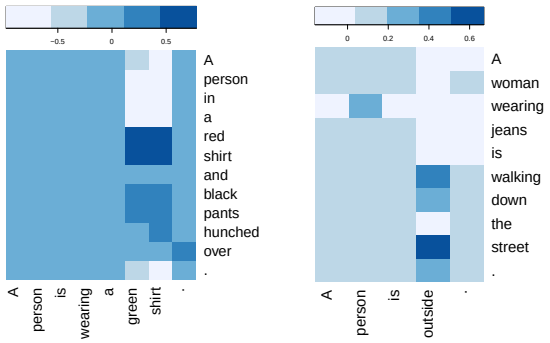
To get an intuitive understanding of how the C-LSTMs work on this problem, we examined the neuron activations in the last aggregation layer while evaluating the test set using TC-LSTMs. We find that some cells are bound to certain roles.

Let $h_{i,j,k}$ denotes the activation of the k-th neuron at the position of (i, j), where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. By visualizing the hidden state $\mathbf{h}_{i,j,k}$ and analyzing the maximum activation, we

[2]To make a fair comparison, we also train a stacked attention-based LSTM with the same setting as our models, while it does not make significant improvement with 83.7% accuracy.

Index of Cell	Word or Phrase Pairs
3-th	(in a pool, swimming), (near a fountain, next to the ocean), (street, outside)
9-th	(doing a skateboard, skateboarding), (sidewalk with, inside), (standing, seated)
17-th	(blue jacket, blue jacket), (wearing black, wearing white), (green uniform, red uniform)
25-th	(a man, two other men), (a man, two girls), (an old woman, two people)

Table 3: Multiple interpretable neurons and the word-pairs/phrase-pairs captured by these neurons.



(a) 3rd neuron (b) 17th neuron

Figure 3: Illustration of two interpretable neurons and some word-pairs capture by these neurons. The darker patches denote the corresponding activations are higher.

can find that there exist multiple interpretable neurons. For example, when some contextualized local perspectives are semantically related at point (i, j) of the sentence pair, the activation value of hidden neuron $h_{i,j,k}$ tend to be maximum, meaning that the model could capture some reasoning patterns.

Figure 3 illustrates this phenomenon. In Figure 3(a), a neuron shows its ability to monitor the local contextual interactions about color. The activation in the patch, including the word pair "`(red, green)`", is much higher than others. This is informative pattern for the relation prediction of these two sentences, whose ground truth is contradiction. An interesting thing is there are two words describing color in the sentence "`A person in a red shirt and black pants hunched over.`". Our model ignores the useless word "`black`", which indicates that this neuron selectively captures pattern by contextual understanding, not just word level interaction.

In Figure 3(b), another neuron shows that it can capture the local contextual interactions, such as "`(walking down the street, outside)`". These patterns can be easily captured by pooling layer and provide a strong support for the final prediction.

Table 3 illustrates multiple interpretable neurons and some representative word or phrase pairs which can activate these neurons. These cases show that our models can capture contextual interactions beyond word level.

6.3.3 Error Analysis

Although our models C-LSTMs are more sensitive to the discrepancy of the semantic capacity between two sentences, some semantic mistakes at the phrasal level still exist. For example, our models failed to capture the key informative pattern when predicting the entailment sentence pair "`A girl` **`takes off her shoes`** `and eats blue cotton candy/The girl is eating while` **`barefoot.`**"

Besides, despite the large size of the training corpus, it's still very different to solve some cases, which depend on the combination of the world knowledge and context-sensitive inferences. For example, given an entailment pair "`a man grabs his crotch during a political demonstration/The man is making a crude gesture`", all models predict "`neutral`". This analysis suggests that some architectural improvements or external world knowledge are necessary to eliminate all errors instead of simply scaling up the basic model.

6.4 Experiment-II: Matching Question and Answer

Matching question answering (MQA) is a typical task for semantic matching. Given a question, we need select a correct answer from some candidate answers.

In this paper, we use the dataset collected from Yahoo! Answers with the getByCategory function

Model	k	P@1(5)	P@1(10)
Random Guess	-	20.0	10.0
NBOW	50	63.9	47.6
single LSTM	50	68.2	53.9
parallel LSTMs	50	66.9	52.1
Attention LSTMs	50	73.5	62.0
Attention LSTMs (w-by-w)	50	75.1	64.0
LC-LSTMs (Single Direction)	50	75.4	63.0
LC-LSTMs	50	76.1	64.1
three stacked LC-LSTMs	50	**78.5**	**66.2**
TC-LSTMs (Single Direction)	50	74.3	62.4
TC-LSTMs	50	74.9	62.9
three stacked TC-LSTMs	50	77.0	65.3

Table 4: Results on Yahoo question-answer pairs dataset.

provided in Yahoo! Answers API, which produces $963,072$ questions and corresponding best answers. We then select the pairs in which the length of questions and answers are both in the interval $[4, 30]$, thus obtaining $220,000$ question answer pairs to form the positive pairs.

For negative pairs, we first use each question's best answer as a query to retrieval top $1,000$ results from the whole answer set with Lucene, where 4 or 9 answers will be selected randomly to construct the negative pairs.

The whole dataset is divided into training, validation and testing data with proportion $20 : 1 : 1$. Moreover, we give two test settings: selecting the best answer from 5 and 10 candidates respectively.

6.4.1 Results

Results of MQA are shown in the Table 4. For our models, due to stacking block more than three layers can not make significant improvements on this task, we just use three stacked C-LSTMs.

By analyzing the evaluation results of question-answer matching in table 4, we can see strong interaction models (attention LSTMs, our C-LSTMs) consistently outperform the weak interaction models (NBOW, parallel LSTMs) with a large margin, which suggests the importance of modelling strong interaction of two sentences.

Our proposed two C-LSTMs surpass the competitor methods and C-LSTMs augmented with multi-directions layers and multiple stacked blocks fully utilize multiple levels of abstraction to directly boost the performance.

Additionally, LC-LSTMs is superior to TC-LSTMs. The reason may be that MQA is a relative simple task, which requires less reasoning abilities, compared with RTE task. Moreover, the parameters of LC-LSTMs are less than TC-LSTMs, which ensures the former can avoid suffering from overfitting on a relatively smaller corpus.

7 Related Work

Our architecture for sentence pair encoding can be regarded as strong interaction models, which have been explored in previous models.

An intuitive paradigm is to compute similarities between all the words or phrases of the two sentences. Socher et al. (2011) firstly used this paradigm for paraphrase detection. The representations of words or phrases are learned based on recursive autoencoders.

A major limitation of this paradigm is the interaction of two sentence is captured by a pre-defined similarity measure. Thus, it is not easy to increase the depth of the network. Compared with this paradigm, we can stack our C-LSTMs to model multiple-granularity interactions of two sentences.

Rocktäschel et al. (2015) used two LSTMs equipped with attention mechanism to capture the iteration between two sentences. This architecture is asymmetrical for two sentences, where the obtained final representation is sensitive to the two sentences' order.

Compared with the attentive LSTM, our proposed C-LSTMs are symmetrical and model the local contextual interaction of two sequences directly.

8 Conclusion and Future Work

In this paper, we propose an end-to-end deep architecture to capture the strong interaction information of sentence pair. Experiments on two large scale text matching tasks demonstrate the efficacy of our proposed model and its superiority to competitor models. Besides, we present an elaborate qualitative analysis of our models, which gives an intuitive understanding how our model worked.

In future work, we would like to incorporate some gating strategies into the depth dimension of our proposed models, like highway or residual network, to enhance the interactions between depth and other di-

mensions thus training more deep and powerful neural networks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. 2015. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

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

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. In *Artificial Neural Networks–ICANN 2007*, pages 549–558. Springer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016. Deep fusion LSTMs for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In

Advances in Neural Information Processing Systems, pages 926–934.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Universal Decompositional Semantics on Universal Dependencies

Aaron Steven White Drew Reisinger Keisuke Sakaguchi Tim Vieira
Sheng Zhang Rachel Rudinger Kyle Rawlins Benjamin Van Durme
Johns Hopkins University

Abstract

We present a framework for augmenting data sets from the Universal Dependencies project with *Universal Decompositional Semantics*. Where the Universal Dependencies project aims to provide a syntactic annotation standard that can be used consistently across many languages as well as a collection of corpora that use that standard, our extension has similar aims for semantic annotation. We describe results from annotating the English Universal Dependencies treebank, dealing with word senses, semantic roles, and event properties.

1 Introduction

This paper describes the *Universal Decompositional Semantics* (Decomp) project, which aims to augments Universal Dependencies (UD) data sets with robust, scalable semantic annotations based in linguistic theory. The UD project[1] aims to provide (i) a syntactic dependency annotation standard that can be used consistently across many languages and (ii) a collection of corpora that use that standard (De Marneffe et al., 2014; Nivre et al., 2015). Decomp provides complementary semantic annotations that scale across different types of semantic information and different languages and can integrate seamlessly with any UD-annotated corpus.

Decomp has two mutually supportive tenets—*semantic decomposition* and *simplicity*. As we discuss further in the next section, these tenets allow us to collect annotations from everyday speakers of a language that are rooted in basic, commonsensical aspects of meaning and that can be straightforwardly explained and generally agreed upon in context.

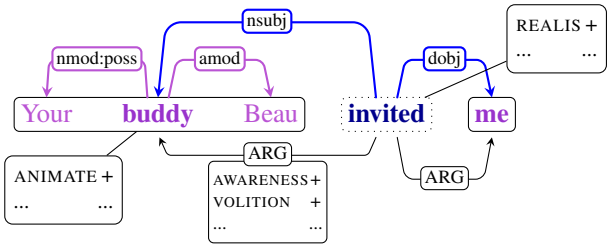


Figure 1: Decompositional semantics atop syntax.

In this paper, we describe Decomp protocols for three domains—semantic role decomposition, event decomposition, and word sense decomposition—and we present annotation results on top of the English UD v1.2 (EUD1.2) treebank.[2] We begin in §2 by connecting Decomp with previous work on decomposition in linguistic theory. In §3, we present *PredPatt*, which is a software package for preprocessing UD annotated corpora for input into Decomp protocols. In §4, we present a major revision to Reisinger et al.'s (2015) semantic role decomposition protocol (SPR1). Our revision, SPR2, brings SPR1 into full alignment with Decomp while adding various new features. In §5, we present Decomp-aligned annotation of event properties, focusing specifically on event realis. Finally, in §6, we describe a Decomp-aligned word sense decomposition protocol and associated set of annotations.

2 Universal Decompositional Semantics

A range of perspectives suggest that the proper representation for word meanings is *decompositional*.

[1] http://universaldependencies.org

[2] All datasets are available at http://decomp.net.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1713–1723,
Austin, Texas, November 1-5, 2016.

For example, Dowty (1979) followed by a substantial amount of research (e.g. Jackendoff 1990; Rappaport-Hovav and Levin 1998; Levin and Rappaport Hovav 2005) suggests that word meanings can be factored into (i) idiosyncratic, item-specific components and (ii) general components, such as CAUSATION, that are used across the lexicon. In the domain of thematic roles, Dowty (1991) argues that notions such as AGENT should be decomposed into simpler, more primitive properties such as volitional participation in an event. Pustejovsky (1991) decomposes word meanings into *qualia structures* that again incorporate more primitive properties of events and individuals. In spite of this wealth of theory, existing annotation protocols rarely take the idea into account, operating at the level that the above approaches decompose with very few exceptions (Greene and Resnik, 2009; Hartshorne et al., 2013; Reisinger et al., 2015). Decomp's premise is that a decompositional approach to large-scale annotation has benefits for both the annotation process and downstream uses of annotated data (cf. He et al. 2015; Bos et al. 2017 for recent non-decompositional approaches).

To capture these benefits, Decomp incorporates *semantic decomposition* directly into its protocols by mapping from decompositional theories, such as Dowty's, into straightforward questions on binary properties that are easily answered. This method of constructing annotation protocols gives rise to *simplicity* in the protocol, since the resulting questions are much more commonsensical and easily explained than the concepts they are decomposing. For instance, Dowty (1991) decomposes the relatively unintuitive notion (for ordinary speakers) of AGENT into much simpler properties, such as VOLITION and MOVEMENT. Instead of asking about whether a verbal argument is an AGENT (with the concomitant complex training process for annotators), a Decomp protocol might then ask about whether the referent of the argument had volition in or moved as a result of the event. Simplicity in the protocol in turn allows a Decomp protocol to gather annotations from untrained native (and naïve) speakers of a language. In large part, the current paper is focused on developing questions that everyday speakers can agree on and that are key for lexical representations.

An added benefit of questions on binary properties is they allow the use of ordinal prompts (Likert scales), allowing annotators to record subjective uncertainty of a property in a given context, which can then be aggregated across multiple responses with less severe impact to inter-annotator agreement.

3 Predicative patterns

In this section, we introduce PredPatt, which is a lightweight tool for identifying the structure of predicates and arguments from Universal Dependencies. We use the PredPatt's output as input to the Decomp aligned annotation protocols we describe in §4 – §6. To ensure that this output is accurate, we evaluate on multiple UD-annotated corpora: automatically generated English parses and gold treebanks in Chinese, English, Hebrew, Hindi, and Spanish.[3]

PredPatt employs deterministic, unlexicalized rules over UD parses. We provide a high-level overview of the process here.[4]

Input UD Parse with Universal POS tags
1. Predicate and argument root identification
2. Argument resolution
3. Predicate and argument phrase extraction
4. Optional Post-processing

Output collection of predicate-argument structures

UD Parse A universal dependency (UD) parse, is a set of labeled pairs. Each pair has the form RELATION(DEPENDENT, GOVERNOR). The UD parse also includes a sequence of Universal POS tags. An example of a UD parse is in Figure 1.

Predicate and argument root identification Predicate and argument roots (i.e., dependency tree nodes) are identified by local configurations—specifically, edges in the UD parse. The simplest example is NSUBJ(s, v) and DOBJ(o, v), which indicate that v is a predicate root, and that s and o are argument roots. Similarly, roots of clausal subjects and clausal complements are also predicate roots. Nominal modifiers inside adverbial modifiers

[3] While we are not aware of a similar tool for Universal Dependencies, PredPatt is similar to ClausIE (Del Corro and Gemulla, 2013) and ArgOE (Gamallo et al., 2012), which supports Spanish, Portuguese, Galician and English.

[4] A detailed description of PredPatt is available at `https://github.com/hltcoe/PredPatt`. PredPatt derives from the system described by Rudinger and Van Durme (2014).

are arguments to the verb being modified, e.g., *Investors turned away from [the stock market]*. PredPatt also extracts relations from appositives, possessives, copula, and adverbial modifiers.

Argument resolution PredPatt includes argument resolution rules to handle missing arguments of many syntactic constructions, including predicate coordination, relative clauses, and embedded clauses. Argument resolution is crucial in languages that mark arguments using morphology, such as Spanish and Portuguese, because there are more cases of covert subjects. Other common cases for argument resolution are when predicates appear in a conjunction, e.g., *Chris likes to sing and dance*, has no arc from *dance* to its subject *Chris*. In relative clauses, the arguments of an embedded clause appear outside the subtree, e.g., *borrowed* in *The books John borrowed from the library are overdue.* has *books* as an argument and so does *are-overdue*.

Predicate extraction PredPatt extracts a descriptive name for *complex predicates*. For example, ***[PredPatt] finds [structure] in [text]*** has a 3-place predicate named (`?a` **finds** `?b` **in** `?c`). The primary logic here is (a) to lift mark and case tokens (e.g., *in*) out of the argument subtree, (b) to add adverbial modifiers, auxiliaries, and negation (e.g., ***[Chris] did not sleep quietly***). PredPatt uses the text order of tokens and arguments to derive a name for the predicate; no effort is made to further canonicalize this name, nor align it to a verb ontology.

Argument phrase extraction Argument extraction filters tokens from the dependency subtree below the argument root. These filters primarily simplify the subtree, e.g., removing relative clauses and appositives inside an argument. The default set of filters were chosen to preserve meaning, since it is not generally the case that all modifiers can safely be dropped (more aggressive argument simplification settings are available as options).

Post-processing PredPatt implements a number of optional post-processing routines, such as conjunction expansion, argument simplification (which filters out non-core arguments, leaving only subjects and objects), and language-specific hooks.[5]

Lang	#Sent	#Output	Precision
Chinese	98	375	69.1% ±4.7%
English	79	210	86.2% ±4.7%
Hebrew	12	30	66.7% ±17.9%
Hindi	22	50	52.0% ±14.3%
Spanish	27	55	70.9% ±12.4%

Table 1: Results of manual evaluation of PredPatt on UD

Gold treebanks in multiple languages We evaluated PredPatt manually on several randomly sampled sentences taken from the UD banks of Chinese, English, Hebrew, Hindi and Spanish. This evaluation runs PredPatt with the gold standard UD parse. We report the number of sentences evaluated along with the number of extractions from those sentences (a proxy for recall) and precision (95% confidence interval) for each language in Table 1.

4 Semantic role decomposition

A decompositional strategy has been successfully used by Reisinger et al. (2015) to annotate thematic role information under their Semantic Proto-Role labeling protocol (SPR1), which is based on Dowty's (1991) seminal thematic proto-role theory.[6]

In this section, we present a major revision to SPR1 aimed at strengthening and generalizing the protocol beyond Reisinger et al.'s dataset. We present three pilots aimed at validating our new protocol as well as a bulk annotation of a large subset of core arguments in EUD1.2. Finally, we describe, deploy, and validate methods for extending SPR2's reach beyond this subset, resulting in SPR2.1.

4.1 SPR1 protocol

In the SPR1 protocol, each core argument of a verb is annotated for the likelihood that particular properties hold of that argument's referent as a participant in the event denoted by the verb.

Property questions The properties selected for this purpose, given in Table 2, are based on those invoked by Dowty (1991) in his prototype-theoretic

[5]UD itself allows for language-specific exceptions to the "universal" standard, and we therefore allow that practice here.

[6]See Kako 2006; Greene and Resnik 2009; Madnani et al. 2010; Hartshorne et al. 2013 for work using similar protocols.

Role property	How likely or unlikely is it that...
instigation	ARG caused the PRED to happen?
volition	ARG chose to be involved in the PRED?
awareness	ARG was/were aware of being involved in the PRED?
sentient	ARG was/were sentient?
change of location	ARG changed location during the PRED?
-exists as physical	ARG existed as a physical object?
existed before	ARG existed before the PRED began?
existed during	ARG existed during the PRED?
existed after	ARG existed after the PRED stopped?
change of possession	ARG changed possession during the PRED?
change of state	ARG was/were altered or somehow changed during or by the end of the PRED?
-stationary	ARG was/were stationary during the PRED?
-location of event	ARG described the location of the PRED?
-physical contact	ARG made physical contact with someone or something else involved in the PRED?
was used	ARG was/were used in carrying out the PRED?
-pred changed arg	The PRED caused a change in ARG?
+was for benefit	PRED happened for the benefit of ARG?
+partitive	*Only* a part or portion of ARG was involved in the PRED?
+change of state continuous	The change in ARG happened throughout the PRED?

Table 2: Questions posed to annotators. + indicates questions new to SPR2; - indicates SPR1 questions dropped in SPR2.

reconstruction of linking theory. Reisinger et al.'s (2015) SPR1 dataset, produced under this protocol, provides annotations of the Wall Street Journal portions of the Penn Treebank (PTB; Marcus et al. 1993) that are also annotated for core argument PropBank (Palmer et al., 2005) roles.

Filtering and data collection In Reisinger et al. 2015, verbs were excluded that occur in certain syntactic environments that interfere with property judgments. In particular, participles and imperatives were excluded, as well as verbs in embedded clauses, in questions, and under negation or auxiliaries. We carry these filters forward to our own bulk annotation by implementing them over PredPatt output and then show how these filters can be lifted.

Annotators To ensure internal consistency of the judgments, Reisinger et al.'s data was based on a single Amazon Mechanical Turk annotator.

4.2 SPR2 protocol

First we update both the set of questions and the method for presenting these questions in order to streamline the annotation process and simplify Reisinger et al.'s protocol. Second, to deal with potentially ungrammatical sentences, as well as to add an extra layer of quality control to the generation of property questions, we add an acceptability judgment question to the protocol. Finally, we collect annotations from multiple trusted annotators with two-way redundancy, allowing us to normalize the data in a way that is impossible with SPR1.

Property questions While Reisinger et al.'s properties were mainly motivated by linguistic theory, in the process of developing SPR2 we identified several redundancies as well as potential sources of error; these changes are summarized in Table 2. Redundancies include `stationary` being essentially the negation of `change of location`, and `predicate changed argument` being almost identical to `change of state`. The property `exists as physical` was dropped because it is a purely referential (non-relational) property of the argument; thus, it is redundant with our more elaborated decompositional word sense protocol. The `location of event` and `physical contact` properties were removed because of lower interannotator agreement and high within-annotator response variance in SPR1.

In addition to this streamlining, we added three new properties that target new types of arguments: benefactives, partitives, and incremental themes. Benefactive arguments and partitive arguments often have special morphosyntactic properties in many languages. In English, for example, benefactives can appear in double-object constructions with verbs like *buy*, and in many languages they correlate with special morphology. Partitives involve partial affectedness and similarly are often marked with morphological case (Kiparsky, 1998). The third new property, `change of state continuous`, is a plain-language version of Dowty's (1991) *incremental theme* proto-role property, which Reisinger et al. (2015) did not include. An argument is an INCREMENTAL THEME with respect to an event if the temporal progress of the event can be measured in terms of, or put into correspondence with, the part-whole structure of that argument which undergoes some gradual change (Tenny, 1987; Krifka, 1989, a.o.). For example, in an event of *mowing the lawn*, *the lawn* is an incremental theme because the progress of mowing is directly related to the portion of the lawn that has been mowed. Though incremental theme is quite abstract in comparison to other proto-role properties, it is widely agreed that something like this property is involved in linking thematic roles to syntactic position.

Dynamic reveal The question corresponding to the `change of state continuous` property

presupposes that the argument under consideration did, in fact, undergo some change of state. This means that if an annotator has previously determined that the property `change of state` does not apply, then asking about `change of state continuous` is at best inefficient, since we can deterministically predict that the answer should be *NA*, and at worst confusing to the annotator, since the question triggers a presupposition failure.

To avoid such presupposition failures in SPR2, which we suspect led to additional noise and annotation time as part of SPR1, we modified the annotation interface so that certain questions are revealed dynamically based on the answers to other questions. The set of questions is now organized hierarchically instead of as a flat list. In this hierarchical structure, `change of state` is a parent of `change of state continuous`, which means that the latter question only appears if the annotator gives a high ordinal value to the former. Questions that remain hidden are assumed to have *NA* as their answer. For SPR2, this pair of properties is the only one affected by the dynamic reveal feature, though this aspect of the protocol will be extended in later versions.

Acceptability judgments Two kinds of grammatical acceptability judgments were collected. The first kind, collected on a five-point scale, asked about the acceptability of the sentence containing the argument in question. The second kind, collected as a binary judgment, asked whether the question was hard to answer because of grammatical errors. This second was triggered only when annotators gave a response on the bottom three values of the ordinal scale for the relevant property question. We do not analyze these judgments here for reasons of space, but they are available as part of the released dataset.

Multiple annotators with redundancy Reisinger et al.'s (2015) reason for not using redundant annotations was that a single annotator would provide internally consistent judgments, but this consistency comes at the cost of potential bias in the judgments.[7] In order to evaluate bias, we move to

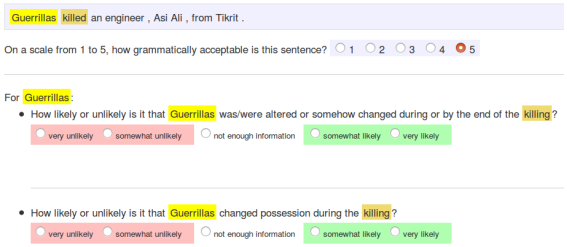


Figure 2: Example of semantic role decomposition task.

two-way redundancy (and later versions of the protocol are compatible with greater redundancy).

Heterogeneous data SPR2 extends the coverage of Semantic Proto-Role Labeling to heterogeneous genres. The SPR1 dataset contains only annotations of newswire text. This is not ideal for either practical or scientific purposes, since newswire tends to be biased toward otherwise rare word senses—often pertaining to financial markets—but low coverage of otherwise common word sense.

To remedy these coverage issues, we extend SPR1 to the English Universal Dependencies (version 1.2) treebank (EUD1.2). EUD1.2 is based on the Linguistic Data Consortium's English Web Treebank (Bies et al., 2012) and contains a much wider set of genres than the Penn Treebank—including weblogs, newsgroup discussions, emails from the EnronSent Corpus, reviews from English Google reviews, and answers from Yahoo! Answers. EUD1.2 has the added benefit of being natively annotated with gold-standard Universal Dependencies (UD) parses (Nivre et al., 2015).

4.3 Pilot experiments

In this section, we present three pilot experiments conducted on a subset of EUD1.2 and aimed at validating the updated protocol in preparation for deployment of the full task. In the first, we use the SPR1 protocol to obtain judgments on a small sample of EUD1.2 sentences from the same trusted annotator that produced SPR1. In the second, we open the same task to multiple annotators. And in the third, we deploy our updated SPR2 protocol on the

[7] For example, in analyzing the SPR1 dataset that Reisinger et al. make available, we noted that their annotator has a somewhat idiosyncratic way of answering the `was used` question, which aims at identifying instruments: the annotator marks *him* in (i) as likely to have been used in carrying out the advising.

(i) Sen. Bill Bradley of New Jersey *advised* **him** that the Dow Jones Industrial Average had declined by 190 points.

This is a general pattern for this question for this annotator.

same subset of EUD1.2—again, open to multiple annotators. We use these three pilots to evaluate interannotator agreement within and across protocols (where possible) and to construct a pool of trusted annotators to work on the full annotation task.

Item selection For each pilot, the same set of sentences were used. These sentences were selected based on properties of both the predicate and its corresponding arguments in each sentence. The pilot experiments were limited to the same 10 verbs (*want*, *put*, *think*, *see*, *know*, *look*, *say*, *take*, *tell*, *give*) that were considered in Reisinger et al.'s pilot.

As in Reisinger et al. 2015, tokens were excluded with verbs that occur in certain syntactic environments that interfere with property judgments. We used the same filters described by those authors, modified for UD. Additionally, verbs occurring as the second item in a conjunction were removed, as EUD1.2 does not have sufficient annotation to identify all arguments of such verbs from the syntax.

Verbal arguments were defined as the subtrees governed by a verb via a core grammatical relation (`nsubj`, `nsubpass`, `dobj`, and `iobj`). In addition, occurrences of the pronoun *it* in subject position were excluded because of inconsistencies in the annotation of expletive subjects in EUD1.2.

Pilots 1 & 2: SPR1 protocol Pilot 1, designed to compare SPR1 directly to SPR2, used the same protocol described in Reisinger et al. 2015 and was deployed on 99 argument tokens selected based on the method above.[8] To ensure that the only difference between SPR1 and this pilot was which sentences were annotated, we obtained the AMT identifier for the SPR1 annotator from Reisinger et al. Thus, the only annotator in this pilot was the same one that produced all the annotations for the SPR1 dataset.

The data from this pilot cannot be compared to the SPR1 dataset on a token level, since the items do not come from the same dataset. But these data can be compared to the SPR1 dataset on a type level by averaging responses to particular questions asked about particular argument positions (e.g., `nsubj`, `dobj`, etc.) for a particular predicate (e.g., *want*, *put*, etc.) and then comparing the correlation between these averages. The average type-level correlation between the average by-predicate, by-argument relation ratings in the SPR1 dataset and those in the current pilot was high for all verb-argument pairs (Spearman ρ=0.82).

Pilot 2 uses the same materials and protocol as Pilot 1. The only difference between the two is that this pilot was open to multiple annotators. A total of 33 annotators participated, one of whom was the same annotator that produced all the annotations for the SPR1 dataset and participated in Pilot 1.

For each argument token, we collected five judgments per property question. Interannotator agreement was calculated by argument token for the likelihood responses using pairwise Spearman rank correlations. The mean ρ across all annotator pairs and argument tokens was 0.562 (95% CI=[0.549, 0.574]) and, due to heavy left skew, the median was 0.618 (95% CI=[0.603, 0.631]). This agreement is relatively high, suggesting that different annotators tend to agree on the relative likelihood of a property applying to an argument.

Since the SPR1 and Pilot 1 annotator was among this group, we can also assess the extent to which the Pilot 1 annotator is consistent with other annotators. Comparing this annotator to every other annotator that annotated the same argument token, the mean ρ was 0.499 (95% CI=[0.451, 0.546]), and the median was 0.565 (95% CI=[0.504, 0.637]). This suggests that, on average, the other annotators are even more consistent with each other than they are with the original SPR1 annotator, vindicating the use of multiple annotators.

Pilot 3: SPR2 protocol Pilot 3 uses the same materials as Pilots 1 and 2 but introduces the SPR2 protocol laid out above. A total of 57 annotators participated in this pilot. For each argument token, we again collected five judgments per property.

Interannotator agreement was calculated by argument token for the likelihood responses using pairwise Spearman rank correlations. The mean ρ across all annotator pairs and argument tokens was 0.622 (95% CI=[0.610, 0.634]) and, again due to heavy left skew, the median was 0.677 (95% CI=[0.662, 0.690]). This higher agreement compared to Pilot 2 likely arises due to the fact that we have fewer questions in the SPR2 protocol and suggests that we suc-

[8] For each verb, 10 arguments were selected, with the exception of *see*, which only had 9 due to an off-by-one error.

ceeded in removing noisy questions without adding questions that were similarly noisy.

Since we use the same materials as Pilots 1 and 2, we can also compare the SPR1 and SPR2 protocols on the subset of questions they share. We find similar mean agreement, at 0.593 (95% CI=[0.580, 0.607]), and median agreement, at 0.665 (95% CI=[0.652, 0.672]), to that we found within the Pilots 2 and 3 results. This suggests that the addition and subtraction of questions does not substantially alter annotators' judgments on the questions that both protocols share.

4.4 Trusted annotator pool

To ensure annotation consistency in our bulk annotation, we constructed a pool of trusted annotators from those annotators that participated in Pilots 2 and 3. We used two metrics to construct this pool: rating agreement and applicability agreement. Both of these metrics control for various factors that might raise or lower agreement independent of the annotator—e.g., the particular question, the particular sentence, the particular argument type, etc.—using generalized linear mixed effects models. This pool contains a total of 86 trusted annotators.

4.5 Bulk task

For our bulk task, we used the SPR2 protocol to annotate a total of 3,806 argument tokens spanning 2,759 unique predicate lemmas. These argument tokens were part of a filtered set constructed using Reisinger et al.'s filtering scheme described above.

We collected two judgments per property, per argument token. Interannotator agreement was calculated in the same way as for the pilots. The mean ρ was 0.617 (95% CI=[0.611, 0.623]), and the median was 0.679, (95% CI=[0.673, 0.686]). This agreement is very close to that found in the pilots, suggesting that rating consistency extends beyond the constrained set of predicates used in the pilots.

One issue with SPR1 that remains unaddressed in SPR2 is the use of filters. This significantly reduces the potential coverage of the protocol and relies on extremely rich syntactic annotation. This second is not problematic when we have gold standard treebanks like EUD1.2, but it becomes an issue when moving beyond such treebanks.

To alleviate this filter issue, we propose a further revision of SPR2. In this version (SPR2.1), we alter the SPR2 instructions to take into account cases where the property questions may be difficult to answer. These fall into at least three categories: eventualities that haven't happened (*irrealis eventualities*), generics, and habituals. In SPR2.1, annotators are instructed about each case and to answer as if a specific event of that kind did actually happen.

We annotated predicates that occurred in a sentence from the previous bulk task but were filtered from that task based on Reisinger et al.'s (2015) filters. We have so far annotated all such predicates with less than 100 instances in all of EUD1.2 and plan to continue annotation to get full coverage of these sentences.

A total of 26 annotators from our trusted pool participated in this annotation. As in the previous bulk task, we collected two judgments per property, per argument token. The interannotator agreement was calculated in the same way as for the previous bulk task and pilots and was reasonably high with a mean ρ of 0.528 (95% CI=[0.522, 0.535]) and median ρ of 0.571 (95% CI=[0.563, 0.580]). This somewhat lower agreement is to be expected, since these predicates were selected to be harder than those in the previous task.

4.6 Discussion

We presented a major revision to Reisinger et al.'s (2015) decompositional Semantic Proto-Role Labeling protocol (SPR1) and deployed this revised protocol (SPR2) in three validation pilots and a bulk task. We then described two extensions to this protocol aimed at expanding the annotable arguments.

One issue that arises with SPR2.1 is that it substantially complicates the instructions, clashing with Decomp *simplicity* tenet. In the next section, we describe a task aimed at allowing us to better target predicates that need these more elaborated instructions, allowing us to use the simpler SPR2 protocol where possible.

5 Event decomposition

As discussed in §4, SPR1 and SPR2 employ filters that run on top of dependency parses to ensure that proto-role property questions about particular argu-

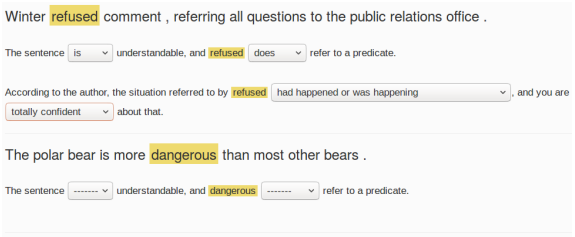


Figure 3: Example of the event decomposition task

ments are answerable. We showed that these filters can be bypassed by altering the instructions given to annotators. This approach substantially increases the length of the tasks instructions, however, and so ideally, these lengthened instructions should be used only when absolutely necessary. One place it seems likely to be necessary is when the event in a sentence did not in fact occur.

In this section, we present a protocol, inspired by the one developed by de Marneffe et al. (2012), for targeting these sorts of sentence with special instructions in future versions of SPR (see also Saurí and Pustejovsky 2012). A major benefit of this protocol is that it produces a foundation for future decompositional event annotations.

Protocol The protocol has four major components: questions about (i) whether or not a particular word refers to an eventuality (event or state); (ii) whether the sentence is understandable; (iii) whether or not, according to the author, the event has already happened is currently happening; and (iv) how confident the annotator is about their answer to (iii).

The first two components were included to filter out items that are either incorrectly labeled as predicates or that the annotator could not annotate for components (iii) and (iv), and if an annotator answered *no* to either for a particular predicate candidate, (iii) and (iv) did not appear. Thus, like SPR2.x, this protocol incorporates a hierarchy of questions that can be elaborated in future versions.

Data collection We applied this protocol to every predicate candidate found in an EUD1.2 sentence annotated under SPR2 and SPR2.1. This yields annotations for a superset of the predicates annotated under SPR2.x, and thus components (i) and (ii) of these annotations can be used as a *post hoc* filter on the SPR2.x annotations or to decide on whether to include a predicate for future SPR2.x tasks.

A total of 6,930 predicate candidates were annotated in batches of 10 by 24 unique annotators recruited from the trusted annotator pool built for SPR2.x. Each predicate candidate was judged by two distinct annotators.

Data validation For each of the four components interannotator agreement was computed by each group of 10 predicates. For the categorical responses, we would ideally use Cohen's κ, but there were so many cases of perfect agreement for the categorical responses that Cohen's κ is ill-defined in many cases. As such, we report raw agreement here.

The mean raw agreement for whether each predicate candidate was a predicate was 0.955 (95% CI=[0.950, 0.960]). The mean raw agreement for whether the sentence was understandable was [0.976, (95% CI=[0.971, 0.980]); and the mean raw agreement for whether the eventuality happened or was happening was 0.820 (95% CI=[0.811, 0.829]).

Discussion We presented the first version of a new event decomposition protocol. This protocol integrates with and is in the same spirit as the SPR2.x protocols produced in the previous section.

In the next section, we describe a complementary protocol for decomposing word sense, focusing specifically on noun senses. This last protocol completes a picture wherein we decompose predicate argument semantics into three parts: the properties of a predicate independent of its arguments, the properties of a predicate's arguments in relation to the event the predicate denotes, and the properties of an argument independent of the predicate.

6 Word sense decomposition

In §4 and §5, we focused on semantic questions that deal with eventualities. In this section, we describe a Decomp protocol for decomposing word sense. Our goal is similar to that in previous sections: elicit responses from everyday speakers of the language regarding basic properties, in relation to the context of a natural language sentence.

Protocol If directly following the strategy explored thus far, we would create an interface that enumerated many dozens (or hundreds) of semantic properties one might ask about a word in context, and in further developments of Decomp there may

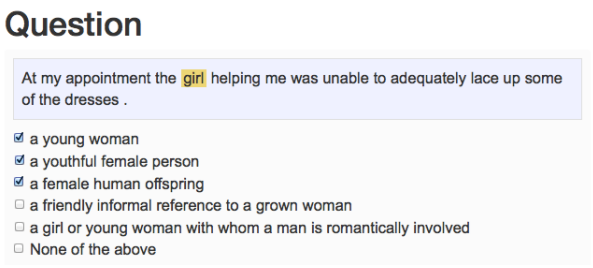


Figure 4: Example of the word sense task.

be specific properties that are deemed essential for direct querying of annotators. However, here we rely on the rich pre-existing taxonomy of lexical knowledge captured in the WordNet hierarchy (Miller, 1995) in order to more efficiently gather implicit property responses. Everyday speakers can perform basic word sense disambiguation (Snow et al., 2008): this falls under the simplicity tenant of Decomp. Once a word is disambiguated in context we then can infer automatically whether an instance is, e.g., a `physical object`.

While WordNet is a valuable resource, the selection of a specific categorical sense under an enumerated set of prespecified options is troubling in a similar way as Dowty was concerned with thematic roles (see Kilgarriff 1997). Therefore we follow a path similar to Sussna (1993) in asking annotators for zero or more senses that are appropriate.[9]

Candidate senses are extracted from WordNet *synsets*. We have grounded argument tokens in WordNet in order to make efficient use of existing lexical semantic resourses, but this protocol could in principle be used with any other lexical semantic resource. We believe these annotations will be useful already in the context of the other annotations, but in addition, future work will use these sense groundings to derive commonsense properties beyond those directly encoded in the WordNet hierarchy.

Data collection A total of 18,054 word tokens (arguments) in 10,833 total sentences extracted from EUD1.2 were annotated for sense by at least three annotators recruited from Amazon Mechanical Turk. Each token had an average of 5.63 candidates senses for annotators to choose from (Figure 4). In total, 1,065 unique annotators participated.

[9]Not only does this weaken the commitment to a single categorical meaning, but it also reduces concerns of annotators being confused by overly fine-grain definitions (Navigli, 2006).

Data validation Inter-annotator agreement was computed by lemma by taking the Jaccard index for each pair of annotators that judged the senses for that lemma: $\frac{\text{\# of senses checked by both annotators}}{\text{\# of senses checked by either annotator}}$. The overall inter-annotator agreement using this measure was 0.592: this is reasonably high considering the extremely low chance-level.

In total, 9,317 token-sense pairs were agreed upon by all annotators. We refer to these token-sense pairs as *gold word sense(s)* for the token. If we relax the agreement threshold for a token-sense pair to be gold to 0.5—i.e. half or more annotators agreed on that pair—the number of *gold word sense(s)* goes up to 27,326. Out of 18,054 individual arguments, 8,553 of them have a single *gold word sense* and 370 have two or more *gold word senses* as in the example in Figure 4. Similarly, if we relax the agreement threshold down to 0.5, 9,656 arguments have a single *gold word sense* and 17,281 arguments have two or more *gold word senses*.

7 Conclusion

We have described the *Universal Decompositional Semantics* (Decomp) project, which aims to construct and deploy a set of cross-linguistically robust semantic annotation protocols that are based in linguistic theory and that integrate seamlessly with the Universal Dependencies project. We then proposed Decomp-aligned protocols for three domains—semantic role decomposition, event decomposition, and word sense decomposition—and presented annotations, all freely available, that use these protocols and are constructed on top of the English UD v1.2 treebank. In future work, we intend to further revise and extend these protocols as well as produce novel protocols aligned with Decomp.

Acknowledgments

This research was supported by the JHU HLTCOE, DARPA DEFT, DARPA LORELEI, and NSF INSPIRE BCS-1344269. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. English web treebank. *Linguistic Data Consortium, Philadelphia, PA*, 2012.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. The Groningen Meaning Bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*. Springer, Berlin, 2017.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333, 2012.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, volume 14, pages 4585–4592, 2014.

Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, 2013.

David Dowty. *Word Meaning and Montague Grammar*. D. Reidel Publishing Company, 1979.

David Dowty. Thematic proto-roles and argument selection. *Language*, 67(3):547–619, 1991.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18, 2012.

Stephan Greene and Philip Resnik. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 503–511. Association for Computational Linguistics, 2009. ISBN 1-932432-41-8.

Joshua K. Hartshorne, Claire Bonial, and Martha Palmer. The VerbCorner Project: Toward an Empirically-Based Semantic Decomposition of Verbs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1438–1442, 2013.

Luheng He, Mike Lewis, and Luke Zettlemoyer. Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, 2015.

Ray S. Jackendoff. *Semantic Structures*. MIT Press, 1990.

Edward Kako. Thematic role properties of subjects and objects. *Cognition*, 101(1):1–42, 2006.

Adam Kilgarriff. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113, 1997.

Paul Kiparsky. Partitive case and aspect. *The Projection of Arguments: Lexical and compositional factors*, 265:307, 1998.

Manfred Krifka. Nominal reference, temporal constitution, and quantification in event semantics. In R. Bartsch, J. van Benthem, and P. von Emde Boas, editors, *Semantics and Contextual Expression*. Foris Publications, 1989.

Beth Levin and Malka Rappaport Hovav. *Argument realization*. Cambridge University Press, 2005.

Nitin Madnani, Jordan Boyd-Graber, and Philip Resnik. Measuring transitivity using untrained annotators. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 188–194, 2010.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330, June 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972470.972475`.

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`.

Roberto Navigli. Meaningful clustering of senses helps boost word sense disambiguation perfor-

mance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 105–112, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220189. URL http://www.aclweb.org/anthology/P06-1014.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. Universal Dependencies 1.2. *http://universaldependencies.github.io/docs/*, November 2015. URL https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1548.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1): 71–106, 2005.

James Pustejovsky. The generative lexicon. *Computational Linguistics*, 17(4):409–441, 1991.

M. Rappaport-Hovav and Beth Levin. Building verb meanings. In M. Butts and W. Geuder, editors, *The Projection of Arguments: Lexical and compositional factors*, pages 97–134. CSLI Publications, 1998.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. Semantic Proto-Roles. *Transactions of the Association for Computational Linguistics*, 3:475–488, 2015.

Rachel Rudinger and Benjamin Van Durme. Is the stanford dependency representation semantic? In *ACL Workshop: EVENTS*, 2014.

Roser Saurí and James Pustejovsky. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299, 2012.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D08-1027.

Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Management*, CIKM '93, pages 67–74, New York, NY, USA, 1993.

Carol Lee Tenny. *Grammaticalizing aspect and affectedness*. Thesis, Massachusetts Institute of Technology, 1987. URL http://dspace.mit.edu/handle/1721.1/14704.

Friends with Motives: Using Text to Infer Influence on SCOTUS

Yanchuan Sim
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ysim@cs.cmu.edu

Bryan R. Routledge
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213, USA
routledge@cmu.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

We present a probabilistic model of the influence of language on the behavior of the U.S. Supreme Court, specifically influence of amicus briefs on Court decisions and opinions. The approach assumes that amici are rational, utility-maximizing agents who try to win votes or affect the language of court opinions. Our model leads to improved predictions of justices' votes and perplexity of opinion language. It is amenable to inspection, allowing us to explore inferences about the persuasiveness of different amici and influenceability of different justices; these are consistent with earlier findings.

"Language is the central tool of our trade."
John G. Roberts, 2007 (Garner, 2010)

1 Introduction

The Supreme Court of the United States (SCOTUS), the highest court in the American judiciary, makes decisions with far-reaching effects. In a typical case, there are four participating parties: **petitioners** and **respondents** who file briefs arguing the merits of their sides of a case ("merits briefs"); third-party entities with an interest (but not a direct stake) in the case, who file **amicus curiae**[1] briefs to provide further arguments and recommendations on either side; and **justices** who, after oral arguments and discussions, vote on the case and write "opinions" to explain the Court's decisions.[2]

In recent years, amicus briefs are increasingly being employed as a lobbying tool to influence the Court's decision-making process (Franze and Anderson, 2015; Kearney and Merrill, 2000). The content of these briefs reveals explicit attempts to persuade justices and provides a fascinating setting for empirical study of influence through language. As such, we take the perspective of an amicus, proposing a probabilistic model of the various parties to a case that accounts for the amicus' goals.

Our model of SCOTUS is considerably more comprehensive than past work in political science, which has focused primarily on **ideal point** models that use votes as evidence. Text has been incorporated more recently as a way of making such models more interpretable, but without changing the fundamental assumptions (Lauderdale and Clark, 2014). Here, we draw on decision theory to posit amici as rational agents. We assume these amici-agents maximize their expected utility by framing their arguments to sway justices towards favorable outcomes.

We build directly on Sim et al. (2015), who used utility functions to explicitly model the goals of amici in a probabilistic setting. Their approach only considered amici in aggregate, inferring nothing about any specific amicus, such as experience or motivation for filing briefs. Here, we enrich their model to allow such analysis and also introduce Court opinions as evidence. By modeling the justices' author-

[1] *Amicus curiae* is Latin for "friends of the court." Hereafter, we use *amicus* in singular and *amici* in plural to refer to these interested third parties. It is common for several amici to co-author a single brief, which we account for in our model.

[2] We use the term *opinions* to refer to these decision-explaining documents, not to abstract positions as the term is used in much NLP research.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1724–1733,
Austin, Texas, November 1-5, 2016.

ing process as well, we can capture an important aspect of amici's goals: influencing the text of the opinions.

In §3, we demonstrate the effectiveness of our approach on vote prediction and perplexity. Furthermore, we present analyses that reveal the persuasiveness of amici and influenceability of justices that are consistent with past findings.

2 Generative Models of SCOTUS

Our approach builds on a series of probabilistic models only recently considered in NLP research. To keep the discussion self-contained, we begin with classical models of votes alone and build up toward our novel contributions.

2.1 Modeling Votes

Ideal point (IP) models are a mainstay in quantitative political science, often applied to voting records to place voters (lawmakers, justices, etc.) in a continuous space. A justice's "ideal point" is a latent variable positioning him in this space. Martin and Quinn (2002) introduced the unidimensional IP model for judicial votes, which posits an IP $\psi_j \in \mathbb{R}$ for each justice j. Often the ψ_j values are interpreted as positions along a liberal-conservative ideological spectrum. Each case i is represented by popularity (a_i) and polarity (b_i) parameters.[3] A probabilistic view of the unidimensional IP model is that justice j votes in favor of case i's petitioner (as opposed to the respondent) with probability

$$p(v_{i,j} = \text{petitioner} \mid \psi_j, a_i, b_i) = \sigma(a_i + b_i\psi_j)$$

where $\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$ is the logistic function. When the popularity parameter a_i is high enough, every justice is more likely to favor the petitioner. The polarity b_i captures the importance of a justice's ideology: polarizing cases (i.e., $|b_i| \gg 0$) push justice j more strongly to the side of the petitioner (if b_i has the same sign as ψ_j) or the respondent (otherwise).

Amici IP models. Sim et al. (2015) introduced a multidimensional IP model that incorporated text from merits and amicus briefs as evidence. They inferred dimensions of IP that are grounded in "topical" space, where topics are learned using latent Dirichlet allocation (Blei et al., 2003). In their proposed model, the merits briefs describe the issues and facts of the case, while amicus briefs were hypothesized to "frame" the facts and potentially influence the outcome of the case. For case i and justice j, the vote probability is

$$\begin{aligned} &p(v_{i,j} = \text{petitioner} \mid \boldsymbol{\psi}_j, \boldsymbol{\theta}_i, \boldsymbol{\Delta}_i, a_i, b_i, \boldsymbol{c}_i) \qquad (1) \\ &= \sigma\Big(a_i + b_i \boldsymbol{\psi}_j^\top \big(\underbrace{\boldsymbol{\theta}_i + \tfrac{1}{|\mathcal{A}_i|}\textstyle\sum_{k\in\mathcal{A}_i} c_i^{s_{i,k}} \boldsymbol{\Delta}_{i,k}}_{\text{case IP}}\big)\Big) \end{aligned}$$

where $\mathcal{A}_i$ is the set of amicus briefs filed on this case, $s_{i,k}$ denotes the side ($\in$ {petitioner, respondent}) supported by the kth brief, and c_i^{p}, and c_i^{r} are the amicus polarities for briefs on either side. The **case IP** is influenced by merits briefs (embedded in $\boldsymbol{\theta}_i$) and by the amicus briefs (embedded in $\boldsymbol{\Delta}_{i,k}$), both of which are rescaled independently by the case discrimination parameters to generate the vote probability. The model assumes that briefs on the same side share a single embedding and that individual briefs on one side influence the vote-specific IP equally.

New IP model: Persuasive amici. Lynch (2004) and others have argued that some amici are more effective than others, with greater influence on justices. We therefore propose a new model which considers amici as individual actors. Starting from Eq. 1, we consider two additional variables: each amicus e's **persuasiveness** ($\pi_e > 0$) and each justice j's **influenceability** ($\chi_j > 0$).[4]

$$\begin{aligned} &p(v_{i,j} = \text{petitioner} \mid \boldsymbol{\psi}_j, \chi_j, \boldsymbol{\theta}_i, \boldsymbol{\Delta}_i, a_i, b_i, \boldsymbol{\pi}) \qquad (2) \\ &= \sigma\Big(a_i + b_i \boldsymbol{\psi}_j^\top \big(\boldsymbol{\theta}_i + \tfrac{\chi_j}{|\mathcal{A}_i|}\textstyle\sum_{k\in\mathcal{A}_i} \bar{\pi}_{i,k} \boldsymbol{\Delta}_{i,k}\big)\Big) \end{aligned}$$

where $\bar{\pi}_{i,k} = \frac{\sum_{e\in\mathcal{E}_{i,k}} \pi_e}{|\mathcal{E}_{i,k}|}$ is the average of their π-values, with $\mathcal{E}_{i,k}$ denoting the set of entities who co-authored the kth amicus brief for case i.

Intuitively, a larger value of χ_j will shift the case IP more towards the contents of the amicus briefs,

[3] This model is also known as a two parameter logistic model in item-response theory (Fox, 2010), where a_i is "difficulty" and b_i is "discrimination."

[4] Note that the amici IP model of Sim et al. (2015), Eq. 1, is a special case of this model where $\chi_j = 1$ and each case has polarity parameters for each side; no information is shared across briefs written by the same amicus-entity for different cases.

thus making the justice seem more "influenced" by amicus. Likewise, briefs co-authored by groups of amici who are more effective (i.e., larger $\bar{\pi}_{i,k}$), will "frame" the case towards their biases. Unlike Sim et al. (2015), we eschew the amicus polarity parameters (c_i) and instead rely on the influenceability and persuasiveness parameters. Furthermore, we note that they performed a post-hoc analysis of amici influence on justices but we do so directly through χ_j.

With appropriate priors on the latent variables, the generative story for votes is:

1. For each topic $t \in \{1, \ldots, T\}$, draw topic-word distributions $\boldsymbol{\phi}_t \sim \text{Dirichlet}(\beta)$.
2. For each justice $j \in \mathcal{J}$, draw justice IP $\boldsymbol{\psi}_j \sim \mathcal{N}(\mathbf{0}, \sigma_J^2\mathbf{I} + \rho\mathbf{1})$[5] and influenceability $\chi_j \sim \log\mathcal{N}(\mathbf{0}, \sigma_I^2\mathbf{I})$.
3. For each amicus-entity $e \in \mathcal{E}$, draw its persuasiveness $\pi_e \sim \log\mathcal{N}(\mathbf{0}, \sigma_P^2\mathbf{I})$.
4. For each case $i \in \mathcal{C}$:

(a) Draw case parameters $a_i, b_i \sim \mathcal{N}(0, \sigma_C^2)$.

(b) Draw topic proportions for merits $\boldsymbol{\theta}_i \sim \text{Dirichlet}(\alpha)$.

(c) For each word $w_{i,n}^{(m)}$ in the merits briefs, draw topic indicators $z_{i,n}^{(m)} \sim \text{Categorical}(\boldsymbol{\theta}_i)$ and $w_{i,n}^{(m)} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{i,n}^{(m)}})$.

(d) For each amicus brief indexed by k:

i. Draw topic proportions $\boldsymbol{\Delta}_{i,k}$ according to a distribution discussed in §2.3.

ii. For each word $w_{i,n}^{(a)}$ in the brief, draw topic indicators $z_{i,k,n}^{(a)} \sim \text{Categorical}(\boldsymbol{\Delta}_{i,k})$ and $w_{i,k,n}^{(a)} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{i,k,n}^{(a)}})$.

(e) For each participating justice $j \in \mathcal{J}_i$, draw vote $v_{i,j}$ according to Eq. 2.

2.2 Modeling Opinions

In most SCOTUS cases, a justice is assigned to author a majority opinion, and justices voting in the majority "join" in the opinion. Justices may author additional opinions concurring or dissenting with the majority, and they may choose to join concurring and dissenting opinions written by others. Here, we extend the IP model of votes to generate the opinions of a case; this marks the second major extension beyond the IP model of Sim et al. (2015).

SCOTUS justices often incorporate language from merits (Feldman, 2016b; Feldman, 2016a) and amicus (Collins et al., 2015; Ditzler, 2011) briefs into their opinions. While amicus briefs are not usually used directly in legal analyses, the background and technical information they provide are often quoted in opinions. As such, we model opinions as a mixture of its justice-authors' topic preferences, topic proportions of the merits briefs ($\boldsymbol{\theta}$), and topic proportions of the amicus briefs ($\boldsymbol{\Delta}$). This can also be viewed as an author-topic model (Rosen-Zvi et al., 2004) where justices, litigants, and groups of amici are all effective authors. To accomplish this, we introduce an explicit switching variable x for each word, which selects between the different sources of topics, to capture the mixture proportions.

Since any justice can author additional opinions explaining the rationale behind their votes, we concatenate all opinions supporting the same side of a case into a single document.[6] However, we note that concurring opinions often contain perspectives that are different from the majority opinion and by concatenating them, we may lose some information about individual justices' styles or preferences. Building on the generative model for votes, the generative story for each case i's two opinions-documents is:

5. For each justice $j \in \mathcal{J}$, draw topics $\boldsymbol{\Gamma}_j \sim \text{Dirichlet}(\alpha)$.
6. For each case $i \in \mathcal{C}$:

(a) For each side $s \in \{\text{petitioner}, \text{respondent}\}$, draw "author"-mixing proportions:

$$\boldsymbol{\tau}_i^s \sim \text{Dirichlet}\left(\begin{bmatrix} p(v_{i,1} = s) \\ \vdots \\ p(v_{i,|\mathcal{J}|} = s) \\ 1 \\ 1 \end{bmatrix}\right) \quad (3)$$

where the last two dimensions are for choosing topics from the merits and amicus briefs, re-

[5] The positive off-diagonal elements of the covariance matrix for justice IPs (ψ_j) orient the issue-specific dimensions in the same direction (i.e., with conservatives at the same end) and provide shrinkage of IP in each dimension to their common mean across dimensions (Lauderdale and Clark, 2014).

[6] Opinions where justices dissent from the majority are concatenated together, and those where justices concur with the majority are concatenated with the majority opinion.

spectively.[7] Intuitively, our model assumes that opinions will incorporate more language from justices who agree with it.

(b) For each side $s \in \{\text{petitioner}, \text{respondent}\}$ and each word $w^{(o)}_{i,s,n}$ in the opinion for side s,

i. Draw $x_{i,s,n} \sim \text{Categorical}(\boldsymbol{\tau}^s_i)$.

ii. If $x_{i,s,n} \in \mathcal{J}_i$, draw $z^{(o)}_{i,s,n} \sim \text{Categorical}(\boldsymbol{\Gamma}_{x_{i,s,n}})$, the justice's topic distribution.

iii. If $x_{i,s,n} = \text{merits}$, draw $z^{(o)}_{i,s,n} \sim \text{Categorical}(\boldsymbol{\theta}_i)$, the merits topic distribution.

iv. If $x_{i,s,n} = \text{amici}$, draw $z^{(o)}_{i,s,n} \sim \text{Categorical}(\boldsymbol{\Delta}^s_i)$, side s's amicus briefs topic distribution.

v. Draw word $w^{(o)}_{i,s,n} \sim \text{Categorical}(\boldsymbol{\phi}_{z^{(o)}_{i,s,n}})$.

Unlike in the Court, where an opinion is mainly authored by a single justice, all the participating justices contribute to an opinion in our generative story, with different proportions. This approach simplifies the computational model and reflects the closed-door nature of discussions held by justices prior to writing their opinions. Our model assumes that justices debate together, and that the arguments are reflected in the final opinions. In future work, we might extend the model to infer an authoring process that separates an initial author from "joiners."

2.3 Amici Utility

Our approach assumes that amici are rational and purposeful decisionmakers who write briefs to influence the outcome of a case; this assumption leads to the design of the distribution over $\boldsymbol{\Delta}$ (generative model step 4(d)i). When writing a brief $\boldsymbol{\Delta}$, an amicus seeks to increase the response to her brief (i.e., votes), while keeping her costs low. We encode her objectives as a utility function, which she aims to maximize with respect to the decision variable $\boldsymbol{\Delta}$:

$$U(\boldsymbol{\Delta}) = R(\boldsymbol{\Delta}) - C(\boldsymbol{\Delta}) \tag{4}$$

where $R(\cdot)$ is the extrinsic response (reward) that an amicus gets from filing brief $\boldsymbol{\Delta}$ and $C(\cdot)$ is the "cost" of filing the brief; dependency on other latent variables is notationally suppressed. When authoring her brief, we assume that the amicus writer has knowledge of the justices (IP and topic preferences), case parameters, and merits, but not the other amici participating in the case.[8]

Amicus curiae are motivated to position themselves (through their briefs) in such a way as to improve the likelihood that their arguments will persuade SCOTUS justices. This is reflected in the way a justice votes or through the language of the opinions. Hence, we investigate two response functions. First, an amicus supporting side s seeks to win votes for s,

$$R^{\text{vote}}(\boldsymbol{\Delta}) = \tfrac{1}{|\mathcal{J}|} \textstyle\sum_{j \in \mathcal{J}} p(v_j = s \mid \ldots), \tag{5}$$

which is the expected number of votes for side s, under the model. This follows Sim et al. (2015).

An alternative is to maximize the (topical) similarity between her brief and the Court's opinion(s) siding with s,

$$R^{\text{opinion}}(\boldsymbol{\Delta}) = 1 - H^2(\boldsymbol{\Delta}, \boldsymbol{\Omega}^s), \tag{6}$$

where $H^2(P, Q) = \frac{1}{2}\|\sqrt{P} - \sqrt{Q}\|_2^2$ is the squared Hellinger (1909) distance between two distributions, and $\boldsymbol{\Omega}^s$ is the expected topic mixture under the model assumptions in §2.2 (which has a closed form). In short, the amicus gains utility by accurately predicting the expected opinion, thereby gaining publicity and demonstrating to members, donors, potential clients, and others that the language of the highly visible SCOTUS opinion was influenced. Both Eqs. 5 and 6 reward amici when justices "agree" with them, for different definitions of agreement.

We assume the cost $C(\boldsymbol{\Delta}) = H^2(\boldsymbol{\Delta}, \boldsymbol{\theta})$, the squared Hellinger distance between the mixture proportions of the amicus brief and merits briefs.[9] The cost term defines the "budget" set of the amicus: briefs cannot be arbitrary text, as there is disutility or

[7] In cases where there are less than nine justices voting, the size of $\boldsymbol{\tau}^{\text{p}}_i$ and $\boldsymbol{\tau}^{\text{r}}_i$ may be smaller.

[8] Capturing strategic amici agents (a petitioner amicus choosing brief topics considering a respondent amicus' brief) would require a game-theoretic model and, we conjecture, would require a much richer representation of policy and goals. That idea is left for future research.

[9] Sim et al. (2015) used a Euclidean distance for cost rather than Hellinger distance, which we believe is a better fit for probability distributions without sacrificing symmetry (cf. KL divergence).

effort required to carefully frame a case, and monetary cost to hiring legal counsel. The key assumption is that framing is costly, while simply matching the merits is cheap (and presumably unnecessary).

Notationally, we use U^{vote} to refer to models where Eq. 5 is in the utility function (in Eq. 4) and U^{opinion} where it is Eq. 6.

Random utility models Recall our assumption that amici are purposeful writers whose briefs are optimized for their utility function. In an ideal setting, the $\boldsymbol{\Delta}$ which we observe will be utility maximizing. We simplify computation by assuming that these amici agents' preferences also contain an idiosyncratic random component that is unobserved to us. This is a common assumption in discrete choice models known as a "random utility model" (McFadden, 1974). We view the utility function as a prior on $\boldsymbol{\Delta}$,

$$p_{\text{util}}(\boldsymbol{\Delta} \mid \ldots) \propto \exp \eta U(\boldsymbol{\Delta}),$$

where our functional equations for utility imply $-1 \leq U(\cdot) \leq 1$. η is a hyperparameter tuned using cross validation. The behavior which we observe (i.e., the amicus' topic mixture proportions) has a likelihood that is proportional to utility.

2.4 Parameter Estimation

The models we described above can be estimated within a Bayesian framework. We decoupled the estimation of the votes model from the opinions model; we first estimate the parameters for the votes model and hold them fixed while we estimate the new latent variables in the opinions model. In our preliminary experiments, we found that estimating parameters for both votes and opinions jointly led to slow mixing and poor predictive performance. Separating the estimation procedure into two stages allows the model to find better parameters for the votes model, which are then fed into the opinions model as priors through the vote probabilities.

We used Metropolis within Gibbs, a hybrid MCMC algorithm, to sample the latent parameters from their posterior distributions (Tierney, 1994).[10] For the Metropolis-Hastings proposal distributions, we used a Gaussian for the case parameters a, b, and justice IPs $\boldsymbol{\psi}$, log-normal distributions for χ and π, and logistic-normal distribution for the variables on the simplex $\boldsymbol{\theta}, \boldsymbol{\Delta}, \boldsymbol{\tau}$, and $\boldsymbol{\Gamma}$. We tuned the hyperparameters of the proposal distributions at each iteration to achieve a target acceptance rate of 15–45%. We used $T = 128$ topics for model and initialized topic proportions $(\boldsymbol{\theta}, \boldsymbol{\Delta})$ and topic-word distributions $(\boldsymbol{\phi})$ using online LDA (Hoffman et al., 2010).

[10] The details of our sampler and hyperparameter settings can be found in §A and §B of the supplementary materials.

3 Experiments

Data. In our experiments, we use SCOTUS cases between 1985–2014; votes and metadata are from Spaeth et al. (2013) and brief texts come from Sim et al. (2015). We concatenate each of the 2,643 cases' merits briefs from both parties to form a single document, where the text is used to infer the representation of the case in topical space ($\boldsymbol{\theta}$; i.e., merits briefs are treated as "facts of the case"). Likewise, opinions supporting the same side of the case (i.e., majority and concurring vs. dissents) were concatenated to form a single document. In our dataset, the opinions are explicitly labeled with the justice who authored them (as well as other justices who decide to "join" it).

As the amicus briefs in the dataset were not explicitly labeled with the side that they support, Sim et al. (2015) built a binary classifier with bag-of-n-gram features that took advantage of cues in the brief content that *strongly* signal the side that the amici supports (e.g., "in support of petitioner"). We used their classifier to label the amici's supporting side. Additionally, we created regular expression rules to identify and standardize amicus authors from the header of briefs. We filtered amici who have participated in fewer than 5 briefs[11] and merged regional chapters of amicus organizations together (i.e., "ACLU of Kansas" and "ACLU of Kentucky" are both labeled "ACLU"). On the other hand, we separated labeled amicus briefs by the U.S. Solicitor General according to the presidential administration when the brief is filed (i.e., an amicus brief filed during Obama's administration will be labeled "USSG-Obama"). The top three amici by number of briefs filed are *American Civil Liberties Union* (463), *Utah* (376), and *National Asso-*

[11] Briefs which have no authors as a result of the filtering process are removed from our dataset. This occurred in about 24% of amicus briefs.

Cases / Votes	2,643 / 23,465
Merits / Amicus briefs	16,416 / 16,303
Opinions	4,187
Phrases	18,207,326

Table 1: Corpus statistics.

ciation of Criminal Defense Lawyers (359).

We represent a document as a bag of n-grams with part of speech tags that follow the simple but effective pattern (Adjective|Cardinal|Noun)+ Noun (Justeson and Katz, 1995). We filter phrases appearing fewer than 100 times or in more than 8,500 documents, obtaining a final set of 48,589 phrase types. Table 1 summarizes the details of our corpus.

Predicting Votes. We quantify the performance of our vote model using 5-fold cross validation and on predicting future votes from past votes. The utility function in the vote model uses the response function in Eq. 5. Due to the specification of IP models, we need the case parameters of new cases to predict the direction of the votes. Gerrish and Blei (2011) accomplished this by using regression on legislative text to predict the case parameters (a, b). Here, we follow a similar approach, fitting ridge regression models on the merits brief topic mixtures $\boldsymbol{\theta}$ to predict a and b for each case.[12] On the held-out test cases, we sampled the mixture proportions for the merits and amicus briefs directly using latent Dirichlet allocation with parameters learned while fitting our vote model. With the parameters from our fitted vote model and ridge regression, we can predict the votes of every justice for every case.

We compared the performance of our model with two strong baselines: (i) a random forest trained on case-centric metadata coded by Spaeth et al. (2013) to make predictions on how justices would vote (Katz et al., 2014) and (ii) Sim et al. (2015)'s amici IP model, which uses amicus briefs and their version of utility; it is a simpler version of our vote model that does not consider the persuasiveness of different amici or the influenceability of different justices. For prediction in Sim et al. (2015), we used the same approach described above to estimate the case parameters a, b, and regressing on amicus brief topics ($\boldsymbol{\Delta}$) instead for amicus polarities c^{p} and c^{r}. Table 2 shows performance on vote prediction.

[12] We tuned the parameters of the regression using 5-fold cross-validation on the training data.

Model	5-fold	2013	2014
Most frequent	0.597	**0.694**	0.650
Random forest	0.651	0.648	0.633
Vote model without U^{vote}	0.661	0.655	0.660
Sim et al. (2015)	0.675	0.658	0.661
Vote model with U^{vote}	**0.685**	0.664	**0.672**

Table 2: Accuracy of vote prediction. There are 70 cases (625 votes) and 69 cases (619 votes) in the 2013 and 2014 test sets, respectively.

We evaluated the models using 5-fold cross validation, as well as on forecasting votes in 2013 and 2014 (trained using data from 1985 to the preceding year). Our model outperformed the baseline models. The improvement in accuracy over Sim et al. (2015) is small; most likely because both models are very similar, the main difference being the parametrization of amicus briefs. In the 2013 test set, the distribution of votes is significantly skewed towards the petitioner (compared to the training data), which resulted in the most frequent class classifier performing much better than everything else. Fig. 1 illustrates our model's estimated ideal points for selected topics.

Predicting Opinions. We also estimated the opinion model using the utility function with response function in Eq. 6. We use perplexity as a proxy to measure the opinion content predictive ability of our model. Perplexity on a test set is commonly used to quantify the generalization ability of probabilistic models and make comparisons among models over the same observation space. For a case with opinion $\boldsymbol{w}$ supporting side s, the perplexity is defined as

$$\exp\left(-\frac{\log p(\boldsymbol{w} \mid s, \ldots)}{N}\right),$$

where N is the number of tokens in the opinion and a lower perplexity indicates better generalization performance. The likelihood term can be approximated using samples from the inference step.

Table 3 shows the perplexity of our model on opinions in the test set. As described in §2.4, we learn the vote model in the first stage before estimating the opinion model. Here, we compare our model against using vote models that do not include U^{vote} to evaluate the sensitivity of our opinion

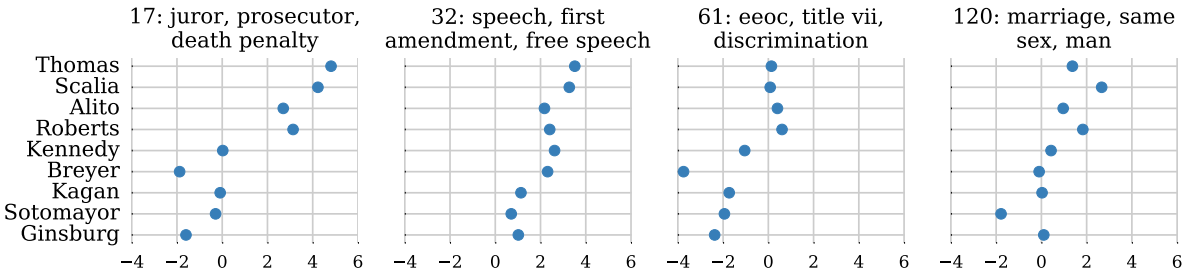


Figure 1: Justices' ideal points for selected topics. Justices whose topic IPs are close to each other are more likely to vote in the same direction on cases involving those topics. The IP estimated by our model is consistent with publicly available knowledge regarding justices' ideological stances on these issues.

model to the vote model parameters. Additionally, we compared against two baselines trained on just the opinions: one using LDA[13] and another using the author-topic model (Rosen-Zvi et al., 2004). For the author-topic model, we treat each opinion as being "authored" by the participating justices, a pseudo-author representing the litigants which is shared between opinions in a case, and a unique amicus author for each side. Our model with U^{opinion} achieves better generalization performance than the simpler baselines, while we do not see significant differences in whether the first stage vote models use U^{vote}. This is not surprising since the vote model's results are similar with or without U^{vote} and it influences the opinion model indirectly through priors and U^{opinion}.

In our model, the latent variable $\mathbf{\Gamma}_j$ captures the proportion of topics that justice j is likely to contribute to an opinion. When j has a high probability of voting for a particular side, our informed prior increases the likelihood that j's topics will be selected for words in the opinion. While $\mathbf{\Gamma}_j$ serves a similar purpose to $\boldsymbol{\psi}_j$ in characterizing j through her ideological positions, $\boldsymbol{\psi}_j$ relies on votes and gives us a "direction" of j's ideological standing, whereas $\mathbf{\Gamma}_j$ is estimated from text produced by the justices and only gives us the "magnitude" of her tendency to author on a particular issue. In Table 4, we identify the top topics in $\mathbf{\Gamma}_j$ by considering the deviation from the mean of all justice's $\mathbf{\Gamma}$, i.e., $\Gamma_{j,k} - \frac{1}{|\mathcal{J}|}\sum_j \Gamma_{j,k}$.

Amici Persuasiveness. The latent variable π_e captures the model's belief about amicus e's brief's effect on the case IP, which we call "persuasiveness." A large π_e indicates that across the dataset, e exerts a larger effect on the case IPs, that is, according to our model, she has a larger impact on the Court's decision than other amici. Fig. 2 is a swarm plot illustrating the distribution of π values for different types of amicus writers.

Model	5-fold	2013	2014
LDA	2.86	2.67	2.63
Author-Topic	2.62	2.36	2.25
Opinion model without U^{opinion}	†2.43	†2.26	†2.13
	2.45	2.27	2.11
Opinion model with U^{opinion}	†2.10	†**1.91**	†1.96
	2.07	1.98	**1.94**

Table 3: Perplexity of Court's opinions ($\times 10^3$). There are 30,133 phrases (98 opinions) and 23,706 phrases (109 opinions) in the 2013 and 2014 test set, respectively. Results marked † are initialized with a vote model U^{vote}.

Our model infers that governmental offices tend to have larger π values than private organizations, especially the U.S. Solicitor General.[14] In fact, Lynch (2004) found through interviews with SCOTUS law clerks that "amicus briefs from the solicitor general are 'head and shoulders' above the rest, and are often considered more carefully than party briefs."

Another interesting observation from Fig. 2 is the low π value for ACLU and ABA, despite being prolific amicus brief filers. While it is tempting to say that amici with low π values are ineffective, we find that there is almost no correlation between π and the proportion of cases where they were on the winning side.[15] Note that our model does not assume that a

[13] We used `scikit-learn`'s LDA module (Pedregosa et al., 2011) which implements the online variational Bayes algorithm (Hoffman et al., 2010).

[14] The average π for *Federal*, *State/Local* and *Others* are 2.35, 1.11, and 0.929 respectively.

[15] The Spearman's ρ between π and the proportion of winning

John G. Roberts
32: speech, first amendment, free speech, message, expression 61: eeoc, title vii, discrimination, woman, civil rights act 52: sec, fraud, security, investor, section ##b
Ruth B. Ginsburg
61: eeoc, title vii, discrimination, woman, civil rights act 80: class, settlement, rule ##, class action, r civ 96: taxpayer, bank, corporation, fund, irs
Antonin Scalia
94: 42 USC 1983, qualified immunity, immunity, official, section #### 57: president, senate, executive, article, framer 80: class, settlement, rule ##, class action, r civ

Table 4: Top three topics contributed to Court opinions for selected justices ($\mathbf{\Gamma}$). The full list can be found in supplementary §C.

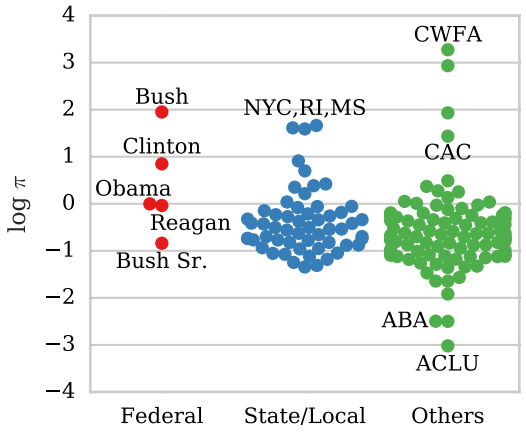


Figure 2: Amici "persuasiveness" by organization type. *Federal* refers to different presidential administration's federal government (and represented by the U.S. Solicitor General) and *State/Local* refers to state and local governments. The abbreviated amici are New York City (NYC), Rhode Island (RI), Mississippi (MS), Concerned Women For America (CWFA), Constitution Accountability Center (CAC), American Bar Association (ABA), and American Civil Liberties Union (ACLU).

"persuasive" amicus tends to win. Instead, an amicus with large π will impact the case IP most, and thus explain a justice's vote or opinion (even dissenting) more than the other components in a case.

Insofar as π explains a vote, we must exercise caution; it is possible that the amicus played no role in the decision-making process and the values of π_e simply reflect our modeling assumptions and/or artifacts of the data. Without entering the minds of SCOTUS justices, or at least observing their closed-door deliberations, it is difficult to measure the influence of amicus briefs on justices' decisions.

Justice Influenceability. The latent variable χ_j measures the relative effect of amicus briefs on justice j's vote IP; when χ_j is large, justice j's vote probability is affected by amicus briefs more. Since χ_j is shared between all cases that a justice participates in, χ_j should correspond to how much they value amicus briefs. Some justices, such as the late Scalia, are known to be dubious of amicus briefs, preferring to leave the task of reading these briefs to their law clerks, who will pick out any notable briefs for them; we would expect Scalia to have a smaller χ than other justices. In Table 5, we compare the χ values of justices with how often they cite an amicus brief in any opinion they wrote (Franze and Anderson, 2015). The χ values estimated by our model are consistent with our expectations.[16]

We note that the χ values correlate considerably with the general ideological leanings of the justices. This might be a coincidence or an inability of the model's specification to discern between ideological extremeness and influenceability.

4 Related Work

The ideal points model was first introduced by Poole and Rosenthal (1985) and has inspired a variety of IP models in SCOTUS (Lauderdale and Clark, 2014; Martin and Quinn, 2002) and Congressional bills (Clinton et al., 2004; Gerrish and Blei, 2011; Heckman and Snyder, 1996). IP has provided a useful framework to characterize voters using roll call information and textual evidence.

We view amicus briefs as "purposeful" texts, where authors are writing to maximize their utility function. This is related to work investigating news media for "slant" to maximize profit (Gentzkow and Shapiro, 2010) and economists choosing research topics maximize certain career outcomes (Jelveh et al., 2015). More generally, extensive literature in

sides is -0.0549. On average, an amicus supports the winning side in 55% of cases. For the ACLU, ABA, CAC, and CWFA, the proportions are 44%, 50%, 47%, and 50% respectively.

[16]The Spearman's ρ between χ_j and citation rates is 0.678.

Justice	χ_j	Citation rate (%)
Sonia Sotomayor	1.590	45
Elena Kagan	0.714	40
Stephen G. Breyer	0.637	38
Ruth B. Ginsburg	0.515	41
John G. Roberts	0.495	42
Anthony M. Kennedy	0.468	42
Samuel A. Alito	0.286	27
Antonin Scalia	0.268	22
Clarence Thomas	0.162	25

Table 5: Justice χ values and their average amicus citation rates between 2010–2015, provided by Franze and Anderson (2015).

econometrics estimates structural utility-based decisions (Berry et al., 1995, *inter alia*).

Researchers have used SCOTUS texts to study authorship (Li et al., 2013), historical changes (Wang et al., 2012), power relationships (Danescu-Niculescu-Mizil et al., 2012; Prabhakaran et al., 2013), and pragmatics (Goldwasser and Daumé, 2014).

5 Conclusion

We presented a random utility model of the Supreme Court that is more comprehensive than earlier work. We considered an individual amicus' persuasiveness and motivations through two different utility functions. On the vote prediction task, our results are consistent with earlier work, and we can infer and compare the relative effectiveness of an individual amicus. Moreover, our opinions model and opinion utility function achieved better generalization performance than simpler methods.

Acknowledgments

The authors thank the anonymous reviewers for their thoughtful feedback and Tom Clark, Philip Resnik, and members of the ARK group for their valuable comments. This research was supported in part by an A*STAR fellowship to Y. Sim, by a Google research award, and by computing resources from the Pittsburgh Supercomputing Center.

References

Steven Berry, James Levinsohn, and Ariel Pakes. 1995. Automobile prices in market equilibrium. *Econometrica: Journal of the Econometric Society*, pages 841–890.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. The statistical analysis of roll call data. *American Political Science Review*, 98:355–370.

Paul M. Collins, Pamela C. Corley, and Jesse Hamner. 2015. The influence of amicus curiae briefs on U.S. Supreme Court opinion content. *Law & Society Review*, 49(4):917–944.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proc. of WWW*.

Megan Ann Ditzler. 2011. Language overlap between solicitor general amicus curiae and Supreme Court majority opinions: An analysis. Master's thesis, Southern Illinois University Carbondale.

Adam Feldman. 2016a. All copying is not created equal: Examining Supreme Court opinions' borrowed language. *Journal of Appellate Practice and Process*, 17.

Adam Feldman. 2016b. A brief assessment of Supreme Court opinion language, 1946–2013. *Mississippi Law Journal*, 85.

J. P. Fox. 2010. *Bayesian Item Response Modeling: Theory and Applications*. Statistics for Social and Behavioral Sciences. Springer-Verlag New York.

Anthony J. Franze and R. Reeves Anderson. 2015. Record breaking term for amicus curiae in Supreme Court reflects new norm. *National Law Journal*, Supreme Court Brief. `http://www.nationallawjournal.com/supremecourtbrief/id=1202735095655/`, August 19, 2015.

Bryan A. Garner. 2010. Interviews with United States Supreme Court justices. In Joseph Kimble, editor, *The Scribes Journal of Legal Writing*, volume 13. American Society of Legal Writers.

Matthew Gentzkow and Jesse M Shapiro. 2010. What drives media slant? Evidence from U.S. daily newspapers. *Econometrica*, 78(1):35–71.

Sean Gerrish and David Blei. 2011. Predicting legislative roll calls from text. In *Proc. of ICML*.

Dan Goldwasser and Hal Daumé. 2014. "I object!" modeling latent pragmatic effects in courtroom dialogues. In *Proc. of EACL*.

James J. Heckman and James M. Snyder. 1996. Linear probability models of the demand for attributes with an empirical application to estimating the preferences of legislators. Working Paper 5785, National Bureau of Economic Research.

Ernst D. Hellinger. 1909. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik (Crelle's Journal)*, 1909(136):210–271.

Matthew Hoffman, Francis R. Bach, and David M. Blei. 2010. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 23*.

Zubin Jelveh, Bruce Kogut, and Suresh Naidu. 2015. Political language in economics. *Columbia Business School Research Paper Series*, 14(57).

John S. Justeson and Slava M. Katz. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27.

Daniel Martin Katz, Michael James Bommarito, and Josh Blackman. 2014. Predicting the behavior of the Supreme Court of the United States: A general approach. `http://ssrn.com/abstract=2463244`.

Joseph D. Kearney and Thomas W. Merrill. 2000. The influence of amicus curiae briefs on the Supreme Court. *University of Pennsylvania Law Review*, pages 743–855.

Benjamin E. Lauderdale and Tom S. Clark. 2014. Scaling politically meaningful dimensions using texts and votes. *American Journal of Political Science*, 58(3):754–771.

William Li, Pablo Azar, David Larochelle, Phil Hill, James Cox, Robert C. Berwick, and Andrew W. Lo. 2013. Using algorithmic attribution techniques to determine authorship in unsigned judicial opinions. *Stanford Technology Law Review*, pages 503–534.

Kelly J. Lynch. 2004. Best friends – Supreme Court law clerks on effective amicus curiae briefs. *Journal of Law & Politics*, 20.

Andrew D. Martin and Kevin M. Quinn. 2002. Dynamic ideal point estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 19531999. *Political Analysis*, 10(2):134–153.

Daniel McFadden. 1974. Conditional logit analysis of qualitative choice behavior. In Paul Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. Available at `http://scikit-learn.org/`.

Keith T. Poole and Howard Rosenthal. 1985. A spatial model for legislative roll call analysis. *American Journal of Political Science*, 29(2):357–384.

Vinodkumar Prabhakaran, Ajita John, and Dorée D. Seligmann. 2013. Who had the upper hand? ranking participants of interactions based on their relative power. In *Proc. of IJCNLP*.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proc. of UAI*.

Yanchuan Sim, Bryan Routledge, and Noah A. Smith. 2015. The utility of text: The case of amicus briefs and the Supreme Court. In *Proc. of AAAI*.

Harold J. Spaeth, Sara Benesh, Lee Epstein, Andrew D. Martin, Jeffrey A. Segal, and Theodore J. Ruger. 2013. Supreme Court Database, Version 2013 Release 01. Database at `http://supremecourtdatabase.org`.

Luke Tierney. 1994. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728.

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. 2012. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proc. of ACL*.

Verb Phrase Ellipsis Resolution Using Discriminative and Margin-Infused Algorithms

Kian Kenyon-Dean **Jackie Chi Kit Cheung** **Doina Precup**
School of Computer Science
McGill University
kian.kenyon-dean@mail.mcgill.ca, {jcheung,dprecup}@cs.mcgill.ca

Abstract

Verb Phrase Ellipsis (VPE) is an anaphoric construction in which a verb phrase has been elided. It occurs frequently in dialogue and informal conversational settings, but despite its evident impact on event coreference resolution and extraction, there has been relatively little work on computational methods for identifying and resolving VPE. Here, we present a novel approach to detecting and resolving VPE by using supervised discriminative machine learning techniques trained on features extracted from an automatically parsed, publicly available dataset. Our approach yields state-of-the-art results for VPE detection by improving F1 score by over 11%; additionally, we explore an approach to antecedent identification that uses the Margin-Infused-Relaxed-Algorithm, which shows promising results.

1 Introduction

Verb Phrase Ellipsis (VPE) is an anaphoric construction in which a verbal constituent has been omitted. In English, an instance of VPE consists of two parts: a **trigger**, typically an auxiliary or modal verb, that indicates the presence of a VPE; and an **antecedent**, which is the verb phrase to which the elided element resolves (Bos and Spenader, 2011; Dalrymple et al., 1991). For example, in the sentence, *"The government includes money spent on residential renovation; Dodge does not"*, the trigger *"does"* resolves to the antecedent *"includes money spent on residential renovation"*.

The ability to perform VPE resolution is important for tasks involving event extraction, especially in conversational genres such as informal dialogue where VPE occurs more frequently (Nielsen, 2005). Most current event extraction systems ignore VPE and derive some structured semantic representation by reading information from a shallow dependency parse of a sentence. Such an approach would not only miss many valid links between an elided verb and its arguments, it could also produce nonsensical extractions if applied directly on an auxiliary trigger. In the example above, a naive approach might produce an unhelpful semantic triple such as $(\textit{Dodge}, \textit{agent}, \textit{do})$.

There have been several previous empirical studies of VPE (Hardt, 1997; Nielsen, 2005; Bos and Spenader, 2011; Bos, 2012; Liu et al., 2016). Many previous approaches were restricted to solving specific subclasses of VPE (e.g., VPE triggered by *do* (Bos, 2012)), or have relied on simple heuristics for some or all of the steps in VPE resolution, such as by picking the most recent previous clause as the antecedent.

In this paper, we develop a VPE resolution pipeline which encompasses a broad class of VPEs (Figure 1), decomposed into the following two steps. In the *VPE detection* step, the goal is to determine whether or not a word triggers VPE. The second step, *antecedent identification*, requires selecting the clause containing the verbal antecedent, as well as determining the exact boundaries of the antecedent, which are often difficult to define.

Our contribution is to combine the rich linguistic analysis of earlier work with modern statistical approaches adapted to the structure of the VPE resolution problem. First, inspired by earlier work,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1734–1743,
Austin, Texas, November 1-5, 2016.

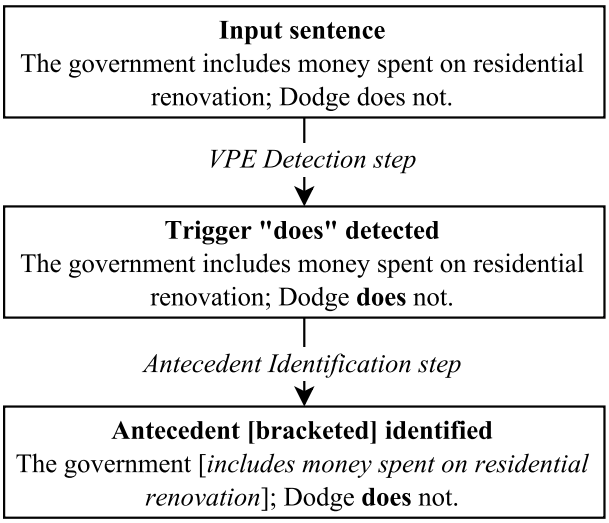


Figure 1: Example of the VPE resolution pipeline on an example found in WSJ file wsj_0036.

our system exploits linguistically informed features specific to VPE in addition to standard features such as lexical features or POS tags. Second, we adapt the Margin-Infused-Relaxed-Algorithm (MIRA) (Crammer et al., 2006), which has been popular in other tasks, such as machine translation (Watanabe et al., 2007) and parsing (McDonald et al., 2005), to antecedent identification. This algorithm admits a partial loss function which allows candidate solutions to overlap to a large degree. This makes it well suited to antecedent identification, as candidate antecedents can overlap greatly as well.

On *VPE detection*, we show that our approach significantly improves upon a deterministic rule-based baseline and outperforms the state-of-the-art system of Liu et al. (2016) by 11%, from 69.52% to 80.78%. For *antecedent identification* we present results that are competitive with the state-of-the-art (Liu et al., 2016). We also present state-of-the-art results with our end-to-end VPE resolution pipeline. Finally, we perform feature ablation experiments to analyze the impact of various categories of features.

2 Related Work

VPE has been the subject of much work in theoretical linguistics (Sag, 1976; Dalrymple et al., 1991, *inter alia*). VPE resolution could have a significant impact on related problems such as event coreference resolution (Lee et al., 2012; Bejan and Harabagiu, 2010; Liu et al., 2014) and event extraction (Ahn, 2006; Kim et al., 2009; Ritter et al., 2012). It has, however, received relatively little attention in the computational literature.

Hardt (1992) engaged in the first study of computational and algorithmic approaches for VPE detection and antecedent identification by using heuristic, linguistically motivated rules. Hardt (1997) extracted a dataset of 260 examples from the WSJ corpus by using an algorithm that exploited null elements in the PTB parse trees. Nielsen (2005) built a dataset that combined sections of the WSJ and BNC; he showed that the more informal settings captured in the BNC corpora show significantly more frequent occurrences of VPE, especially in dialogue excerpts from interviews and plays. Using this dataset, he created a full VPE pipeline from raw input text to a full resolution by replacing the trigger with the intended antecedent[1].

Bos and Spenader (2011) annotated the WSJ for occurrences of VPE. They found over 480 instances of VPE, and 67 instances of the similar phenomenon of *do-so* anaphora. Bos (2012) studied *do*-VPE by testing algorithmic approaches to VPE detection and antecedent identification that utilize Discourse Representation Theory.

Concurrently with the present work, Liu et al. (2016) explored various decompositions of VPE resolution into detection and antecedent identification subtasks, and they corrected the BNC annotations created by Nielsen (2005), which were difficult to use because they depended on a particular set of preprocessing tools. Our work follows a similar pipelined statistical approach. However, we explore an expanded set of linguistically motivated features and machine learning algorithms adapted for each subtask. Additionally, we consider all forms of VPE, including *to*-VPE, whereas Liu et al. only consider modal or light verbs (be, do, have) as candidates for triggering VPE. This represented about 7%

[1] e.g., the resolution of the example in Figure 1 would be "The government includes money spent on residential renovation; Dodge does not [include money spent on residential renovation]". We did not pursue this final step due to the lack of a complete dataset that explicitly depicts the correct grammatical resolution of the VPE.

Auxiliary Type	Example	Frequency
Do	does, done	214 (39%)
Be	is, were	108 (19%)
Have	has, had	44 (8%)
Modal	will, can	93 (17%)
To	to	29 (5%)
So	do so/same[3]	67 (12%)
TOTAL		554

Table 1: Auxiliary categories for VPE and their frequencies in all 25 sections of the WSJ.

of the dataset that they examined.

3 Approach and Data

We divide the problem into two separate tasks: VPE detection (Section 4), and antecedent identification (Section 5). Our experiments use the entire dataset presented in (Bos and Spenader, 2011). For preprocessing, we used CoreNLP (Manning et al., 2014) to automatically parse the raw text of WSJ for feature extraction. We also ran experiments using gold-standard parses; however, we did not find significant differences in our results[2]. Thus, we only report results on automatically generated parses.

We divide auxiliaries into the six different categories shown in Table 1, which will be relevant for our feature extraction and model training process, as we will describe. This division is motivated by the fact that different auxiliaries exhibit different behaviours (Bos and Spenader, 2011). The results we present on the different auxiliary categories (see Tables 2 and 4) are obtained from training a single classifier over the entire dataset and then testing on auxiliaries from each category, with the *ALL* result being the accuracy obtained over all of the test data.

[2]An anonymous reviewer recommended that further experiments could be performed by using the more informative NPs created with NML nodes (Vadas and Curran, 2007) on the gold-standard parsed WSJ.

[3]For example, "John will **go to the store** and Mary will *do the same/likewise/the opposite*". *Do X* anaphora and modals are not technically auxiliary verbs, as noted by the annotators of our dataset (Bos and Spenader, 2011), but for the purposes of this study we generalize them all as auxiliaries while simultaneously dividing them into their correct lexical categories.

4 VPE Detection

The task of VPE detection is structured as a binary classification problem. Given an auxiliary, a, we extract a feature vector f, which is used to predict whether or not the auxiliary is a trigger for VPE. In Figure 1, for example, there is only one auxiliary present, "does", and it is a trigger for VPE. In our experiments, we used a logistic regression classifier.

4.1 Feature Extraction

We created three different sets of features related to the auxiliary and its surrounding context.

Auxiliary. Auxiliary features describe the characteristics of the specific auxiliary, including the following:

- ◇ word identity of the auxiliary
- ◇ lemma of the auxiliary
- ◇ auxiliary type (as shown in Table 1)

Lexical. These features represent:

- ◇ the three words before and after the trigger
- ◇ their part-of-speech (POS) tags
- ◇ their POS bigrams

Syntactic. We devise these features to encode the relationship between the candidate auxiliary and its local syntactic context. These features were determined to be useful through heuristic analysis of VPE instances in a development set. The feature set includes the following binary indicator features (a = the auxiliary):

- ◇ a c-commands[4] a verb
- ◇ a c-commands a verb that comes after it
- ◇ a verb c-commands a
- ◇ a verb *locally*[5] c-commands a
- ◇ a locally c-commands a verb
- ◇ a is c-commanded by "than", "as", or "so"
- ◇ a is preceded by "than", "as", or "so"
- ◇ a is next to punctuation
- ◇ the word "to" precedes a
- ◇ a verb immediately follows a
- ◇ a is followed by "too" or "the same"

[4]A word *A* c-commands another word *B* if *A*'s nearest branching ancestor in the parse tree is an ancestor of B, following the definition of Carnie (2013). We use this term purely to define a syntactic relation between two points in a parse tree.

[5]A word *A* and word *B* share a local structure if they have the same closest S-node ancestor in the parse tree.

4.2 Baseline

As a baseline, we created a rule-based system inspired by Nielsen's (2005) approach to solving VPE detection. The baseline algorithm required significant experimental tuning on the development set because different linguistically hand-crafted rules were needed for each of the six trigger forms. For example, the following rule for *modals* achieved 80% F1-accuracy (see Table 2): "assume VPE is occurring if the modal does not c-command a verb that follows it". The other trigger forms, however, required several layers of linguistic rules. The rules for *be* and *have* triggers were the most difficult to formulate.

4.3 Experiments

We evaluate our models as usual using precision, recall and F1 metric for binary classification. The primary results we present in this section are obtained through 5-fold cross validation over all 25 sections of the automatically-parsed dataset. We use cross validation because the train-test split suggested by Bos and Spenader (2011) could result in highly varied results due to the small size of the dataset (see Table 1). Because the vast majority of auxiliaries do not trigger VPE, we over-sample the positive cases during training. Table 2 shows a comparison between the machine learning technique and a rule-based baseline for the six auxiliary forms. Table 3 shows results obtained from using the same train-test split used by Liu et al. (2016) in order to provide a direct comparison.

Results. Using a standard logistic regression classifier, we achieve an 11% improvement in accuracy over the baseline approach, as can be seen in Table 2. The rule-based approach was insufficient for *be* and *have* VPE, where logistic regression provides the largest improvements. Although we improve upon the baseline by 29%, the accuracy achieved for *be*-VPE is still low; this occurs mainly because: (i) *be* is the most commonly used auxiliary, so the number of negative examples is high compared to the number of positive examples; and, (ii) the analysis of the some of the false positives showed that there may have been genuine cases of VPE that were missed by the annotators of the dataset (Bos and Spenader, 2011). For example, this sentence (in file *wsj_2057*) was missed by the annotators (trigger in bold, antecedent italicized) "Some people tend to ignore that a 50-point move is *less in percentage terms* than it **was** when the stock market was lower."; here it is clear that **was** is a trigger for VPE.

Auxiliary	Baseline	ML	Change
Do	0.83	0.89	+0.06
Be	0.34	0.63	+0.29
Have	0.43	0.75	+0.32
Modal	0.80	0.86	+0.06
To	0.76	0.79	+0.03
So	0.67	0.86	+0.19
ALL	0.71	0.82	+0.11

Table 2: VPE detection results (baseline F1, Machine Learning F1, ML F1 improvement) obtained with 5-fold cross validation.

Test Set Results	P	R	F1
Liu et al. (2016)	**0.8022**	0.6134	0.6952
This work	0.7574	**0.8655**	**0.8078**

Table 3: Results (precision, recall, F1) for VPE detection using the train-test split proposed by Bos and Spenader (2011).

In Table 3, we compare our results to those achieved by Liu et al. (2016) when using WSJ sets 0-14 for training and sets 20-24 for testing. We improve on their overall accuracy by over 11%, due to the 25% improvement in recall achieved by our method. Our results show that oversampling the positive examples in the dataset and incorporating linguistically motivated syntactic features provide substantial gains for VPE detection. Additionally, we consider every instance of the word *to* as a potential trigger, while they do not - this lowers their recall because they miss every gold-standard instance of *to*-VPE. Thus, not only do we improve upon the state-of-the-art accuracy, but we also expand the scope of VPE-detection to include *to*-VPE without causing a significant decrease in accuracy.

5 Antecedent Identification

In this section we assume that we are given a trigger, from which we have to determine the correct antecedent; i.e., in the example in Figure 1, our task would be to identify "includes money spent on res-

idential renovation" as the correct antecedent. Our approach to this problem begins with generating a list of candidate antecedents. Next, we build a feature vector for each candidate by extracting features from the context surrounding the trigger and antecedent. Lastly, we use these features to learn a weight vector by using the Margin-Infused-Relaxed-Algorithm.

5.1 Candidate Generation

We generate a list of candidate antecedents by first extracting all VPs and ADJPs (and all contiguous combinations of their constituents) from the current sentence and the prior one. We then filter these candidates by predefining possible POS tags that an antecedent can start or end with according to the training set's gold standard antecedents. This method generates an average of 55 candidate antecedents per trigger, where triggers in longer sentences cause the creation of a larger number of candidate antecedents due to the larger number of VPs. This strategy accounts for 92% of the gold antecedents on the validation set by head match. We experimented with a less restrictive generation filter, but performance was not improved due to the much larger number of candidate antecedents.

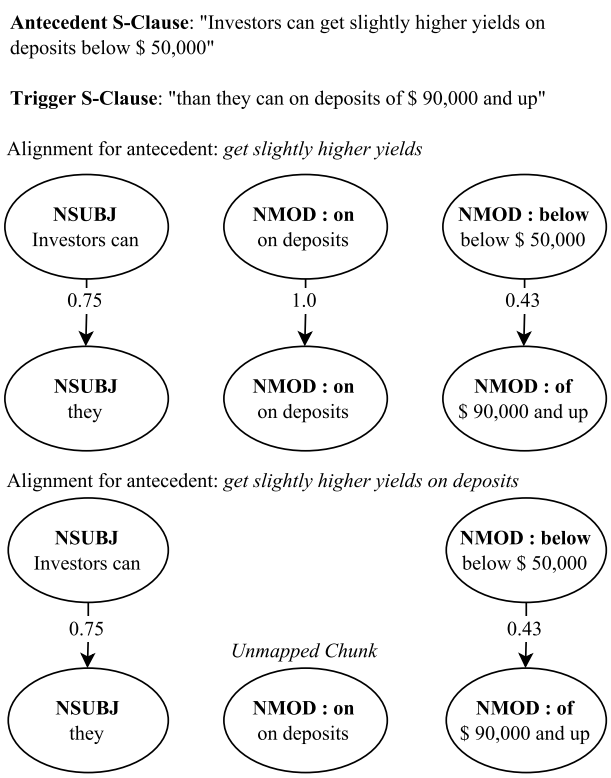


Figure 2: Alignment algorithm example with simplified dependencies.

5.2 Feature Extraction

We construct a feature vector representation for each candidate antecedent; in the example in Figure 1, for example, we would need feature vectors that differentiate between the two potential antecedents "includes money" and "includes money spent on residential renovation".

Alignment. This feature set results from an alignment algorithm that creates a mapping between the S-clause nearest to the trigger, S_t, and the S-clause nearest to the potential antecedent, S_a. The purpose of these features is to represent the parallelism (or lack thereof) between an antecedent's local vicinity with that of the trigger. The creation of this alignment algorithm was motivated by our intuition that the clause surrounding the trigger will have a parallel structure to that of the antecedent, and that an alignment between the two would best capture this parallelism. In the example sentence in Figure 2 (trigger in bold, antecedent italicized) "Investors can *get slightly higher yields* on deposits below $50,000 than they **can** on deposits of $90,000 and up" a simple observation of parallelism is that both the trigger and the correct antecedent are followed by the phrase "on deposits".

Formally, for each $S \in \{S_a, S_t\}$, we extract the dependencies in S as chunks of tokens, where each dependency chunk d_i contains all tokens between its governor and dependent (whichever comes first). Next, for each $d_i \in S_a$, if d_i contains any tokens that belong to the antecedent, delete those tokens. Similarly, for each $d_i \in S_t$, delete any token in d_i that belongs to T. We then perform a bipartite matching to align the $d_i \in S_t$ to the $d_j \in S_a$, where each edge's weight is determined by a scoring function $s(d_i, d_j)$. The scoring function we use considers the F1-similarity between the lemmas, POS-tags, and words shared between the two chunks, as well as whether or not the chunks share the same dependency name.

In the example in Figure 2 we can see that the correct antecedent, "*get slightly higher yields*", has a stronger alignment than the incorrect one, "*get slightly higher yields on deposits*". This occurs because we remove the candidate antecedent from its S-clause before creating the chunks; this leaves three nodes for the correct antecedent which map to the three nodes of the trigger's S-clause. However, this process only leaves two nodes for the incorrect candidate antecedent, thus causing one chunk to be unmapped, thus creating a weaker alignment.

We then use this mapping to generate a feature vector for the antecedent, which contains: the minimum, maximum, average, and standard deviation of the scores between chunks in the mapping; the number and percentage of unmapped chunks; the dependencies that have (and have not) been mapped to; the dependency pairs that were mapped together; and the minimum, maximum, average, and standard deviation of the cosine-similarity between the average word embedding of the words in a chunk between each d_i, d_j pair in the mapping.

NP Relation. These features compare the Noun Phrase (NP) closest to the antecedent to the NP closest to the trigger. This is motivated by an observation of many instances of VPE where it is often the case that the entity preceding the trigger is either repeated, similar, or corefers to the entity preceding the antecedent. The relationship between each *NP* is most significantly represented by features created with pre-trained *word2vec* word embeddings (Mikolov et al., 2013). For each NP, and for each word in the NP, we extract its pre-trained word embedding and then average them all together. We then use the cosine similarity between these two vectors as a feature.

Syntactic. Syntactic features are based on the relationship between the candidate antecedent's parse tree with that of the trigger. This feature set includes the following features, with the last three being influenced by Hardt's (1997) "preference factors" (a = candidate antecedent, t = trigger):

- ◇ if a's first word is an auxiliary
- ◇ if a's head (i.e., first main verb) is an auxiliary
- ◇ the POS tag of a's first and last words
- ◇ the frequency of each POS tag in the antecedent
- ◇ the frequency of each phrase (i.e., NP, VP, ADJP, etc.) in a's sentence and t's sentence
- ◇ if "than", "as", or "so" is between a and t
- ◇ if the word before a has the same POS-tag or lemma as t
- ◇ if a word in a c-commands a word in t
- ◇ if a's first or last word c-commands the trigger
- ◇ *Be-Do Form:* if the lemma of the token preceding a is *be* and the t's lemma is *do*
- ◇ *Recency:* distance between a and t and the distance between the t's nearest VP and a
- ◇ *Quotation:* if t is between quotation marks and similarly for a

Matching. This last feature set was influenced by the features described by Liu et al. (2016). We only use the "Match" features described by them; namely: whether the POS-tags, lemmas, or words in a two-token window before the start of the antecedent exactly match the two before the trigger; and whether the POS-tag, lemma, or word of the ith token before the antecedent equals that of the i-1th token before the trigger (for $i \in \{1, 2, 3\}$, where $i = 1$ considers the trigger itself).

5.3 Training Algorithm - MIRA

Since many potential antecedents share relatively similar characteristics, and since we have many features and few examples, we use the Margin-Infused-Relaxed-Algorithm (MIRA) in order to identify the most likely potential antecedent. MIRA maximizes the margin between the best candidate and the rest of the potential antecedents according to a loss function. It has been used for tasks with similar characteristics, such as statistical machine translation (Watanabe et al., 2007).

The training algorithm begins with a random initialization of the weight vector w. The training set contains triggers, each trigger's candidate antecedents, and their gold standard antecedents; it is reshuffled after each training epoch. We find the K highest-scoring potential antecedents, $a_1, \ldots, a_k$, according to the current weight value. A learning rate parameter determines how much we retain the new weight update with respect to the previous weight vector values.

MIRA defines the update step of the standard online training algorithm: it seeks to learn a weight

vector that, when multiplied with a feature vector f_i, gives the highest score to the antecedent that is most similar to the gold standard antecedent, a_*. This is posed as an optimization problem:

$$\begin{aligned} \underset{w_i}{\text{minimize}} \quad & \|w_i - w_{i-1}\| + C \sum_{k}^{K} \xi_k \\ \text{subject to} \quad & w_i \cdot a_* - w_i \cdot a_k + \xi_k \geq L(a_*, a_k), \\ & k = 1, \ldots, K \end{aligned} \tag{1}$$

Here, L is the loss function that controls the margin between candidates and the gold standard; it is defined as the evaluation metric proposed by Bos and Spenader (2011) (described in Section 5.5).

The ξ are slack variables and $C \geq 0$ is a hyper-parameter that controls the acceptable margin. This problem is solved by converting it to its Lagrange dual form[6].

5.4 Baseline Algorithm

The baseline we created was motivated by Bos's (2012) baseline algorithm: given a trigger, return as the antecedent the nearest VP that does not include the trigger. This is a naïve approach to antecedent identification because it does not consider the relationship between the context surrounding the antecedent and the context surrounding the trigger.

5.5 Experiments

We evaluate our results following the proposed metrics of Bos and Spenader (2011), as do Liu et al. (2016). Accuracy for antecedent identification is computed according to n = the number of correctly identified tokens between the candidate antecedent and the gold standard antecedent. Precision is n divided by the length of the candidate antecedent, recall is n divided by the length of the correct antecedent, and accuracy is the harmonic mean of precision and recall. For MIRA, final results are determined by choosing the weight vector that achieved the best performance on a validation set that is split off from part of the training set, as calculated after each update step.

[6]In this study, the dual form was implemented by hand using Gurobi's python API (Gurobi Optimization Inc., 2015).

Auxiliary	Baseline	MIRA	Change
do	0.42	0.71	+0.29
be	0.37	0.63	+0.26
modal	0.42	0.67	+0.25
so	0.15	0.53	+0.38
have	0.39	0.61	+0.22
to	0.03	0.58	+0.55
ALL	0.36	0.65	+0.29

Table 4: Results (baseline accuracy, MIRA accuracy, accuracy improvement) for antecedent identification; obtained with 5-fold cross validation.

End-to-end Results	P	R	F1
Liu et al. (2016)	**0.5482**	0.4192	0.4751
This work	0.4871	**0.5567**	**0.5196**

Table 5: End-to-end results (precision, recall, F1) using the train-test split proposed by Bos and Spenader (2011).

MIRA has several hyper-parameters that were tuned through a grid search over the validation set. The most crucial parameters were the learning rate α, and C, while the value of K did not cause significant changes in accuracy.

Results. In Table 4, we see that MIRA improves upon the baseline with a 29% increase in overall accuracy. MIRA provides significant gains for each form of VPE, although there is room for improvement, especially when identifying the antecedents of *do-so* triggers.

Liu et al. (2016) achieve an accuracy of 65.20% with their joint resolution model for antecedent identification when using the train-test split proposed by Bos and Spenader (2011); our model achieves 62.20% accuracy. However, their experimental design was slightly different than ours — they only considered antecedents of triggers detected by their oracle trigger detection method, while we use all gold-standard triggers, meaning our results are not directly comparable to theirs. Our cross validated results (65.18% accuracy) paint a better picture of the quality of our model because the small size of the dataset (554 samples) can cause highly varied results.

Excluded	P	R	F1
Auxiliary	0.7982	0.7611	0.7781
Lexical	0.6937	0.8408	0.7582
Syntactic	0.7404	0.7330	0.7343
NONE	0.8242	0.8120	0.8170

Table 6: Feature ablation results (feature set excluded, precision, recall, F1) on VPE detection; obtained with 5-fold cross validation.

Features Excluded	Accuracy
Alignment	0.6511
NP Relation	0.6428
Syntactic	0.5495
Matching	0.6504
NONE	0.6518

Table 7: Feature ablation results (feature set excluded, precision, recall, F1) on antecedent identification; obtained with 5-fold cross validation.

In Table 5 we present end-to-end results obtained from our system when using the triggers detected by our VPE detection model (see Section 4). We compare these results to the end-to-end results of the best model of Liu et al. (2016). Following Liu et al., we assign partial credit during end-to-end evaluation in the following way: for each correctly detected (true positive) trigger, the Bos and Spenader (2011) antecedent evaluation score between the trigger's predicted antecedent and its gold antecedent is used (as opposed to a value of 1). As can be seen from Table 5, we trade about 6 points of precision for 14 points of recall, thus improving state-of-the-art end-to-end accuracy from 47.51% to 51.96%.

6 Feature Ablation Studies

We performed feature ablation experiments in order to determine the impact that the different feature sets had on performance.

Trigger Detection. In Table 6 we can see that the syntactic features were essential for obtaining the best results, as can be seen by the 8.3% improvement, from 73.4% to 81.7%, obtained from including these features. This shows that notions from theoretical linguistics can prove to be invaluable when approaching the problem of VPE detection and that extracting these features in related problems may improve performance.

Antecedent Identification. Table 7 presents the results from a feature ablation study on antecedent identification. The most striking observation is that the alignment features do not add any significant improvement in the results. This is either because there simply is not an inherent parallelism between the trigger site and the antecedent site, or because the other features represent the parallelism adequately without necessitating the addition of the alignment features. The heuristic syntactic features provide a large (10%) accuracy improvement when included. These results show that a dependency-based alignment approach to feature extraction does not represent the parallelism between the trigger and antecedent as well as features based on the lexical and syntactic properties of the two.

7 Conclusion and Future Work

We presented an approach for the tasks of Verb Phrase Ellipsis detection and antecedent identification that leverages features informed both by theoretical linguistics and NLP, and employs machine learning methods to build VPE detection and antecedent identification tools using these features. Our results show the importance of distinguishing VPE triggers from each other, and highlight the importance of using the notion of c-command for both tasks.

For VPE detection, we improve upon the accuracy of the state-of-the-art system by over 11%, from 69.52% to 80.78%. For antecedent identification, our results significantly improve upon a baseline algorithm and we present results that are competitive with the state-of-the-art, as well as state-of-the-art results for an end-to-end system. We also expand the scope of previous state-of-the-art by including the detection and resolution of *to*-VPE, thus building a system that encompasses the entirety of the Bos and Spenader (2011) VPE dataset.

In future work, we would like to further inves-

tigate other margin-based optimizations similar to MIRA, but perhaps even more resilient to overfitting. We also seek to improve the antecedent identification approach by extracting stronger features.

Acknowledgments

This work was funded by McGill University and the Natural Sciences and Engineering Research Council of Canada via a Summer Undergraduate Research Project award granted to the first author. We thank the anonymous reviewers for their helpful suggestions, and we thank Nielsen, Hector Liu, and Edgar Gonzàlez for their clarifying remarks over email.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics.

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of VP ellipsis. *Language Resources and Evaluation*, 45(4):463–494.

Johan Bos. 2012. Robust VP ellipsis resolution in DR theory. In Staffan Larsson and Lars Borin, editors, *From Quantification to Conversation*, volume 19 of *Tributes*, pages 145–159. College Publications.

Andrew Carnie. 2013. *Syntax: A generative introduction*. John Wiley & Sons.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Mary Dalrymple, Stuart M Shieber, and Fernando CN Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Gurobi Optimization Inc. 2015. Gurobi optimizer reference manual.

Daniel Hardt. 1992. An algorithm for VP ellipsis. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics*, pages 9–14. Association for Computational Linguistics.

Daniel Hardt. 1997. An empirical approach to VP ellipsis. *Computational Linguistics*, 23(4):525–541.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP '09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics.

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*, pages 4539–4544.

Zhengzhong Liu, Edgar Gonzalez, and Dan Gillick. 2016. Exploring the steps of verb phrase ellipsis. In *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016), co-located with NAACL 2016*, pages 32–40.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

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

Leif Arda Nielsen. 2005. *A Corpus-Based Study of Verb Phrase Ellipsis Identification and Resolution*. Ph.D. thesis, King's College London.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Ivan A Sag. 1976. *Deletion and logical form.* Ph.D. thesis, Massachusetts Institute of Technology.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Annual Meeting - Association for Computational Linguistics*, volume 45, page 240.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773.

Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser

Adhiguna Kuncoro[♠] **Miguel Ballesteros**[◇] **Lingpeng Kong**[♠]
Chris Dyer[♠♣] **Noah A. Smith**[♡]
[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[◇]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♣]Google DeepMind, London, UK
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
{akuncoro,cdyer,lingpenk}@cs.cmu.edu
miguel.ballesteros@upf.edu, nasmith@cs.washington.edu

Abstract

We introduce two first-order graph-based dependency parsers achieving a new state of the art. The first is a consensus parser built from an ensemble of independently trained greedy LSTM transition-based parsers with different random initializations. We cast this approach as minimum Bayes risk decoding (under the Hamming cost) and argue that weaker consensus within the ensemble is a useful signal of difficulty or ambiguity. The second parser is a "distillation" of the ensemble into a single model. We train the distillation parser using a structured hinge loss objective with a novel cost that incorporates ensemble uncertainty estimates for each possible attachment, thereby avoiding the intractable cross-entropy computations required by applying standard distillation objectives to problems with structured outputs. The first-order distillation parser matches or surpasses the state of the art on English, Chinese, and German.

1 Introduction

Neural network dependency parsers achieve state of the art performance (Dyer et al., 2015; Weiss et al., 2015; Andor et al., 2016), but training them involves gradient descent on non-convex objectives, which is unstable with respect to initial parameter values. For some tasks, an **ensemble** of neural networks from different random initializations has been found to improve performance over individual models (Sutskever et al., 2014; Vinyals et al., 2015, *inter alia*). In §3, we apply this idea to build a first-order graph-based (FOG) ensemble parser (Sagae and Lavie, 2006) that seeks consensus among 20 randomly-initialized stack LSTM parsers (Dyer et al., 2015), achieving nearly the best-reported performance on the standard Penn Treebank Stanford dependencies task (94.51 UAS, 92.70 LAS).

We give a probabilistic interpretation to the ensemble parser (with a minor modification), viewing it as an instance of **minimum Bayes risk** inference. We propose that disagreements among the ensemble's members may be taken as a signal that an attachment decision is difficult or ambiguous.

Ensemble parsing is not a practical solution, however, since an ensemble of N parsers requires N times as much computation, plus the runtime of finding consensus. We address this issue in §5 by **distilling** the ensemble into a **single** FOG parser with discriminative training by defining a new *cost function*, inspired by the notion of "soft targets" (Hinton et al., 2015). The essential idea is to derive the cost of each possible attachment from the ensemble's division of votes, and use this cost in discriminative learning. The application of distilliation to structured prediction is, to our knowledge, new, as is the idea of empirically estimating cost functions.

The distilled model performs almost as well as the ensemble consensus and much better than (i) a strong LSTM FOG parser trained using the conventional Hamming cost function, (ii) recently published strong LSTM FOG parsers (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016), and (iii) many higher-order graph-based parsers (Koo and Collins, 2010; Martins et al., 2013; Le and Zuidema, 2014). It represents a new state of the art for graph-based dependency parsing for English, Chinese, and

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1744–1753,
Austin, Texas, November 1-5, 2016.

German. The code to reproduce our results is publicly available.[1]

2 Notation and Definitions

Let $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ denote an n-length sentence. A dependency parse for $\boldsymbol{x}$, denoted $\boldsymbol{y}$, is a set of tuples (h, m, ℓ), where h is the index of a head, m the index of a modifier, and ℓ a dependency label (or relation type). Most dependency parsers are constrained to return $\boldsymbol{y}$ that form a directed tree.

A first-order graph-based (**FOG**; also known as "arc-factored") dependency parser exactly solves

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \underbrace{\sum_{(h,m) \in \boldsymbol{y}} s(h, m, \boldsymbol{x})}_{S(\boldsymbol{y}, \boldsymbol{x})}, \quad (1)$$

where $\mathcal{T}(\boldsymbol{x})$ is the set of directed trees over $\boldsymbol{x}$, and s is a local scoring function that considers only a single dependency arc at a time. (We suppress dependency labels; there are various ways to incorporate them, discussed later.) To define s, McDonald et al. (2005a) used hand-engineered features of the surrounding and in-between context of x_h and x_m; more recently, Kiperwasser and Goldberg (2016) used a bidirectional LSTM followed by a single hidden layer with non-linearity.

The exact solution to Eq. 1 can be found using a minimum (directed) spanning tree algorithm (McDonald et al., 2005b) or, under a projectivity constraint, a dynamic programming algorithm (Eisner, 1996), in $O(n^2)$ or $O(n^3)$ runtime, respectively. We refer to parsing with a minimum spanning tree algorithm as **MST parsing**.

An alternative that runs in linear time is **transition-based** parsing, which recasts parsing as a sequence of actions that manipulate auxiliary data structures to incrementally build a parse tree (Nivre, 2003). Such parsers can return a solution in a faster $O(n)$ asymptotic runtime. Unlike FOG parsers, transition-based parsers allow the use of scoring functions with history-based features, so that attachment decisions can interact more freely; the best performing parser at the time of this writing employ neural networks (Andor et al., 2016).

[1] https://github.com/adhigunasurya/distillation_parser.git

Let $h_{\boldsymbol{y}}(m)$ denote the parent of x_m in $\boldsymbol{y}$ (using a special null symbol when m is the root of the tree), and $h_{\boldsymbol{y'}}(m)$ denotes the parent of x_m in the predicted tree $\boldsymbol{y'}$. Given two dependency parses of the same sentence, $\boldsymbol{y}$ and $\boldsymbol{y'}$, the **Hamming cost** is

$$C_H(\boldsymbol{y}, \boldsymbol{y'}) = \sum_{m=1}^{n} \begin{cases} 0 & \text{if } h_{\boldsymbol{y}}(m) = h_{\boldsymbol{y'}}(m) \\ 1 & \text{otherwise} \end{cases}$$

This cost underlies the standard dependency parsing evaluation scores (unlabeled and labeled attachment scores, henceforth UAS and LAS). More generally, a **cost function** C maps pairs of parses for the same sentence to non-negative values interpreted as the cost of mistaking one for the other, and a **first-order cost function** (FOC) is one that decomposes by attachments, like the Hamming cost.

Given a cost function C and a probabilistic model that defines $p(\boldsymbol{y} \mid \boldsymbol{x})$, **minimum Bayes risk** (MBR) decoding is defined by

$$\begin{aligned} \hat{\boldsymbol{y}}_{\text{MBR}}(\boldsymbol{x}) &= \arg\min_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \sum_{\boldsymbol{y'} \in \mathcal{T}(\boldsymbol{x})} p(\boldsymbol{y'} \mid \boldsymbol{x}) \cdot C(\boldsymbol{y}, \boldsymbol{y'}) \\ &= \arg\min_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x})}[C(\boldsymbol{y}, \boldsymbol{Y})]. \end{aligned} \quad (2)$$

Under the Hamming cost, MBR parsing equates algorithmically to FOG parsing with $s(h, m, \boldsymbol{x}) = p((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x})$, the posterior marginal of the attachment under p. This is shown by linearity of expectation; see also Titov and Henderson (2006).

Apart from MBR decoding, cost functions are also used for *discriminative training* of a parser. For example, suppose we seek to estimate the parameters $\boldsymbol{\theta}$ of scoring function $S_{\boldsymbol{\theta}}$. One approach is to minimize the structured hinge loss of a training dataset $\mathcal{D}$ with respect to $\boldsymbol{\theta}$:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}} [&- S_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) \\ &+ \max_{\boldsymbol{y'} \in \mathcal{T}(\boldsymbol{x})} \big(S_{\boldsymbol{\theta}}(\boldsymbol{y'}, \boldsymbol{x}) + C(\boldsymbol{y'}, \boldsymbol{y}) \big)] \end{aligned} \quad (3)$$

Intuitively, this amounts to finding parameters that separate the model score of the correct parse from any wrong parse by a distance proportional to the cost of the wrong parse. With regularization, this is equivalent to the structured support vector machine

(Taskar et al., 2005; Tsochantaridis et al., 2005), and if $S_{\boldsymbol{\theta}}$ is (sub)differentiable, many algorithms are available. Variants have been used extensively in training graph-based parsers (McDonald et al., 2005b; Martins et al., 2009), which typically make use of Hamming cost, so that the inner $\max$ can be solved efficiently using FOG parsing with a slightly revised local scoring function:

$$s'(h, m, \boldsymbol{x}) = s(h, m, \boldsymbol{x}) + \begin{cases} 0 & \text{if } (h, m) \in \boldsymbol{y} \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

Plugging this into Eq. 1 is known as **cost-augmented** parsing.

3 Consensus and Minimum Bayes Risk

Despite the recent success of neural network dependency parsers, most prior works exclusively report single-model performance. Ensembling neural network models trained from different random starting points is a standard technique in a variety of problems, such as machine translation (Sutskever et al., 2014) and constituency parsing (Vinyals et al., 2015). We aim to investigate the benefit of ensembling independently trained neural network dependency parsers by applying the parser ensembling method of Sagae and Lavie (2006) to a collection of N strong neural network base parsers.

Here, each base parser is an instance of the greedy, transition-based parser of Dyer et al. (2015), known as the stack LSTM parser, trained from a different random initial estimate. Given a sentence $\boldsymbol{x}$, the consensus FOG parser (Eq. 1) defines score $s(h, m, \boldsymbol{x})$ as the number of base parsers that include the attachment (h, m), which we denote $votes(h, m)$.[2] An example of this scoring function with an ensemble of 20 models is shown in Figure 1 We assign to dependency (h, m) the label most frequently selected by the base parsers that attach m to h.

Next, note that if we let $s(h, m, \boldsymbol{x}) = votes(h, m)/N$, this has no effect on the parser (we have only scaled by a constant factor). We can therefore view s as a posterior marginal, and the ensemble parser as an MBR parser (Eq. 2).

[2]An alternative to building an ensemble of stack LSTM parsers in this way would be to average the softmax decisions at each timestep (transition), similar to Vinyals et al. (2015).

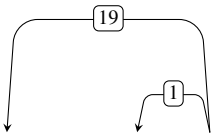


Figure 1: Our ensemble's votes (20 models) on an ambiguous PP attachment of *with*. The ensemble is nearly but not perfectly unanimous in selecting *saw* as the head.

Model	UAS	LAS	UEM
Andor et al. (2016)	**94.61**	**92.79**	-
$N = 1$ (stack LSTM)	93.10	90.90	47.60
ensemble, $N = 5$, MST	93.91	91.94	50.12
ensemble, $N = 10$, MST	94.34	92.47	52.07
ensemble, $N = 15$, MST	94.40	92.57	51.86
ensemble, $N = 20$, MST	94.51	92.70	**52.44**

Table 1: PTB-SD task: ensembles improve over a strong greedy baseline. UEM indicates unlabeled exact match.

Experiment. We consider this approach on the Stanford dependencies version 3.3.0 (De Marneffe and Manning, 2008) Penn Treebank task. As noted, the base parsers instantiate the greedy stack LSTM parser (Dyer et al., 2015).[3]

Table 1 shows that ensembles, even with small N, strongly outperform a single stack LSTM parser. Our ensembles of greedy, locally normalized parsers perform comparably to the best previously reported, due to Andor et al. (2016), which uses a beam (width 32) for training and decoding.

4 What is Ensemble Uncertainty?

While previous works have already demonstrated the merit of ensembling in dependency parsing (Sagae and Lavie, 2006; Surdeanu and Manning, 2010), usually with diverse base parsers, we consider whether the posterior marginals estimated by $\hat{p}((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x}) = votes(h, m)/N$ can be interpreted. We conjecture that disagreement among base parsers about where to attach x_m (i.e., uncertainty in the posterior) is a sign of difficulty or am-

[3]We use the standard data split (02–21 for training, 22 for development, 23 for test), automatically predicted part-of-speech tags, same pretrained word embedding as Dyer et al. (2015), and recommended hyperparameters; `https://github.com/clab/lstm-parser`, each with a different random initialization; this differs from past work on ensembles, which often uses different base model architectures.

Sentence: It will **go** for **work** ranging from refinery **modification** to **changes** in the distribution **system**, *including* the way service stations **pump** fuel into cars.

x_h	posterior	new cost	Hamming
go	0.143	0.143	1
work	0.095	0.191	1
modification	0.190	0.096	1
changes	**0.286**	**0.000**	**0**
system	0.095	0.191	1
pump	0.190	0.096	1
stations	0.000	0.286	1

Table 2: An ambiguous sentence from the training set and the posteriors[4] of various possible parents for *including*. The last two columns are, respectively, the contributions to the distillation cost C_D (explained in §5.1, Eq. 5) and the standard Hamming cost C_H. The most probable head under the ensemble is *changes*, which is also the correct answer.

biguity. If this is true, then the ensemble provides information about which confusions are more or less reasonable—information we will exploit in our distilled parser (§5).

A complete linguistic study is out of scope here; instead, we provide a motivating example before empirically validating our conjecture. Table 2 shows an example where there is considerable disagreement among base parsers over the attachment of a word (*including*). We invite the reader to attempt to select the correct attachment and gauge the difficulty of doing so, before reading on.

Regardless of whether our intuition that this is an inherently difficult and perhaps ambiguous case is correct, it is uncontroversial to say that the words in the sentence not listed, which received zero votes (e.g., both instances of *the*), are obviously implausible attachments.

Our next idea is to transform ensemble uncertainty into a new estimate of cost—a replacement for the Hamming cost—and use it in discriminative training of a single FOG parser. This allows us to distill what has been learned by the ensemble into a single model.

5 Distilling the Ensemble

Despite its state of the art performance, our ensemble requires N parsing calls to decode each sentence. To reduce the computational cost, we introduce a method for "distilling" the ensemble's knowledge into a single parser, making use of a novel **cost function** to communicate this knowledge from the ensemble to the distilled model. While models that combine the outputs of other parsing models have been proposed before (Martins et al., 2008; Nivre and McDonald, 2008; Zhang and Clark, 2008, *inter alia*), these works incorporated the scores or outputs of the baseline parsers as **features** and as such require running the first-stage models at test-time. Creating a cost function from a data analysis procedure is, to our knowledge, a new idea.

The idea is attractive because cost functions are model-agnostic; they can be used with any parser amenable to discriminative training. Further, only the training procedure changes; parsing at test time does not require consulting the ensemble at all, avoiding the costly application of the N parsers to new data, unlike model combination techniques like stacking and beam search.

Distilling an ensemble of classifiers into one simpler classifer that behaves similarly is due to Bucilă et al. (2006) and Hinton et al. (2015); they were likewise motivated by a desire to create a simpler model that was cheaper to run at test time. In their work, the ensemble provides a probability distribution over labels for each input, and this predicted distribution serves as the training target for the distilled model (a sum of two cross entropies objective is used, one targeting the empirical training distribution and the other targeting the ensemble's posterior distribution). This can be contrasted with the supervision provided by the training data alone, which conventionally provides a single correct label for each instance. These are respectively called "soft" and "hard" targets.

We propose a novel adaptation of the soft target idea to the structured output case. Since a sentence

[4] In §3, we used 20 models. Since those 20 models were trained on the whole training set, they cannot be used to obtain the uncertainty estimates on the training set, where the example sentence in Table 2 comes from. Therefore we trained a new ensemble of 21 models from scratch with five-way jackknifing. The same jackknifing setting is used in the distillation parser (§6).

Sentence: John **saw** the **woman** *with* a telescope

x_h	soft	hard
John	0.0	0
saw	**0.95**	**1**
the	0.0	0
woman	0.05	0
a	0.0	0
telescope	0.0	0

Table 3: Example of soft targets (taken from our 20-model ensemble's uncertainty on the sentence) and hard targets (taken from the gold standard) for possible parents of *with*. The soft target corresponds with the posterior (second column) in Table 2, but the hard target differs from the Hamming cost (last column of Table 2) since the hard target assigns a value of 1 to the correct answer and 0 to all others (the reverse is true for Hamming cost).

has an exponential (in its length) number of parses, representing the posterior distribution over parses predicted by the ensemble is nontrivial. We solve this problem by taking a single parse from each model, representing the N-sized ensemble's parse distribution using N samples.

Second, rather than considering uncertainty at the level of complete parse trees (which would be analogous to the classification case) or larger structures, we instead consider uncertainty about *individual* attachments, and seek to "soften" the attachment targets used in training the parser. An illustration for the prepositional phrase attachment ambiguity in Fig. 1, taken from the ensemble output for the sentence, is shown in Table 3. Soft targets allow us to encode the notion that mistaking *woman* as the parent of *with* is less bad than attaching *with* to *John* or *telescope*. Hard targets alone do not capture this information.

5.1 Distillation Cost Function

The natural place to exploit this additional information when training a parser is in the cost function. When incorporated into discriminative training, the Hamming cost encodes hard targets: the correct attachment should receive a higher score than all incorrect ones, with the same margin. Our distillation cost function aims to reduce the cost of decisions that—based on the ensemble uncertainty—appear to be more difficult, or where there may be multiple plausible attachments.

Let $\pi(h, m) =$

$$1 - \hat{p}((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x}) = \frac{N - votes(h, m)}{N}.$$

Our new cost function is defined by $C_D(\boldsymbol{y}, \boldsymbol{y'}) =$

$$\begin{aligned} &\textstyle\sum_{m=1}^{n} \max\left\{0, \pi(h_{\boldsymbol{y'}}(m), m) - \pi(h_{\boldsymbol{y}}(m), m)\right\} \\ &= \textstyle\sum_{m=1}^{n} \max\left\{0, \hat{p}(h_{\boldsymbol{y}}(m), m) - \hat{p}(h_{\boldsymbol{y'}}(m), m)\right\}. \end{aligned} \quad (5)$$

Recall that $\boldsymbol{y}$ denotes the correct parse, according to the training data, and $\boldsymbol{y'}$ is a candidate parse.

This function has several attractive properties:

1. When a word x_m has more than one plausible (according to the ensemble) but incorrect (according to the annotations) attachment, each one has a diminished cost (relative to Hamming cost and all implausible attachments).
2. The correct attachment (according to the gold-standard training data) always has zero cost since $h_{\boldsymbol{y}}(m) = h_{\boldsymbol{y'}}(m)$ and Eq. 5 cancels out.
3. When the ensemble is confident, cost for its choice(s) is lower than it would be under Hamming cost—even when the ensemble is wrong. This means that we are largely training the distilled parser to simulate the ensemble, including mistakes and correct predictions. This encourages the model to replicate the state of the art ensemble performance.
4. Further, when the ensemble is perfectly confident and correct, every incorrect attachment has a cost of 1, just as in Hamming cost.
5. The cost of any attachment is bounded above by the proportion of votes assigned to the correct attachment.

One way to understand this cost function is to imagine that it gives the parser more ways to achieve a zero-cost[5] attachment. The first is to correctly attach a word to its correct parent. The second is to predict a parent that the ensemble prefers to the correct parent, i.e., $\pi(h_{\boldsymbol{y'}}(m), m) < \pi(h_{\boldsymbol{y}}(m), m)$. Any other decision will incur a non-zero cost that is

[5]It is important to note the difference between cost (Eq. 5) and loss (Eq. 3).

proportional to the implausibility of the attachment, according to the ensemble. Hence the model is supervised both by the hard targets in the training data annotations and the soft targets from the ensemble.

While it may seem counter-intuitive to place zero cost on an incorrect attachment, recall that the *cost* is merely a margin that must separate the scores of parses containing correct and incorrect arcs. In contrast, the *loss* (in our case, the structured hinge loss) is the "penalty" the learner tries to minimize while training the graph-based parser, which depends on both the *cost* and *model score* as defined in Equation 3. When an incorrect arc is preferred by the ensemble over the gold arc (hence assigned a cost/margin of 0), the model will still incur a loss if $s(h_{\boldsymbol{y}}(m), m, \boldsymbol{x}) < s(h_{\boldsymbol{y'}}(m), m, \boldsymbol{x})$. In other words, the score of *any* incorrect arc (including those strongly preferred by the ensemble) cannot be higher than the score of the gold arc.

The learner only incurs 0 loss if $s(h_{\boldsymbol{y}}(m), m, \boldsymbol{x}) \geq s(h_{\boldsymbol{y'}}(m), m, \boldsymbol{x})$. This means that the gold score and the predicted score can have a margin of 0 (i.e., have the same score and incur no loss) when the ensemble is highly confident of that prediction, but the score of the correct parse cannot be lower regardless of how confident the ensemble is (hence the objective does not encourage incorrect trees at the expense of gold ones).

In the example in Table 2, we show the (additive) contribution to the distillation cost by each attachment decision (column labeled "new cost"). Note that more plausible attachments according to the ensemble have a lower cost than less plausible ones (e.g., the cost for *modification* is less than *system*, though both are incorrect). While in the last line *stations* received no votes in the ensemble (implausible attachment), its contribution to the cost is bounded by the proportion of votes for correct attachment. The intuition is that, when the ensemble is not certain of the correct answer, it should not assign a large cost to implausible attachments. In contrast, Hamming cost would assign a cost of 1 (column labeled "Hamming") in all incorrect cases.

5.2 Distilled Parser

Our distilled parser is trained discriminatively with the structured hinge loss (Eq. 3). This is a natural choice because it makes the cost function explicit and central to learning.[6] Further, because our ensemble's posterior gives us information about each attachment individually, the cost function we construct can be first-order, which simplifies training with exact inference.

This approach to training a model is well-studied for a FOG parser, but not for a transition-based parser, which is comprised of a collection of classifiers trained to choose good sequences of transitions—not to score whole trees for good attachment accuracy. Transition-based approaches are therefore unsuitable for our proposed distillation cost function, even though they are asymptotically faster. We proceed with a FOG parser (with Eisner's algorithm for English and Chinese, and MST for German since it contains a considerable number of non-projective trees) as the distilled model.

Concretely, we use a bidirectional LSTM followed by a hidden layer of non-linearity to calculate the scoring function $s(h, m, \boldsymbol{x})$, following Kiperwasser and Goldberg (2016) with minor modifications. The bidirectional LSTM maps each word x_i to a vector $\bar{\mathbf{x}}_i$ that embeds the word in context (i.e., $\boldsymbol{x}_{1:i-1}$ and $\boldsymbol{x}_{i+1:n}$). Local attachment scores are given by:

$$s(h, m, \boldsymbol{x}) = \mathbf{v}^\top \tanh\left(\mathbf{W}[\bar{\mathbf{x}}_h; \bar{\mathbf{x}}_m] + \mathbf{b}\right) \quad (6)$$

where the model parameters are $\mathbf{v}$, $\mathbf{W}$, and $\mathbf{b}$, plus the bidirectional LSTM parameters. We will refer to this parsing model as **neural FOG**.

Our model architecture is nearly identical to that of Kiperwasser and Goldberg (2016), with two primary differences. The first difference is that we fix the pretrained word embeddings and compose them with learned embeddings and POS tag embeddings (Dyer et al., 2015), allowing the model to simultaneously leverage pretrained vectors and learn a task-specific representation.[7] Unlike Kiperwasser and Goldberg (2016), we did not observe any degradation by incorporating the pretrained vectors. Second,

[6] Alternatives that do not use cost functions include probabilistic parsers, whether locally normalized like the stack LSTM parser used within our ensemble, or globally normalized, as in Andor et al. (2016); cost functions can be incorporated in such cases with minimum risk training (Smith and Eisner, 2006) or softmax margin (Gimpel and Smith, 2010).

[7] To our understanding, Kiperwasser and Goldberg (2016) initialized with pretrained vectors and backpropagated during training.

we apply a per-epoch learning rate decay of 0.05 to the Adam optimizer. While the Adam optimizer automatically adjusts the global learning rate according to past gradient magnitudes, we find that this additional per-epoch decay consistently improves performance across all settings and languages.

6 Experiments

We ran experiments on the English PTB-SD version 3.3.0, Penn Chinese Treebank (Xue et al., 2002), and German CoNLL 2009 (Hajič et al., 2009) tasks.

Experimental settings. We used the standard splits for all languages. Like Chen and Manning (2014) and Dyer et al. (2015), we use predicted tags with the Stanford tagger (Toutanova et al., 2003) for English and gold tags for Chinese. For German we use the predicted tags provided by the CoNLL 2009 shared task organizers. All models were augmented with pretrained structured-skipgram (Ling et al., 2015) embeddings; for English we used the Gigaword corpus and 100 dimensions, for Chinese Gigaword and 80, and for German WMT 2010 monolingual data and 64.

Hyperparameters. The hyperparameters for neural FOG are summarized in Table 4. For the Adam optimizer we use the default settings in the CNN neural network library.[8] Since the ensemble is used to obtain the uncertainty on the training set, it is imperative that the stack LSTMs do not overfit the training set. To address this issue, we performed five-way jackknifing of the training data for each stack LSTM model to obtain the training data uncertainty under the ensemble. To obtain the ensemble uncertainty on each language, we use 21 base models for English (see footnote 4), 17 for Chinese, and 11 for German.

Speed. One potential drawback of using a quadratic or cubic time parser to distill an ensemble of linear-time transition-based models is speed. Our FOG model is implemented using the same CNN library as the stack LSTM transition-based parser. On the same single-thread CPU hardware, the distilled MST parser[9] parses 20 sentences per second without any pruning, while a single stack LSTM model is only three times faster at 60 sentences per second. Running an ensemble of 20 stack LSTMs is at least 20 times slower (without multi-threading), not including consensus parsing. In the end, the distilled parser is more than ten times faster than the ensemble pipeline.

Bi-LSTM dimension	100
Bi-LSTM layers	2
POS tag embedding	12
Learned word embedding	32
Hidden Layer Units	100
Labeler Hiden Layer Units	100
Optimizer	Adam
Learning rate decay	0.05

Table 4: Hyperparameters for the distilled FOG parser. Both the model architecture and the hyperparameters are nearly identical with Kiperwasser and Goldberg (2016). We apply a per-epoch learning rate decay to the Adam optimizer, which consistently improves performance across all datasets.

Accuracy. All scores are shown in Table 5. First, consider the neural FOG parser trained with Hamming cost (C_H in the second-to-last row). This is a very strong benchmark, outperforming many higher-order graph-based and neural network models on all three datasets. Nonetheless, training the same model with distillation cost gives consistent improvements for all languages. For English, we see that this model comes close to the slower ensemble it was trained to simulate. For Chinese, it achieves the best published scores, for German the best published UAS scores, and just after Bohnet and Nivre (2012) for LAS.

Effects of Pre-trained Word Embedding. As an ablation study, we ran experiments on English without pre-trained word embedding, both with the Hamming and distillation costs. The model trained with Hamming cost achieved 93.1 UAS and 90.9 LAS, compared to 93.6 UAS and 91.1 LAS for the model with distillation cost. This result further showcases the consistent improvements from using the distillation cost across different settings and languages.

We conclude that "soft targets" derived from ensemble uncertainty offer useful guidance, through the distillation cost function and discriminative training of a graph-based parser. Here we consid-

[8] https://github.com/clab/cnn.git

[9] The runtime of the Hamming-cost bidirectional LSTM FOG parser is the same as the distilled parser.

System	Method	P?	PTB-SD		CTB		German CoNLL'09	
			UAS	LAS	UAS	LAS	UAS	LAS
Zhang and Nivre (2011)	Transition (beam)		-	-	86.0	84.4	-	-
Bohnet and Nivre (2012)†	Transition (beam)		-	-	87.3	85.9	91.37	89.38
Chen and Manning (2014)	Transition (greedy)	✓	91.8	89.6	83.9	82.4	-	-
Dyer et al. (2015)	Transition (greedy)	✓	93.1	90.9	87.2	85.7	-	-
Weiss et al. (2015)	Transition (beam)	✓	94.0	92.0	-	-	-	-
Yazdani and Henderson (2015)	Transition (beam)		-	-	-	-	89.6	86.0
Ballesteros et al. (2015)	Transition (greedy)		91.63	89.44	85.30	83.72	88.83	86.10
Ballesteros et al. (2016)	Transition (greedy)	✓	93.56	91.42	87.65	86.21	-	-
Kiperwasser and Goldberg (2016)	Transition (greedy)	✓	93.9	91.9	87.6	86.1	-	-
Andor et al. (2016)	Transition (beam)	✓	**94.61**	**92.79**	-	-	90.91	89.15
Ma and Zhao (2012)	Graph (4th order)		-	-	87.74	-	-	-
Martins et al. (2013)	Graph (3rd order)		93.1	-	-	-	-	-
Le and Zuidema (2014)	Reranking/blend	✓	93.8	91.5	-	-	-	-
Zhu et al. (2015)	Reranking/blend	✓	-	-	85.7	-	-	-
Kiperwasser and Goldberg (2016)	Graph (1st order)		93.1	91.0	86.6	85.1	-	-
Wang and Chang (2016)	Graph (1st order)	✓	94.08	91.82	87.55	86.23	-	-
This work: ensemble, $N = 20$, MST	Transition (greedy)	✓	94.51	92.70	**89.80**	**88.56**	**91.86**	**89.98**
This work: neural FOG, C_H	Graph (1st order)	✓	93.76	91.60	87.32	85.82	91.22	88.82
This work: neural FOG, C_D (distilled)	Graph (1st order)	✓	94.26	92.06	88.87	87.30	91.60	89.24

Table 5: Dependency parsing performance on English, Chinese, and German tasks. The "P?" column indicates the use of pretrained word embeddings. Reranking/blend indicates that the reranker score is interpolated with the base model's score. Note that previous works might use different predicted tags for English. We report accuracy without punctuation for English and Chinese, and with punctuation for German, using the standard evaluation script in each case. We only consider systems that do not use additional training data. The best overall results are indicated with bold (this was achieved by the ensemble of greedy stack LSTMs in Chinese and German), while the best non-ensemble model is denoted with an underline. The † sign indicates the use of predicted tags for Chinese in the original publication, although we report accuracy using gold Chinese tags based on private correspondence with the authors.

ered a FOG parser, though future work might investigate any parser amenable to training to minimize a cost-aware loss like the structured hinge.

7 Related Work

Our work on ensembling dependency parsers is based on Sagae and Lavie (2006) and Surdeanu and Manning (2010); an additional contribution of this work is to show that the normalized ensemble votes correspond to MBR parsing. Petrov (2010) proposed a similar model combination with random initializations for phrase-structure parsing, using products of constituent marginals. The local optima in his base model's training objective arise from latent variables instead of neural networks (in our case).

Model distillation was proposed by Bucilă et al. (2006), who used a single neural network to simulate a large ensemble of classifiers. More recently, Ba and Caruana (2014) showed that a single shallow neural network can closely replicate the performance of an ensemble of deep neural networks in phoneme recognition and object detection. Our work is closer to Hinton et al. (2015), in the sense that we do not simply *compress* the ensemble and hit the "soft target," but also the "hard target" at the same time[10]. These previous works only used model compression and distillation for classification; we extend the work to a structured prediction problem (dependency parsing).

Täckström et al. (2013) similarly used an ensemble of other parsers to guide the prediction of a seed model, though in a different context of "ambiguity-aware" ensemble training to re-lexicalize a transfer model for a target language. We similarly use an ensemble of models as a supervision for a sin-

[10]Our cost is zero when the correct arc is predicted, regardless of what the soft target thinks, something a compression model without gold supervision cannot do.

gle model. By incorporating the ensemble uncertainty estimates in the cost function, our approach is cheaper, not requiring any marginalization during training. An additional difference is that we learn from the gold labels ("hard targets") rather than only ensemble estimates on unlabeled data.

Kim and Rush (2016) proposed a distillation model at the sequence level, with application in sequence-to-sequence neural machine translation. There are two primary differences with this work. First, we use a global model to distill the ensemble, instead of a sequential one. Second, Kim and Rush (2016) aim to distill a larger model into a smaller one, while we propose to distill an ensemble instead of a single model.

8 Conclusions

We demonstrate that an ensemble of 20 greedy stack LSTMs (Dyer et al., 2015) can achieve state of the art accuracy on English dependency parsing. This approach corresponds to minimum Bayes risk decoding, and we conjecture that the arc attachment posterior marginals quantify a notion of uncertainty that may indicate difficulty or ambiguity. Since running an ensemble is computationally expensive, we proposed discriminative training of a graph-based model with a novel cost function that *distills* the ensemble uncertainty. Deriving a cost function from a statistical model and extending distillation to structured prediction are new contributions. This distilled model, trained to simulate the slower ensemble parser, improves over the state of the art on Chinese and German.

Acknowledgments

We thank Swabha Swayamdipta, Sam Thomson, Jesse Dodge, Dallas Card, Yuichiro Sawai, Graham Neubig, and the anonymous reviewers for useful feedback. We also thank Juntao Yu and Bernd Bohnet for re-running the parser of Bohnet and Nivre (2012) on Chinese with gold tags. This work was sponsored in part by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114; it was also supported in part by Contract No. W911NF-15-1-0543 with the DARPA and the Army Research Office (ARO). Approved for public release, distribution unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Miguel Ballesteros was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA).

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Proc. of NIPS*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. of EMNLP*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack-LSTM parser. In *Proc. of EMNLP*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Kevin Gimpel and Noah A Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian

Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CONLL 2009 Shared Task*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proc. of EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proc. of COLING*.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

André F. T. Martins, Mariana S.C. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Proc.of NAACL*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proc. of NAACL*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of ACL*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proc. of NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Oscar Täckström, Ryan T. McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proc. of NAACL*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proc. of ICML*.

Ivan Titov and James Henderson. 2006. Bayes risk minimization in natural language parsing. Technical report, University of Geneva.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proc. of ACL*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL*.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proc. of COLING*.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proc. of CoNLL*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. of EMNLP*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proc. of ACL*.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proc. of ACL-IJCNLP*.

LSTM Shift-Reduce CCG Parsing

Wenduan Xu
Computer Laboratory
University of Cambridge
wx217@cam.ac.uk

Abstract

We describe a neural shift-reduce parsing model for CCG, factored into four unidirectional LSTMs and one bidirectional LSTM. This factorization allows the linearization of the complete parsing history, and results in a highly accurate greedy parser that outperforms all previous beam-search shift-reduce parsers for CCG. By further deriving a globally optimized model using a task-based loss, we improve over the state of the art by up to 2.67% labeled F1.

1 Introduction

Combinatory Categorial Grammar (CCG; Steedman, 2000) parsing is challenging due to its so-called "spurious" ambiguity that permits a large number of non-standard derivations (Vijay-Shanker and Weir, 1993; Kuhlmann and Satta, 2014). To address this, the *de facto* models resort to chart-based CKY (Hockenmaier, 2003; Clark and Curran, 2007), despite CCG being naturally compatible with shift-reduce parsing (Ades and Steedman, 1982). More recently, Zhang and Clark (2011) introduced the first shift-reduce model for CCG, which also showed substantial improvements over the long-established state of the art (Clark and Curran, 2007).

The success of the shift-reduce model (Zhang and Clark, 2011) can be tied to two main contributing factors. First, without any feature locality restrictions, it is able to use a much richer feature set; while intensive feature engineering is inevitable, it has nevertheless delivered an effective and conceptually simpler alternative for both parameter estimation and inference. Second, it couples beam search with global optimization (Collins, 2002; Collins and Roark, 2004; Zhang and Clark, 2008), which makes it less prone to search errors than fully greedy models (Huang et al., 2012).

In this paper, we present a neural architecture for shift-reduce CCG parsing based on long short-term memories (LSTMs; Hochreiter and Schmidhuber, 1997). Our model is inspired by Dyer et al. (2015), in which we explicitly linearize the complete history of parser states in an incremental fashion by requiring no feature engineering (Zhang and Clark, 2011; Xu et al., 2014), and no atomic feature sets (Chen and Manning, 2014). However, a key difference is that we achieve this linearization without relying on any additional control operations or compositional tree structures (Socher et al., 2010; Socher et al., 2011; Socher et al., 2013), both of which are vital in the architecture of Dyer et al. (2015). Crucially, unlike the sequence-to-sequence transduction model of Vinyals et al. (2015), which primarily conditions on the input words, our model is sensitive to all aspects of the parsing history, including arbitrary positions in the input.

As another contribution, we present a global LSTM parsing model by adapting an expected F-measure loss (Xu et al., 2016). As well as naturally incorporating beam search during training, this loss optimizes the model towards the final evaluation metric (Goodman, 1996; Smith and Eisner, 2006; Auli and Lopez, 2011b), allowing it to learn shift-reduce action sequences that lead to parses with high expected F-scores. We further show the globally optimized model can be leveraged with greedy inference, resulting in a deterministic parser as accurate

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1754–1764,
Austin, Texas, November 1-5, 2016.

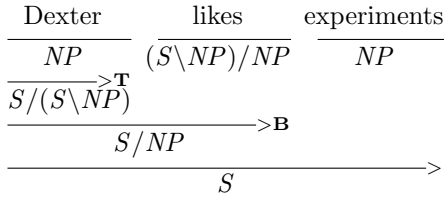


Figure 1: A CCG derivation, in which each point corresponds to the result of a shift-reduce action. In this example, composition (**B**) and application (>) are re actions, and type-raising (**T**) is a un action.

as its beam-search counterpart.

On standard CCGBank tests, we clearly outperform all previous shift-reduce CCG parsers; and by combining the parser with an attention-based LSTM supertagger (§4), we obtain further significant improvements (§5).

2 Shift-Reduce CCG Parsing

CCG is strongly lexicalized by definition. A CCG grammar extracted from CCGBank (Hockenmaier and Steedman, 2007) contains over 400 lexical types and over 1,500 non-terminals (Clark and Curran, 2007), which is an order of magnitude more than those of a typical CFG parser. This lexicalized nature raises another unique challenge for parsing—any parsing model for CCG needs to perform lexical disambiguation. This is true even in the approach of Fowler and Penn (2010), in which a context-free cover grammar extracted from CCGBank is used to parse CCG. Indeed, as noted by Auli and Lopez (2011a), the search problem for CCG parsing is equivalent to finding an optimal derivation in the weighted intersection of a regular language (generated by the supertagger) and a mildly context-sensitive language (generated by the parser), which can quickly become expensive.

The shift-reduce paradigm (Aho and Ullman, 1972; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) applied to CCG (Zhang and Clark, 2011) presents a more elegant solution to this problem by allowing the parser to conduct lexical assignment "incrementally" as a complete parse is being built by the decoder. This is not possible with a chart-based parser, in which complete derivations must be built first. Therefore, a shift-reduce parser is able to consider a much larger set of categories per word for a given input, achieving higher lexical assignment accuracy than the C&C parser (Clark and Curran, 2007), even with the same supertagging model (Zhang and Clark, 2011; Xu et al., 2014).

In our parser, we follow this strategy and adopt the Zhang and Clark (2011) style shift-reduce transition system, which assumes a set of lexical categories has been assigned to each word using a supertagger (Bangalore and Joshi, 1999; Clark and Curran, 2004). Parsing then proceeds by applying a sequence of actions to transform the input maintained on a *queue*, into partially constructed derivations, kept on a *stack*, until the queue and available actions on the stack are both exhausted. At each time step, the parser can choose to *shift* (sh) one of the lexical categories of the front word onto the stack, and remove that word from the queue; *reduce* (re) the top two subtrees on the stack using a CCG rule, replacing them with the resulting category; or take a *unary* (un) action to apply a CCG type-raising or type-changing rule to the stack-top element. For example, the deterministic sequence of shift-reduce actions that builds the derivation in Fig.1 is: sh $\Rightarrow NP$, un $\Rightarrow S/(S\backslash NP)$, sh $\Rightarrow (S\backslash NP)/NP$, re $\Rightarrow S/NP$, sh $\Rightarrow NP$ and re $\Rightarrow S$, where we use $\Rightarrow$ to indicate the CCG category produced by an action.[1]

3 LSTM Shift-Reduce Parsing

3.1 LSTM

Recurrent neural networks (RNNs; e.g., see Elman, 1990) are factored into an input layer $\mathbf{x}_t$ and a hidden state (layer) $\mathbf{h}_t$ with recurrent connections, and they can be represented by the following recurrence:

$$\mathbf{h}_t = \Phi_\theta(\mathbf{x}_t, \mathbf{h}_{t-1}), \quad (1)$$

where $\mathbf{x_t}$ is the current input, $\mathbf{h}_{t-1}$ is the previous hidden state and Φ is a set of affine transformations parametrized by θ. Here, we use a variant of RNN referred to as LSTMs, which augment Eq. 1 with a cell state, $\mathbf{c}_t$, s.t.

$$\mathbf{h}_t, \mathbf{c}_t = \Phi_\theta(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}). \quad (2)$$

Compared with conventional RNNs, this extra facility gives LSTMs more persistent memories over

[1] Our parser models normal-form derivations (Eisner, 1996) in CCGBank. However, unlike Zhang and Clark (2011), derivations are not restricted to be normal-form during inference.

longer time delays and makes them less susceptible to the vanishing gradient problem (Bengio et al., 1994). Hence, they are better at modeling temporal events that are arbitrarily far in a sequence.

Several extensions to the vanilla LSTM have been proposed over time, each with a modified instantiation of Φ_θ that exerts refined control over e.g., whether the cell state could be reset (Gers et al., 2000) or whether extra connections are added to the cell state (Gers and Schmidhuber, 2000). Our instantiation is as follows for all LSTMs:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \\
&\quad \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

where σ is the sigmoid activation and $\odot$ is the element-wise product.

In addition to unidirectional LSTMs that model an input sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ in a strict left-to-right order, we also use bidirectional LSTMs (Graves and Schmidhuber, 2005) (BLSTMs), which read the input from both directions with two independent LSTMs. At each step, the forward hidden state $\mathbf{h}_t$ is computed using Eq. 2 for $t = (0, 1, \ldots, n-1)$; and the backward hidden state $\hat{\mathbf{h}}_t$ is computed similarly but from the reverse direction for $t = (n-1, n-2, \ldots, 0)$. Together, the two hidden states at each step t capture both past and future contexts, and the representation for each $\mathbf{x}_t$ is obtained as the concatenation $[\mathbf{h}_t; \hat{\mathbf{h}}_t]$.

3.2 Embeddings

The neural network model employed by Chen and Manning (2014), and followed by a number of other parsers (Weiss et al., 2015; Zhou et al., 2015; Ambati et al., 2016; Andor et al., 2016; Xu et al., 2016) allows higher-order feature conjunctions to be automatically discovered from a set of dense feature embeddings. However, a set of atomic feature templates, which are only sensitive to contexts from the top few elements on the stack and queue are still needed to dictate the choice of these embeddings. Instead, we dispense with such templates and seek to design a model that is sensitive to both local and non-local contexts, on both the stack and queue.

input: $w_0 \ldots w_{n-1}$

axiom: $0 : (0, \epsilon, \beta, \phi)$

goal: $2n - 1 + \mu : (n, \delta, \epsilon, \Delta)$

$$\frac{t : (j, \delta, x_{w_j}|\beta, \Delta)}{t+1 : (j+1, \delta|x_{w_j}, \beta, \Delta)} \quad (\mathsf{sh}; 0 \le j < n)$$

$$\frac{t : (j, \delta|s_1|s_0, \beta, \Delta)}{t+1 : (j, \delta|x, \beta, \Delta \cup \langle x \rangle))} \quad (\mathsf{re}; s_1 s_0 \to x)$$

$$\frac{t : (j, \delta|s_0, \beta, \Delta)}{t+1 : (j, \delta|x, \beta, \Delta)} \quad (\mathsf{un}; s_0 \to x)$$

Figure 2: The shift-reduce deduction system. For the sh deduction, x_{w_j} denotes an available lexical category for w_j; for re, $\langle x \rangle$ denotes the set of dependencies on x.

Consequently, embeddings represent atomic input units that are added to our parser and are preserved throughout parsing. In total we use four types of embeddings, namely, word, CCG category, POS and action, where each has an associated look-up table that maps a string of that type to its embedding. The look-up table for words is $\mathcal{L}_w \in \mathbb{R}^{k \times |w|}$, where k is the embedding dimension and $|w|$ is the size of the vocabulary. Similarly, we have look-up tables for CCG categories, $\mathcal{L}_c \in \mathbb{R}^{l \times |c|}$, for the three types of actions, $\mathcal{L}_a \in \mathbb{R}^{m \times 3}$, and for POS tags, $\mathcal{L}_p \in \mathbb{R}^{n \times |p|}$.

3.3 Model

Parser. Fig. 2 shows the deduction system of our parser.[2] We denote each parse item as $(j, \delta, \beta, \Delta)$, where j is the positional index of the word at the front of the queue, δ is the stack (with its top element s_0 to the right), and β is the queue (with its top element w_j to the left) and Δ is the set of CCG dependencies realized for the input consumed so far. Each item is also associated with a step indicator t, signifying the number of actions applied to it and the goal is reached in $2n - 1 + \mu$ steps, where μ is the total number of un actions. We also define each action in our parser as a 4-tuple $(\tau_t, c_t, w_{c_t}, p_{w_{c_t}})$, where $\tau_t \in \{\mathsf{sh}, \mathsf{re}, \mathsf{un}\}$ for $t \ge 1$, c_t is the resulting category of τ_t, and w_{c_t} is the head word attached to

[2] We assume greedy inference unless otherwise stated.

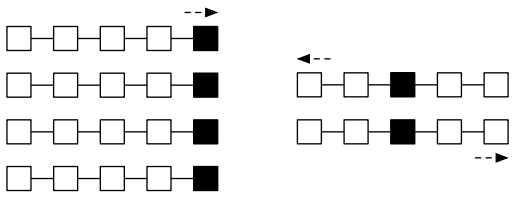


Figure 3: An example representation for a parse item at time step t, with the 4 unidirectional LSTMs (left) and the bidirectional LSTM (right). The shaded cells on the left represent $\boldsymbol{\delta}_t = [\mathbf{h}_t^{\text{U}}; \mathbf{h}_t^{\text{V}}; \mathbf{h}_t^{\text{X}}; \mathbf{h}_t^{\text{Y}}]$ (Eq. 3); and the shaded cells on the right represent $\mathbf{w}_j = [\mathbf{h}_j^{\text{W}}; \hat{\mathbf{h}}_j^{\text{W}}]$.

c_t with $p_{w_{c_t}}$ being its POS tag.[3]

LSTM model. LSTMs are designed to handle time-series data, in a purely sequential fashion; and we try to exploit this fact by completely linearizing all aspects of the parsing history. Concretely, we factor the model as five LSTMs, comprising four unidirectional ones, denoted as U, V, X and Y, and an additional BLSTM, denoted as W (Fig. 3). Before parsing each sentence, we feed W with the complete input (padded with a special embedding $\perp$ as the end of sentence token); and we use $\mathbf{w}_j = [\mathbf{h}_j^{\text{W}}; \hat{\mathbf{h}}_j^{\text{W}}]$ to represent w_j in subsequent steps.[4] We also add $\perp$ to the other 4 unidirectional LSTMs as initialization.

Given this factorization, the stack representation for a parse item $(j, \delta, \beta, \Delta)$ at step t, for $t \geq 1$, is obtained as

$$\boldsymbol{\delta}_t = [\mathbf{h}_t^{\text{U}}; \mathbf{h}_t^{\text{V}}; \mathbf{h}_t^{\text{X}}; \mathbf{h}_t^{\text{Y}}], \tag{3}$$

and together with $\mathbf{w}_j$, $[\boldsymbol{\delta}_t; \mathbf{w}_j]$ gives a representation for the parse item. For the axiom item, we represent it as $[\boldsymbol{\delta}_0; \mathbf{w}_0]$, where $\boldsymbol{\delta}_0 = [\mathbf{h}_\perp^{\text{U}}; \mathbf{h}_\perp^{\text{V}}; \mathbf{h}_\perp^{\text{X}}; \mathbf{h}_\perp^{\text{Y}}]$ is a representation for its stack.

Each time the parser applies an action $(\tau_t, c_t, w_{c_t}, p_{w_{c_t}})$, we update the model by adding the embedding of τ_t, denoted as $\mathcal{L}_a(\tau_t)$, onto U, and adding the other three embeddings of the action 4-tuple, namely $\mathcal{L}_c(c_t)$, $\mathcal{L}_w(w_{c_t})$ and $\mathcal{L}_p(p_{w_{c_t}})$, onto V, X and Y respectively.

To predict the next action, we first derive an action hidden layer $\mathbf{b}_t$, by passing the parse item representation $[\boldsymbol{\delta}_t; \mathbf{w}_j]$ through an affine transformation, s.t.

$$\mathbf{b}_t = f(\mathbf{B}[\boldsymbol{\delta}_t; \mathbf{w}_j] + \mathbf{r}), \tag{4}$$

where $\mathbf{B}$ is a parameter matrix of the model, $\mathbf{r}$ is a bias vector and f is a ReLU non-linearity (Nair and Hinton, 2010). Then we apply another affine transformation (with $\mathbf{A}$ as the weights and $\mathbf{s}$ as the bias) to $\mathbf{b}_t$:

$$\mathbf{a}_t = f(\mathbf{A}\mathbf{b}_t + \mathbf{s}),$$

and obtain the probability of the i^{th} action in $\mathbf{a}_t$ as

$$p(\tau_t^i | \mathbf{b}_t) = \frac{\exp\{\mathbf{a}_t^i\}}{\sum_{\tau_t^k \in \mathcal{T}(\delta_t, \beta_t)} \exp\{\mathbf{a}_t^k\}},$$

where $\mathcal{T}(\delta_t, \beta_t)$ is the set of feasible actions for the current parse item, and $\tau_t^i \in \mathcal{T}(\delta_t, \beta_t)$.

3.4 Derivations and Dependency Structures

Our model naturally linearizes CCG derivations "incrementally" following their post-order traversals. As such, the four unidirectional LSTMs always have the same number of steps; and at each step, the concatenation of their hidden states (Eq. 3) represents a point in a CCG derivation (i.e., an action 4-tuple). Due to the large amount of flexibility in how dependencies are realized in CCG (Hockenmaier, 2003; Clark and Curran, 2007), and in line with most existing CCG parsing models, including dependency models, we have chosen to model CCG derivations, rather than dependency structures.[5] We also hypothesize that tree structures are not necessary for the current model, since they are already implicit in the linearized derivations; and similarly, we have found the action embeddings to be nonessential (§5.2).

3.5 Training

As a baseline, we first train a greedy model, in which we maximize the log-likelihood of each target action in the training data. More specifically, let $(\tau_1^g, \ldots, \tau_{T_n}^g)$ be the gold-standard action sequence for a training sentence n, a cross-entropy criterion is used to obtain the error gradients, and for each sentence, training involves minimizing

$$L(\theta) = -\log \prod_{t=1}^{T_n} p(\tau_t^g | \mathbf{b}_t) = -\sum_{t=1}^{T_n} \log p(\tau_t^g | \mathbf{b}_t),$$

where θ is the set of all parameters in the model.

[3] In case of multiple heads, we always choose the first one.

[4] Word and POS embeddings are concatenated at each input position j, for $0 \leq j < n$; and $\mathbf{w}_n = [\mathbf{h}_\perp^{\text{W}}; \hat{\mathbf{h}}_\perp^{\text{W}}]$.

[5] Most CCG dependency models (e.g., see Clark and Curran (2007) and Xu et al. (2014)) model CCG derivations with dependency features.

As other greedy models (e.g., see Chen and Manning (2014) and Dyer et al. (2015)), our greedy model is *locally* optimized, and suffer from the label bias problem (Andor et al., 2016). A partial solution to this is to use beam search at test time, thereby recovering higher scoring action sequences that would otherwise be unreachable with fully greedy inference. In practice, this has limited effect (Table 2), and a number of more principled solutions have been recently proposed to derive *globally* optimized models during training (Watanabe and Sumita, 2015; Weiss et al., 2015; Zhou et al., 2015; Andor et al., 2016). Here, we extend our greedy model into a global one by adapting the expected F-measure loss of Xu et al. (2016). To our best knowledge, this is the first attempt to train a globally optimized LSTM shift-reduce parser.

Let $\theta = \{\mathbf{U}, \mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{W}, \mathbf{B}, \mathbf{A}\}$ be the weights of the baseline greedy model,[6] we initialize the weights of the global model, which has the same architecture as the baseline, to θ, and we reoptimize θ in multiple training epochs as follows:

1. Pick a sentence x_n from the training set, decode it with beam search, and generate a k-best list of output parses with the *current* θ, denoted as $\Lambda(x_n)$.[7]

2. For each parse y_i in $\Lambda(x_n)$, compute its sentence-level F1 using the set of dependencies in the Δ field of its parse item. In addition, let $|y_i|$ be the total number of actions that derived y_i and $s_\theta(y_i^j)$ be the softmax action score of the j^{th} action, given by the LSTM model. Compute the log-linear score of its action sequence as $\rho(y_i) = \sum_{j=1}^{|y_i|} \log s_\theta(y_i^j)$.

3. Compute the negative expected F1 objective (defined below) for x_n and minimize it using stochastic gradient descent (maximizing expected F1). Repeat these three steps for the remaining sentences.

[6] We use boldface letters to designate the weights of the corresponding LSTMs, and omit bias terms for brevity.

[7] As in Xu et al. (2016), we did not preset k, and found $k = 11.06$ on average with a beam size of 8 that we used for this training.

More formally, the loss $J(\theta)$, is defined as

$$\begin{aligned} J(\theta) &= -\text{XF1}(\theta) \\ &= -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\text{F1}(\Delta_{y_i}, \Delta_{x_n}^G), \end{aligned}$$

where $\text{F1}(\Delta_{y_i}, \Delta_{x_n}^G)$ is the sentence level F1 of the parse derived by y_i, with respect to the gold-standard dependency structure $\Delta_{x_n}^G$ of x_n; $p(y_i|\theta)$ is the normalized probability score of the action sequence y_i, computed as

$$p(y_i|\theta) = \frac{\exp\{\gamma\rho(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\gamma\rho(y)\}},$$

where γ is a parameter that sharpens or flattens the distribution (Tromble et al., 2008).[8] Different from the maximum-likelihood objective, XF1 optimizes the model on a sequence level and towards the final evaluation metric, by taking into account all action sequences in $\Lambda(x_n)$.

4 Attention-Based LSTM Supertagging

In addition to the size of the label space, supertagging is difficult because CCG categories can encode long-range dependencies and tagging decisions frequently depend on non-local contexts. For example, in *He went to the zoo with a cat*, a possible category for *with*, $(S\backslash NP)\backslash(S\backslash NP)/NP$, depends on the word *went* further back in the sentence.

Recently a number of RNN models have been proposed for CCG supertagging (Xu et al., 2015; Lewis et al., 2016; Vaswani et al., 2016; Xu et al., 2016), and such models show dramatic improvements over non-recurrent models (Lewis and Steedman, 2014b). Although the underlying models differ in their exact architectures, all of them make each tagging decision using only the hidden states at the current input position, and this imposes a potential bottleneck in the model. To mitigate this, we generalize the attention mechanisms of Bahdanau et al. (2015) and Luong et al. (2015), and adapt them to supertagging, by allowing the model to explicitly use hidden states from more than one input positions for tagging each word. Similar to Bahdanau et al. (2015) and Luong et al. (2015), a key feature in

[8] We found $\gamma = 1$ to be a good choice during development.

our model is a soft alignment vector that weights the relative importance of the considered hidden states.

For an input sentence $w_0, w_1, \ldots, w_{n-1}$, we consider $\mathbf{w}_t = [\mathbf{h}_t; \hat{\mathbf{h}}_t]$ (§3.1) to be the representation of the t^{th} word ($0 \leq t < n$, $\mathbf{w}_t \in \mathbb{R}^{2d \times 1}$), given by a BLSTM with a hidden state size d for both its forward and backward layers.[9] Let k be a context window size hyperparameter, we define $\mathbf{H}_t \in \mathbb{R}^{2d \times (k-1)}$ as

$$\mathbf{H}_t = [\mathbf{w}_{t-\lfloor k/2 \rfloor}, \ldots, \mathbf{w}_{t-1}, \mathbf{w}_{t+1}, \ldots, \mathbf{w}_{t+\lfloor k/2 \rfloor}],$$

which contains representations for all words in the size k window except $\mathbf{w}_t$. At each position t, the attention model derives a context vector $\mathbf{c}_t \in \mathbb{R}^{2d \times 1}$ (defined below) from $\mathbf{H}_t$, which is used in conjunction with $\mathbf{w}_t$ to produce an attentional hidden layer:

$$\mathbf{x}_t = f(\mathbf{M}[\mathbf{c}_t; \mathbf{w}_t] + \mathbf{m}), \tag{5}$$

where f is a ReLU non-linearity, $\mathbf{M} \in \mathbb{R}^{g \times 4d}$ is a learned weight matrix, $\mathbf{m}$ is a bias term, and g is the size of $\mathbf{x}_t$. Then $\mathbf{x}_t$ is used to produce another hidden layer (with $\mathbf{N}$ as the weights and $\mathbf{n}$ as the bias):

$$\mathbf{z}_t = \mathbf{N}\mathbf{x}_t + \mathbf{n},$$

and the predictive distribution over categories is obtained by feeding $\mathbf{z}_t$ through a softmax activation.

In order to derive the context vector $\mathbf{c}_t$, we first compute $\mathbf{b}_t \in \mathbb{R}^{(k-1) \times 1}$ from $\mathbf{H}_t$ and $\mathbf{w}_t$ using $\boldsymbol{\alpha} \in \mathbb{R}^{1 \times 4d}$, s.t. the i^{th} entry in $\mathbf{b}_t$ is

$$\mathbf{b}_t^i = \boldsymbol{\alpha}[\mathbf{w}_{T[i]}; \mathbf{w}_t],$$

for $i \in [0, k-1)$, $T = [t-\lfloor k/2 \rfloor, \ldots, t-1, t+1, \ldots, t+\lfloor k/2 \rfloor]$; and $\mathbf{c}_t$ is derived as follows:

$$\mathbf{a}_t = \text{softmax}(\mathbf{b}_t),$$
$$\mathbf{c}_t = \mathbf{H}_t \mathbf{a}_t,$$

where $\mathbf{a}_t$ is the alignment vector. We also experiment with two types of attention reminiscent of the *global* and *local* models in Luong et al. (2015), where the first attends over all input words ($k = n$) and the second over a local window.

It is worth noting that two other works have concurrently tackled supertagging with BLSTM models. In Vaswani et al. (2016), a language model layer is added on top of a BLSTM, which allows embeddings of previously predicted tags to propagate through and influence the pending tagging decision. However, the language model layer is only effective when both scheduled sampling for training (Bengio et al., 2015) and beam search for inference are used. We show our attention-based models can match their performance, with only standard training and greedy decoding. Additionally, Lewis et al. (2016) presented a BLSTM model with two layers of stacking in each direction; and as an internal baseline, we show a non-stacking BLSTM without attention can achieve the same accuracy.

[9] Unlike in the parsing model, POS tags are excluded.

5 Experiments

Dataset and baselines. We conducted all experiments on CCGBank (Hockenmaier and Steedman, 2007) with the standard splits.[10] We assigned POS tags with the C&C POS tagger, and used 10-fold jackknifing for both POS tagging and supertagging. All parsers were evaluated using F1 over labeled CCG dependencies.

For supertagging, the baseline models are the RNN model of Xu et al. (2015), the bidirectional RNN (BRNN) model of Xu et al. (2016), and the BLSTM supertagging models in Vaswani et al. (2016) and Lewis et al. (2016). For parsing experiments, we compared with the global beam-search shift-reduce parsers of Zhang and Clark (2011) and Xu et al. (2014). One neural shift-reduce CCG parser baseline is Ambati et al. (2016), which is a beam-search shift-reduce parser based on Chen and Manning (2014) and Weiss et al. (2015); and the others are the RNN shift-reduce models in Xu et al. (2016). Additionally, the chart-based C&C parser was included by default.

Model and training parameters.[11] All our LSTM models are non-stacking with a single layer.[12] For the supertagging models, the LSTM

[10] Training: Sections 02-21 (39,604 sentences). Development: Section 00 (1,913 sentences). Test: Section 23 (2,407 sentences).

[11] We implemented all models using the CNN toolkit: `https://github.com/clab/cnn`.

[12] The BLSTMs have a single layer in each direction. We experimented with 2 layers in all models during development and found negligible improvements.

Model	Dev	Test
C&C	91.50	92.02
Xu et al. (2015)	93.07	93.00
Xu et al. (2016)	93.49	93.52
Lewis et al. (2016)	94.1	94.3
Vaswani et al. (2016)	94.08	-
Vaswani et al. (2016) +LM +beam	94.24	94.50
BLSTM	94.11	94.29
BLSTM-local	**94.31**	**94.46**
BLSTM-global	94.22	94.42

Table 1: 1-best supertagging results on both the dev and test sets. BLSTM is the baseline model without attention; BLSTM-local and -global are the two attention-based models.

hidden state size is 256, and the size of the attentional hidden layer ($\mathbf{x}_t$, Eq. 5) is 200. All parsing model LSTMs have a hidden state size of 128, and the size of the action hidden layer ($\mathbf{b}_t$, Eq. 4) is 80.

Pretrained word embeddings for all models are 100-dimensional (Turian et al., 2010), and all other embeddings are 50-dimensional. We also pretrained CCG lexical category and POS embeddings on the concatenation of the training data and a Wikipedia dump parsed with C&C.[13] All other parameters were uniformly initialized in $\pm\sqrt{6/(r+c)}$, where r and c are the number of rows and columns of a matrix (Glorot and Bengio, 2010).

For training, we used plain non-minibatched stochastic gradient descent with an initial learning rate $\eta_0 = 0.1$ and we kept iterating in epochs until accuracy no longer increases on the dev set. For all models, a learning rate schedule $\eta_e = \eta_0/(1+\lambda e)$ with $\lambda = 0.08$ was used for $e \geq 11$. Gradients were clipped whenever their norm exceeds 5. Dropout training as suggested by Zaremba et al. (2014), with a dropout rate of 0.3, and an ℓ_2 penalty of 1×10^{-5}, were applied to all models.

5.1 Supertagging Results

Table 1 summarizes 1-best supertagging results. Our baseline BLSTM model without attention achieves the same level of accuracy as Lewis et al. (2016) and the baseline BLSTM model of Vaswani et al. (2016). Compared with the latter, our hidden state size is 50% smaller (256 vs. 512).

For training and testing the local attention model (BLSTM-local), we used an attention window size of 5 (tuned on the dev set), and it gives an improvement of 0.94% over the BRNN supertagger (Xu et al., 2016), achieving an accuracy on par with the beam-search (size 12) model of Vaswani et al. (2016) that is enhanced with a language model. Despite being able to consider wider contexts than the local model, the global attention model (BLSTM-global) did not show further gains, hence we used BLSTM-local for all parsing experiments below.

[13] We used the gensim word2vec toolkit: https://radimrehurek.com/gensim/.

	Supertagger β				
Beam	0.09	0.07	0.06	0.01	0.001
1	86.49	86.52	86.56	86.26	85.80
2	86.55	86.58	86.63	86.39	86.01
8	86.61	86.64	**86.67**	86.40	86.07

Table 2: Tuning beam size and supertagger β on the dev set.

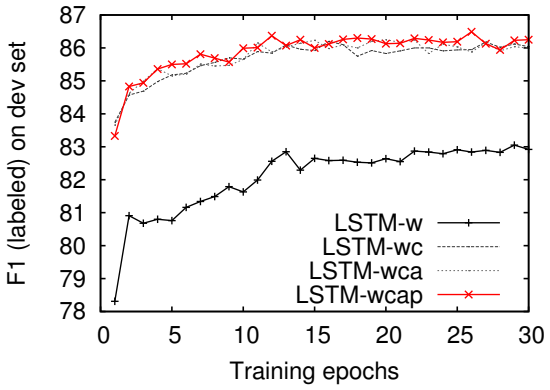
Model	LP	LR	LF	CAT
LSTM-w	**90.13**	76.99	83.05	94.24
LSTM-w+c	89.37	83.25	86.20	94.34
LSTM-w+c+a	89.31	83.39	86.25	94.38
LSTM-w+c+a+p	89.43	**83.86**	**86.56**	**94.47**

Table 3: F1 on dev for all the greedy models.

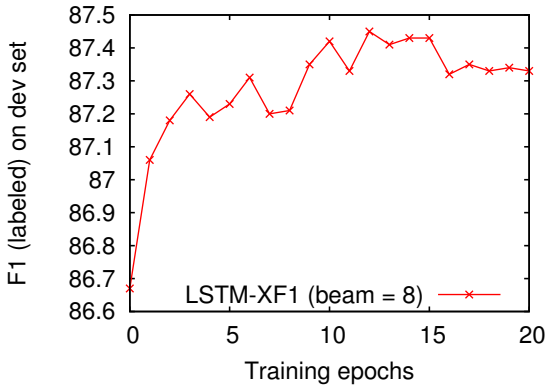
5.2 Parsing Results

All parsers we consider use a supertagger probability cutoff β to prune categories less likely than β times the probability of the best category in a distribution: for the C&C parser, it uses an *adaptive* strategy to backoff to more relaxed β values if no spanning analysis is found given an initial β setting; for all the shift-reduce parsers, fixed β values are used without backing off. Since β determines the derivation space of a parser, it has a large impact on the final parsing accuracy.

For the maximum-likelihood greedy model, we found using a small β value (bigger ambiguity) for training significantly improved accuracy, and we chose $\beta = 1 \times 10^{-5}$ (5.22 categories per word with jackknifing) via development experiments. This reinforces the findings in a number of other CCG parsers (Clark and Curran, 2007; Auli and Lopez, 2011a; Zhang and Clark, 2011; Lewis and Steedman, 2014a; Xu et al., 2014): even though a more relaxed β increases ambiguity, it leads to more accurate models at test time. On the other hand, we found using smaller β values at test time led to significantly better results (Table 2). And this observation differs from the beam-search models which use the same β value for both training and testing.



(a) Dev F1 of the greedy models



(b) Dev F1 of the XF1 model

Figure 4: Learning curves with dev F-scores for all models.

		Section 00				Section 23			
Model	Beam	LP	LR	LF	CAT	LP	LR	LF	CAT
C&C (normal-form)	-	85.18	82.53	83.83	92.39	85.45	83.97	84.70	92.83
C&C (dependency hybrid)	-	86.07	82.77	84.39	92.57	86.24	84.17	85.19	93.00
Zhang and Clark (2011)	16	87.15	82.95	85.00	92.77	87.43	83.61	85.48	93.12
Xu et al. (2014)	128	86.29	84.09	85.18	92.75	87.03	85.08	86.04	93.10
Ambati et al. (2016)	16	-	-	85.69	93.02	-	-	85.57	92.86
Xu et al. (2016)-greedy	1	88.12	81.38	84.61	93.42	88.53	81.65	84.95	93.57
Xu et al. (2016)-XF1	8	88.20	83.40	85.73	93.56	88.74	84.22	86.42	93.87
LSTM-greedy	1	89.43	83.86	86.56	**94.47**	89.75	84.10	86.83	**94.63**
LSTM-XF1	1	**89.68**	85.29	87.43	94.41	**89.85**	85.51	87.62	94.53
LSTM-XF1	8	89.54	**85.46**	**87.45**	94.39	89.81	**85.81**	**87.76**	94.57

Table 4: Parsing results on the dev (Section 00) and test (Section 23) sets with 100% coverage, with all LSTM models using the BLSTM-local supertagging model. All experiments using auto POS. CAT (lexical category assignment accuracy). LSTM-greedy is the full greedy parser.

The greedy model. Table 3 shows the dev set results for all greedy models, where the four types of embeddings, that is, word (w), CCG category (c), action (a) and POS (p), are gradually introduced. The full model LSTM-w+c+a+p surpasses all previous shift-reduce models (Table 4), achieving a dev set accuracy of 86.56%. Category embeddings (LSTM-w+c) yielded a large gain over using word embeddings alone (LSTM-w); action embeddings (LSTM-w+c+a) provided little improvement, but further adding POS embeddings (LSTM-w+c+a+p) gave noticeable recall (+0.61%) and F1 improvements (+0.36%) over LSTM-w+c. Fig. 4a shows the learning curves, where all models converged in under 30 epochs.

The XF1 model. Table 4 also shows the results for the XF1 models (LSTM-XF1), which use all four types of embeddings. We used a beam size of 8, and a β value of 0.06 for both training and testing (tuned on the dev set); and training took 12 epochs to converge (Fig. 4b), with an F1 of 87.45% on the dev set. Decoding the XF1 model with greedy inference only slightly decreased recall and F1, and this resulted in a highly accurate deterministic parser. On the test set, our XF1 greedy model gives +2.67% F1 improvement over the greedy model in Xu et al. (2016); and the beam-search XF1 model achieves an F1 improvement of +1.34% compared with the XF1 model of Xu et al. (2016).

Model	Dev	Test
LSTM-BRNN	85.86	86.37
LSTM-BLSTM	86.26	86.64
LSTM-greedy	**86.56**	**86.83**

Table 5: Effect of different supertaggers on the full greedy parser. LSTM-greedy is the same parser as in Table 4, which uses the BLSTM-local supertagger.

Model	LP	LR	LF
Xu et al. (2015)	87.68	86.41	87.04
Lewis et al. (2016)	87.7	86.7	87.2
Lewis et al. (2016)*	88.6	87.5	88.1
Vaswani et al. (2016)*	-	-	88.32
Lee et al. (2016)	-	-	**88.7**
LSTM-XF1 (beam = 1)	89.85	85.51	87.62
LSTM-XF1 (beam = 8)	89.81	85.81	87.76

Table 6: Comparison of our XF1 models with chart-based parsers on the test set. $\star$ denotes a tri-trained model and $*$ indicates a different POS tagger.

Effect of the supertagger. To isolate the parsing model from the supertagging model, we first experimented with the BRNN supertagging model as in Xu et al. (2016) for both training and testing the full greedy LSTM parser. Using this supertagger, we still achieved the highest F1 (85.86%) on the dev set (LSTM-BRNN, Table 5) in comparison with all previous shift-reduce models; and an improvement of 1.42% F1 over the greedy model of Xu et al. (2016) was obtained on the test set (Table 4). We then experimented with using the baseline BLSTM supertagging model for parsing (LSTM-BLSTM), and observed the attention-based setup (LSTM-greedy) outperformed it, despite the attention-based supertagger (BLSTM-local) did not give better multi-tagging accuracy. We owe this to the fact that very tight β cutoff values—resulting in almost deterministic supertagging decisions on average—are required by the parser during inference; for instance, BLSTM-local has an average ambiguity of 1.09 on the dev set with $\beta = 0.06$.[14]

Comparison with chart-based models. For completeness and to put our results in perspective, we compare our XF1 models with other CCG parsers in the literature (Table 6): Xu et al. (2015) is the log-linear C&C dependency hybrid model with an RNN supertagger front-end; Lewis et al. (2016) is an LSTM supertagger-factored parser using the A* CCG parsing algorithm of Lewis and Steedman (2014a); Vaswani et al. (2016) combine a BLSTM supertagger with a new version of the C&C parser (Clark et al., 2015) that uses a max-violation percetrpon, which significantly improves over the original C&C models; and finally, a global recursive neural network model with A* decoding (Lee et al., 2016). We note that all these alternative models—with the exception of Xu et al. (2015) and Lewis et al. (2016)—use structured training that accounts for violations of the gold-standard, and we conjecture further improvements for our model are possible by incorporating such mechanisms.[15]

6 Conclusion

We have presented an LSTM parsing model for CCG, with a factorization allowing the linearization of the complete parsing history. We have shown that this simple model is highly effective, with results outperforming all previous shift-reduce CCG parsers. We have also shown global optimization benefits an LSTM shift-reduce model; and contrary to previous findings with the averaged percetpron (Zhang and Clark, 2008), we empirically demonstrated beam-search inference is not necessary for our globally optimized model. For future work, a natural direction is to explore integrated supertagging and parsing in a single neural model (Zhang and Weiss, 2016).

Acknowledgment

I acknowledge the support from the Carnegie Trust for the Universities of Scotland through a Carnegie Scholarship.

References

Anthony Ades and Mark Steedman. 1982. On the order of words. In *Linguistics and philosophy*. Springer.

Alfred Aho and Jeffrey Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2016. Shift-reduce CCG parsing using neural network models. In *Proc. of NAACL (Volume 2)*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Michael Auli and Adam Lopez. 2011a. A comparison of loopy belief propagation and dual decomposition for

[14] All β cutoffs were tuned on the dev set; for BRNN, we found the same β settings as in Xu et al. (2016) to be optimal; for BLSTM, $\beta = 4 \times 10^{-5}$ for training (with an ambiguity of 5.27) and $\beta = 0.02$ for testing (with an ambiguity of 1.17).

[15] Our XF1 training considers shift-reduce action sequences, but not violations of the gold-standard (e.g., see Huang et al. (2012), Watanabe and Sumita (2015), Zhou et al. (2015) and Andor et al. (2016)).

integrated CCG supertagging and parsing. In *Proc. of ACL*.

Michael Auli and Adam Lopez. 2011b. Training a log-linear parser with loss functions via softmax-margin. In *Proc. of EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. In *Computational linguistics*. MIT Press.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. In *Neural Networks, IEEE Transactions on*. IEEE.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. of NIPS*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Stephen Clark and James Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proc. of COLING*.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. In *Computational Linguistics*. MIT Press.

Stephen Clark, Darren Foong, Luana Bulat, and Wenduan Xu. 2015. The Java version of the C&C parser. Technical report, University of Cambridge Computer Laboratory.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proc. of ACL*.

Jeffrey Elman. 1990. Finding structure in time. In *Cognitive science*. Elsevier.

Timothy Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proc. of ACL*.

Felix Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks*. IEEE.

Felix Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. In *Neural computation*. MIT Press.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Neural Networks*. Elsevier.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*. MIT Press.

Julia Hockenmaier and Mark Steedman. 2007. CCGBank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. In *Computational Linguistics*. MIT Press.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. of NAACL*.

Marco Kuhlmann and Giorgio Satta. 2014. A new parsing algorithm for Combinatory Categorial Grammar. In *Transactions of the Association for Computational Linguistics*. ACL.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Proc. of EMNLP*.

Mike Lewis and Mark Steedman. 2014a. A* CCG parsing with a supertag-factored model. In *Proc. of EMNLP*.

Mike Lewis and Mark Steedman. 2014b. Improved CCG parsing with semi-supervised supertagging. In *Transactions of the Association for Computational Linguistics*. ACL.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proc. of NAACL*.

Minh-Thang Luong, Hieu Pham, and Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICML*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of COLING*.

David Smith and Jason Eisner. 2006. Minimum-risk annealing for training log-linear models. In *Proc. of COLING-ACL*.

Richard Socher, Christopher Manning, and Andrew Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In

Proc. of the NIPS Deep Learning and Unsupervised Feature Learning Workshop.

Richard Socher, Cliff Lin, Christopher Manning, and Andrew Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proc. of ICML*.

Richard Socher, John Bauer, Christopher Manning, and Andrew Ng. 2013. Parsing with compositional vector grammars. In *Proc. of ACL*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proc. of EMNLP*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMs. In *Proc. of NAACL (Volume 2)*.

Krishnamurti Vijay-Shanker and David Weir. 1993. Parsing some constrained grammar formalisms. In *Computational Linguistics*. MIT Press.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proc. of ACL*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL*.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-reduce CCG parsing with a dependency model. In *Proc. of ACL*.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. In *Proc. of ACL (Volume 2)*.

Wenduan Xu, Michael Auli, and Stephen Clark. 2016. Expected F-measure training for shift-reduce parsing with recurrent neural networks. In *Proc. of NAACL*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proc. of IWPT*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. In *Proc. of ICLR*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. of EMNLP*.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proc. of ACL*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proc. of ACL*.

An Evaluation of Parser Robustness for Ungrammatical Sentences

Homa B. Hashemi
Intelligent Systems Program
University of Pittsburgh
hashemi@cs.pitt.edu

Rebecca Hwa
Computer Science Department
University of Pittsburgh
hwa@cs.pitt.edu

Abstract

For many NLP applications that require a parser, the sentences of interest may not be well-formed. If the parser can overlook problems such as grammar mistakes and produce a parse tree that closely resembles the correct analysis for the intended sentence, we say that the parser is *robust*. This paper compares the performances of eight state-of-the-art dependency parsers on two domains of ungrammatical sentences: learner English and machine translation outputs. We have developed an evaluation metric and conducted a suite of experiments. Our analyses may help practitioners to choose an appropriate parser for their tasks, and help developers to improve parser robustness against ungrammatical sentences.

1 Introduction

Previous works have shown that, in general, parser performances degrade when applied to out-of-domain sentences (Gildea, 2001; McClosky et al., 2010; Foster, 2010; Petrov et al., 2010; Foster et al., 2011). If a parser performs reasonably well for a wide range of out-of-domain sentences, it is said to be *robust* (Bigert et al., 2005; Kakkonen, 2007; Foster, 2007).

Sentences that are ungrammatical, awkward, or too casual/colloquial can all be seen as special kinds of out-of-domain sentences. These types of sentences are commonplace for NLP applications, from product reviews and social media analysis to intelligent language tutors and multilingual processing. Since parsing is an essential component for many applications, it is natural to ask: Are some parsers more robust than others against sentences that are not well-formed? Previous works on parser evaluation that focused on accuracy and speed (Choi et al., 2015; Kummerfeld et al., 2012; McDonald and Nivre, 2011; Kong and Smith, 2014) have not taken ungrammatical sentences into consideration.

In this paper, we report a set of empirical analyses of eight leading dependency parsers on two domains of ungrammatical text: English-as-a-Second Language (ESL) learner text and machine translation (MT) outputs. We also vary the types of training sources; the parsers are trained with the Penn Treebank (to be comparable with other studies) and Tweebank, a treebank on tweets (to be a bit more like the test domain) (Kong et al., 2014).

The main contributions of the paper are:

- a metric and methodology for evaluating ungrammatical sentences without referring to a gold standard corpus;
- a quantitative comparison of parser accuracy of leading dependency parsers on ungrammatical sentences; this may help practitioners to select an appropriate parser for their applications; and
- a suite of robustness analyses for the parsers on specific kinds of problems in the ungrammatical sentences; this may help developers to improve parser robustness in the future.

2 Evaluation of Parser Robustness

Parser evaluation for ungrammatical texts presents some domain-specific challenges. The typical approach to evaluate parsers is to compare parser out-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1765–1774,
Austin, Texas, November 1-5, 2016.

puts against manually annotated gold standards. Although there are a few small semi-manually constructed treebanks on learner texts (Geertzen et al., 2013; Ott and Ziai, 2010) or tweets (Daiber and van der Goot, 2016), their sizes make them unsuitable for the evaluation of parser robustness. Moreover, some researchers have raised valid questions about the merit of creating a treebank for ungrammatical sentences or adapting the annotation schema (Cahill, 2015; Ragheb and Dickinson, 2012).

A "gold-standard free" alternative is to compare the parser output for each problematic sentence with the parse tree of the corresponding correct sentence. Foster (2004) used this approach over a small set of ungrammatical sentences and showed that parser's accuracy is different for different types of errors. A limitation of this approach is that the comparison works best when the differences between the problematic sentence and the correct sentence are small. This is not the case for some ungrammatical sentences (especially from MT systems). Another closely-related approach is to semi-automatically create treebanks from artificial errors. For example, Foster generated artificial errors to the sentences from the Penn Treebank for evaluating the effect of error types on parsers (Foster, 2007). In another work, Bigert et al. (2005) proposed an unsupervised evaluation of parser robustness based on the introduction of artificial spelling errors in error-free sentences. Kakkonen (2007) adapted a similar method to compare the robustness of four parsers over sentences with misspelled words.

Our proposed evaluation methodology is similar to the "gold-standard free" approach; we compare the parser output for an ungrammatical sentence with the automatically generated parse tree of the corresponding correct sentence. In the next section, we discuss our evaluation metric to address the concerns that some ungrammatical sentences may be very different from their corrected versions. This allows us to evaluate parsers with more realistic data that exhibit a diverse set of naturally occurring errors, instead of artificially generated errors or limited error types.

3 Proposed Evaluation Methodology

For the purpose of robustness evaluation, we take the automatically produced parse tree of a well-formed sentence as "gold-standard" and compare the parser output for the corresponding problematic sentence against it. Even if the "gold-standard" is not perfectly correct in absolute terms, it represents the norm from which parse trees of problematic sentences diverge: if a parser were robust against ungrammatical sentences, its output for these sentences should be similar to its output for the well-formed ones.

Determining the evaluation metric for comparing these trees, however, presents another challenge. Since the words of the ungrammatical sentence and its grammatical counterpart do not necessarily match (an example is given in Figure 1), we cannot use standard metrics such as Parseval (Black et al., 1991). We also cannot use adapted metrics for comparing parse trees of unmatched sentences (e.g., Sparseval (Roark et al., 2006)), because these metrics consider all the words regardless of the mismatches (extra or missing words) between two sentences. This is a problem for comparing ungrammatical sentences to grammatical ones because a parser is unfairly penalized when it assigns relations to extra words and when it does not assign relations to missing words. Since a parser cannot modify the sentence, we do not want to penalize these extraneous or missing relations; on the other hand, we do want to identify cascading effects on the parse tree due to a grammar error. For the purpose of evaluating parser robustness against ungrammatical sentences, we propose a modified metric in which the dependencies connected to unmatched (extra or missing) error words are ignored. A more formal definition is as follows:

- *Shared dependency* is a mutual dependency between two trees;
- *Error-related dependency* is a dependency connected to an extra word[1] in the sentence;
- *Precision* is (# of shared dependencies) / (# of dependencies of the ungrammatical sentence -

[1]The extra word in the ungrammatical sentences is an unnecessary word error, and the extra word in the grammatical sentence is a missing word error.

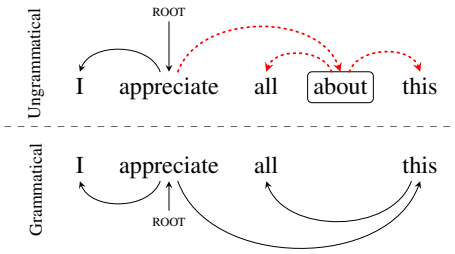


Figure 1: Example of evaluating robustness of an ungrammatical sentence (top) dependency parse tree with its corresponding grammatical sentence (bottom).

# of error-related dependencies of the ungrammatical sentence);

- *Recall* is (# of shared dependencies) / (# of dependencies of the grammatical sentence - # of error-related dependencies of the grammatical sentence); and
- *Robustness* F_1 is the harmonic mean of precision and recall.

Figure 1 shows an example in which the ungrammatical sentence has an unnecessary word, "about", so the three dependencies connected to it are counted as error-related dependencies. The two shared dependencies between the trees result in a precision of $2/(5-3) = 1$, recall of $2/(4-0) = 0.5$, and Robustness F_1 of 66%.

4 Experimental Setup

Our experiments are conducted over a wide range of dependency parsers that are trained on two different treebanks: Penn Treebank (PTB) and Tweebank. We evaluate the robustness of parsers over two datasets that contain ungrammatical sentences: writings of English-as-a-Second language learners and machine translation outputs. We choose datasets for which the corresponding correct sentences are available (or easily reconstructed).

4.1 Parsers

Our evaluation is over eight state-of-the-art dependency parsers representing a wide range of approaches. We use the publicly available versions of each parser with the standard parameter settings.

Malt Parser (Nivre et al., 2007)[2] A greedy transition-based dependency parser. We use LIBLINEAR setting in the learning phase.

Mate Parser v3.6.1 (Bohnet, 2010)[3] A graph-based dependency parser that uses second-order maximum spanning tree.

MST Parser (McDonald and Pereira, 2006)[4] A first-order graph-based parser that searches for maximum spanning trees.

Stanford Neural Network Parser (SNN) (Chen and Manning, 2014)[5] A transition-based parser that uses word embeddings. We use pre-trained word embeddings from Collobert et al. (2011) as recommended by the authors.

SyntaxNet (Andor et al., 2016)[6] A transition-based neural network parser. We use the globally normalized training of the parser with default parameters.

Turbo Parser v2.3 (Martins et al., 2013)[7] A graph-based dependency parser that uses dual decomposition algorithm with third-order features.

Tweebo Parser (Kong et al., 2014)[8] An extension of the Turbo Parser specialized to parse tweets. Tweebo Parser adds a new constraint to the Turbo Parser's integer linear programming to ignore some Twitter tokens from parsing, but also simultaneously uses these tokens as parsing features.

Yara Parser (Rasooli and Tetreault, 2015)[9] A transition-based parser that uses beam search training and dynamic oracle.

4.2 Data

We train all the parsers using two treebanks and test their robustness over two ungrammatical datasets.

4.2.1 Parser Training Data

Penn Treebank (PTB) We follow the standard splits of Penn Treebank, using section 2-21 for training, section 22 for development, and section 23 for

[2] www.maltparser.org
[3] code.google.com/p/mate-tools
[4] seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html
[5] nlp.stanford.edu/software/nndep.shtml
[6] github.com/tensorflow/models/tree/master/syntaxnet
[7] www.cs.cmu.edu/~ark/TurboParser
[8] github.com/ikekonglp/TweeboParser
[9] github.com/yahoo/YaraParser

testing. We transform bracketed sentences from PTB into dependency formats using Stanford Basic Dependency representation (De Marneffe et al., 2006) from Stanford parser v3.6. We assign POS tags to the training data using Stanford POS tagger (Toutanova et al., 2003) with ten-way jackknifing (with 97.3% accuracy).

Tweebank Tweebank is a Twitter dependency corpus annotated by non-experts containing 929 tweets (Kong et al., 2014). Kong et al. (2014) used 717 of tweets for training and 201 for test[10]. We follow the same split in our experiments. We use pretrained POS tagging model of Kong et al. (2014) (with 92.8% accuracy) over the tweets.

The elements in tweets that have no syntactic function (such as hashtags, URLs and emoticons) are annotated as unselected tokens (no tokens as the heads). In order to be able to use Tweebank in other parsers, we link the unselected tokens to the wall symbol (i.e. root as the heads). This assumption will generate more arcs from the root, but since we use the same evaluation setting for all the parsers, the results are comparable. We evaluate the accuracy of the trained parser on Tweebank with the unlabeled attachment F_1 score (same procedure as Kong et al. (2014)).

4.2.2 Robustness Test Data

To test the robustness of parsers, we choose two datasets of ungrammatical sentences for which their corresponding correct sentences are available. For a fair comparison, we automatically assign POS tags to the test data. When parsers are trained on PTB, we use the Stanford POS tagger (Toutanova et al., 2003). When parsers are trained on Tweebank, we coarsen POS tags to be compatible with the Twitter POS tags using the mappings specified by Gimpel et al. (2011).

English-as-a-Second Language corpus (ESL) For the ungrammatical sentences, we use the First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011) that contains the writings of English as a second language learners and their corresponding error corrections. Given the errors and their corrections, we can easily reconstruct the corrected version of each ungrammatical ESL sentence. From this corpus, we randomly select 10,000 sentences with at least one error; there are 4954 with one error; 2709 with two errors; 1290 with three; 577 with four; 259 with five; 111 with six; and 100 with 7+ errors.

Machine Translation corpus (MT) Machine translation outputs are another domain of ungrammatical sentences. We use the LIG (Potet et al., 2012) which contains 10,881 and LISMI's TRACE corpus[11] which contains 6,693 French-to-English machine translation outputs and their human post-editions. From these corpora, we randomly select 10,000 sentences with at least edit distance one (upon words) with their human-edited sentence. The distribution of the number of sentences with their edit distances from 1 to 10+ is as follows (beginning with 1 edit distance and ending with 10+): 674; 967; 1019; 951; 891; 802; 742; 650; 547; and 2752.

4.3 Evaluation Metric

In the robustness evaluation metric (Section 3), shared dependencies and error-related dependencies are detected based on alignments between words in the ungrammatical and grammatical sentences. We find the alignments in the FCE and MT data in a slightly different way. In the FCE dataset, in which the error words are annotated, the grammatical and ungrammatical sentences can easily be aligned. In the MT dataset, we use the TER (Translation Error Rate) tool (default settings)[12] to find alignments.

In our experiments, we present unlabeled robustness F_1 micro-averaged across the test sentences. We consider punctuations when parsers are trained with the PTB data, because punctuations can be a source of ungrammaticality. However, we ignore punctuations when parsers are trained with the Tweebank data, because punctuations are not annotated in the tweets with their dependencies.

5 Experiments

The experiments aim to address the following questions given separate training and test data:

1. How do parsers perform on erroneous sentences? (Section 5.1)

[10] github.com/ikekonglp/TweeboParser/tree/master/Tweebank

[11] anrtrace.limsi.fr/trace_postedit.tar.bz2

[12] www.cs.umd.edu/~snover/tercom

Parser	Train on PTB §1-21			Train on Tweebank$_{train}$		
	UAS	Robustness F_1		UAF_1	Robustness F_1	
	PTB §23	ESL	MT	Tweebank$_{test}$	ESL	MT
Malt	*89.58*	93.05	76.26	77.48	**94.36**	80.66
Mate	**93.16**	93.24	77.07	76.26	91.83	75.74
MST	91.17	*92.80*	76.51	73.99	92.37	77.71
SNN	90.70	93.15	74.18	*53.4*	88.90	*71.54*
SyntaxNet	93.04	93.24	76.39	75.75	*88.78*	**81.87**
Turbo	92.84	**93.72**	**77.79**	79.42	93.28	78.26
Tweebo	-	-	-	**80.91**	93.39	79.47
Yara	93.09	93.52	*73.15*	78.06	93.04	75.83

Table 1: Parsers' performance in terms of accuracy and robustness. The best result in each column is given in bold, and the worst result is in italics.

2. To what extent is each parser negatively impacted by the increase in the number of errors in sentences? (Section 5.2)

3. To what extent is each parser negatively impacted by the interactions between multiple errors? (Section 5.3)

4. What types of errors are more problematic for parsers? (Section 5.4)

5.1 Overall Accuracy and Robustness

The overall performances of all parsers are shown in Table 1. Note that the Tweebo Parser's performance is not trained on the PTB because it is a specialization of the Turbo Parser, designed to parse tweets. Table 1 shows that, for both training conditions, the parser that has the best robustness score in the ESL domain has also high robustness for the MT domain. This suggests that it might be possible to build robust parsers for multiple ungrammatical domains. The training conditions do matter – Malt performs better when trained from Tweebank than from the PTB. In contrast, Tweebank is not a good fit with the neural network parsers due to its small size. Moreover, SNN uses pre-trained word embeddings, and 60% of Tweebank tokens are missing.

Next, let us compare parsers within each train/test configuration for their relative robustness. When trained on the PTB, all parsers are comparably robust on ESL data, while they exhibit more differences on the MT data, and, as expected, everyone's performance is much lower because MT errors are more diverse than ESL errors. We expected that by training on Tweebank, parsers will perform better on ESL data (and maybe even MT data), since Tweebank is arguably more similar to the test domains than the PTB; we also expected Tweebo to outperform others. The results are somewhat surprising. On the one hand, the highest parser score increased from 93.72% (Turbo trained on PTB) to 94.36% (Malt trained on Tweebank), but the two neural network parsers performed significantly worse, most likely due to the small training size of Tweebank. Interestingly, although SyntaxNet has the lowest score on ESL, it has the highest score on MT, showing promise in its robustness.

5.2 Parser Robustness by Number of Errors

To better understand the overall results, we further breakdown the test sentences by the number of errors each contains. Our objectives are: (1) to observe the speed with which the parsers lose their robustness as the sentences become more error-prone; (2) to determine whether some parsers are more robust than others when handling noisier data.

Figure 2 presents four graphs, plotting robustness F_1 scores against the number of errors for all parsers under each train/test configuration. In terms of the parsers' general degradation of robustness, we observe that: 1) parsing robustness degrades faster with the increase of errors for the MT data than the ESL data; 2) training on the PTB led to a more similar behavior between the parsers than when training on Tweebank; 3) training on Tweebank does help some parsers to be more robust against many errors.

In terms of relative robustness between parsers,

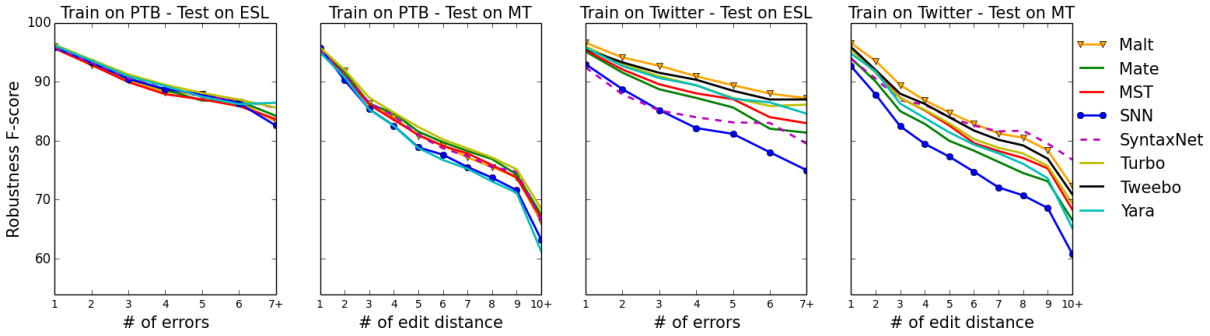


Figure 2: Variation in parser robustness as the number of errors in the test sentences increases.

we observe that Malt, Turbo, and Tweebo parsers are more robust than others given noisier inputs. The SNN parser is a notable outlier when trained on Tweebank due to insufficient training examples.

5.3 Impact of Error Distances

This experiment explores the impact of the interactivity of errors. We assume that errors have more interaction if they are closer to each other, and less interaction if they are scattered throughout the sentence. We define "near" to be when there is at most 1 word between errors and "far" to be when there are at least 6 words between errors. We expect all parsers to have more difficulty on parsing sentences when their errors have more interaction, but how do the parsers compare against each other? We conduct this experiment using a subset of sentences that have exactly three errors; we compare parser robustness when these three errors are near each other with the robustness when the errors are far apart.

Table 2 presents the results as a collection of shaded bars. This aims to give an at-a-glance visualization of the outcomes. In this representation, all parsers with the same train data and *test domain* (including both the *near* and *far* sets) are treated as one group. The top row specifies the lowest score of all parsers on both test sets; the bottom row specifies the highest score. The shaded area of each bar indicates the *relative robustness* of each parser with respect to the lowest and highest scores of the group. An empty bar indicates that it is the least robust (corresponding to the lowest score in the top row); a fully shaded bar means it is the most robust (corresponding to the highest score in the bottom row). Consider the left-most box, in which parsers trained on PTB and tested on ESL are compared. In this group[13], Yara (near) is the least robust parser with a score of $F_1 = 87.3\%$, while SNN (far) is the most robust parser with a score of $F_1 = 93.4\%$; as expected, all parsers are less robust when tested on sentences with near errors than far errors, but they do exhibit relative differences: Turbo parser seems most robust in this setting. Turbo parser's lead in handling error interactivity holds for most of the other train/test configurations as well; the only exception is for Tweebank/MT, where SyntaxNet and Malt are better. Compared to ESL data, near errors in MT data are more challenging for all parsers; when trained on PTB, most are equally poor, except for Yara, which has the worst score (79.1%), even though it has the highest score when the errors are far apart (91.5%). Error interactivity has the most effect on Yara parser in all but one train/test configuration (Tweebank/ESL).

5.4 Impact of Error Types

In the following experiments, we examine the impact of different error types. To remove the impact due to interactivity between multiple errors, these studies use a subset of sentences that have only one error. Although all parsers are fairly robust for sentences containing one error, our focus here is on the relative performances of parsers over different error types: We want to see whether some error types are more problematic for some parsers than others.

5.4.1 Impact of grammatical error types

The three main grammar error types are replacement (a word need replacing), missing (a word missing), and unnecessary (a word is redundant). Our

[13] As previously explained, Tweebo is not trained on PTB, so it has no bars associated with it.

Parser	Train on PTB §1-21				Train on Tweebank$_{train}$			
	ESL		MT		ESL		MT	
	Near	Far	Near	Far	Near	Far	Near	Far
min	87.3 (Yara)		79.1 (Yara)		82.4 (SyntaxNet)		80.6 (SNN)	
Malt								
Mate								
MST								
SNN								
SyntaxNet								
Turbo								
Tweebo								
Yara								
max	93.4 (SNN)		91.5 (Yara)		94.5 (Malt)		94.4 (Malt)	

Table 2: Parser performance on test sentences with three near and three far errors. Each box represents one train/test configuration for all parsers and error types. The bars within indicate the level of robustness scaled to the lowest score (empty bar) and highest score (filled bar) of the group.

Parser	Train on PTB §1-21						Train on Tweebank$_{train}$					
	ESL			MT			ESL			MT		
	Repl.	Miss.	Unnec.	Repl.	Miss.	Unnec.	Repl.	Miss.	Unnec.	Repl.	Miss.	Unnec.
min	93.7 (MST)			92.8 (Yara)			89.4 (SyntaxNet)			87.8 (SNN)		
Malt												
Mate												
MST												
SNN												
SyntaxNet												
Turbo												
Tweebo												
Yara												
max	96.9 (Turbo)			97.2 (SNN)			97.8 (Malt)			97.6 (Malt)		

Table 3: Parser robustness on sentences with one grammatical error, each can be categorized as a replacement error, a missing word error or an unnecessary word error.

goal is to see whether different error types have different effect on parsers. If yes, is there a parser that is more robust than others?

As shown in Table 3, replacement errors are the least problematic error type for all the parsers; on the other hand, missing errors are the most difficult error type for parsers. This finding suggests that a preprocessing module for correcting missing and unnecessary word errors may be helpful in the parsing pipeline.

5.4.2 Impact of error word category

Another factor that might affect parser performances is the class of errors; for example, we might expect an error on a preposition to have a higher impact (since it is structural) than an error on an adjective. We separate the sentences into two groups: error occurring on an open- or closed-class word. We expect closed-class errors to have a stronger negative impact on the parsers because they contain function words such as determiners, pronouns, conjunctions and prepositions.

Table 4 shows results. As expected, closed-class errors are generally more difficult for parsers. But when parsers are trained on PTB and tested on MT, there are some exceptions: Turbo, Mate, MST and Yara parsers tend to be more robust on closed-class errors. This result corroborates the importance of building grammar error correction systems to handle closed-class errors such as preposition errors.

5.4.3 Impact of error semantic role

An error can be either in a verb role, an argument role, or no semantic role. We extract semantic role of the error by running Illinoise semantic role labeler (Punyakanok et al., 2008) on corrected version of the

Parser	Train on PTB §1-21				Train on Tweebank$_{train}$			
	ESL		MT		ESL		MT	
	Open class	Closed class	Open class	Closed class	Open class	Closed class	Open class	Closed class
min	95.1 (SNN)		94.5 (Yara)		89.6 (SyntaxNet)		91.5 (SNN)	
Malt								
Mate								
MST								
SNN								
SyntaxNet								
Turbo								
Tweebo								
Yara								
max	96.8 (Malt)		96.1 (SNN)		97.6 (Malt)		97.0 (Malt)	

Table 4: Parser robustness on sentences with one error, where the error either occurs on an open-class (lexical) word or a closed-class (functional) word.

Parser	Train on PTB §1-21						Train on Tweebank$_{train}$					
	ESL			MT			ESL			MT		
	Verb	Argument	No role	Verb	Argument	No role	Verb	Argument	No role	Verb	Argument	No role
min	94.1 (SyntaxNet)			91.8 (Malt)			91.8 (SNN)			92.2 (SNN)		
Malt												
Mate												
MST												
SNN												
SyntaxNet												
Turbo												
Tweebo												
Yara												
max	96.7 (Turbo)			96.7 (SyntaxNet)			96.9 (Malt)			96.9 (Malt)		

Table 5: Parser robustness on sentences with one error where the error occurs on a word taking on a verb role, an argument role, or a word with no semantic role.

sentences. We then obtain the role of the errors using alignments between ungrammatical sentence and its corrected counterpart.

Table 5 shows the average robustness of parsers when parsing sentences that have one error. For parsers trained on the PTB data, handling sentences with argument errors seem somewhat easier than those with other errors. For parsers trained on the Tweebank, the variation in the semantic roles of the errors does not seem to impact parser performance; each parser performs equally well or poorly across all roles; comparing across parsers, Malt seems particularly robust to error variations due to semantic roles.

6 Conclusions and Recommendations

In this paper, we have presented a set of empirical analyses on the robustness of processing ungrammatical text for several leading dependency parsers, using an evaluation metric designed for this purpose. We find that parsers indeed have different responses to ungrammatical sentences of various types. We recommend practitioners to examine the range of ungrammaticality in their input data (whether it is more like tweets or has grammatical errors like ESL writings). If the input data contains text more similar to tweets (e.g. containing URLs and emoticons), Malt or Turbo parser may be good choices. If the input data is more similar to the machine translation outputs; SyntaxNet, Malt, Tweebo and Turbo parser are good choices.

Our results also suggest that some preprocessing steps may be necessary for ungrammatical sentences, such as handling redundant and missing word errors. While there are some previous works on fixing the unnecessary words in the literature (Xue and Hwa, 2014), it is worthy to develop better NLP methods for catching and mitigating the missing word errors prior to parsing. Finally, this work corroborate the importance of building grammar error correction systems for handling closed-class er-

rors such as preposition errors.

Acknowledgments

This work was supported in part by the National Science Foundation Award #1550635. We would like to thank the anonymous reviewers and the Pitt NLP group for their helpful comments.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Johnny Bigert, Jonas Sjöbergh, Ola Knutsson, and Magnus Sahlgren. 2005. Unsupervised evaluation of parser robustness. In *Computational Linguistics and Intelligent Text Processing*, pages 142–154.

E. Black, S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 306–311.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97.

Aoife Cahill. 2015. Parsing learner text: to shoehorn or not to shoehorn. In *Proceedings of LAW IX - The 9th Linguistic Annotation Workshop*, page 144.

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

Jinho D Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 26–31.

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

Joachim Daiber and Rob van der Goot. 2016. The denoised web treebank: Evaluating dependency parsing under noisy input conditions. In *LREC*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*, number 2006, pages 449–454.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef Van Genabith, et al. 2011. # hardtoparse: POS tagging and parsing the twitterverse. In *proceedings of the Workshop On Analyzing Microtext (AAAI 2011)*, pages 20–25.

Jennifer Foster. 2004. Parsing ungrammatical input: an evaluation procedure. In *LREC*.

Jennifer Foster. 2007. Treebanks gone bad. *International Journal of Document Analysis and Recognition*, 10(3-4):129–145.

Jennifer Foster. 2010. "cba to check the spelling" investigating parser performance on discussion forum posts. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale l2 databases: the EF-Cambridge open language database (EFCamDat). In *Proceedings of the 31st Second Language Research Forum*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *ACL-HLT*, pages 42–47.

Tuomo Kakkonen. 2007. Robustness evaluation of two CCG, a PCFG and a link grammar parsers. *Proceedings of the 3rd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*.

Lingpeng Kong and Noah A Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. *arXiv preprint arXiv:1404.4314*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1048–1059.

André FT Martins, Miguel Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 617–622.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Niels Ott and Ramon Ziai. 2010. Evaluating dependency parsing performance on german learner language. *Proceedings of the Ninth Workshop on Treebanks and Linguistic Theories (TLT-9)*, 9:175–186.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 705–713.

Marion Potet, Emmanuelle Esperança-Rodier, Laurent Besacier, and Hervé Blanchon. 2012. Collection of a large database of French-English SMT output corrections. In *LREC*, pages 4043–4048.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Marwa Ragheb and Markus Dickinson. 2012. Defining syntax for learner language annotation. In *COLING (Posters)*, pages 965–974.

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy G Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, et al. 2006. Sparseval: Evaluation metrics for parsing speech. In *LREC*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, pages 173–180.

Huichao Xue and Rebecca Hwa. 2014. Redundancy detection in esl writings. In *EACL*, pages 683–691.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 180–189.

Neural Shift-Reduce CCG Semantic Parsing

Dipendra K. Misra and **Yoav Artzi**
Department of Computer Science and Cornell Tech
Cornell University
New York, NY 10011
{dkm,yoav}@cs.cornell.edu

Abstract

We present a shift-reduce CCG semantic parser. Our parser uses a neural network architecture that balances model capacity and computational cost. We train by transferring a model from a computationally expensive log-linear CKY parser. Our learner addresses two challenges: selecting the best parse for learning when the CKY parser generates multiple correct trees, and learning from partial derivations when the CKY parser fails to parse. We evaluate on AMR parsing. Our parser performs comparably to the CKY parser, while doing significantly fewer operations. We also present results for greedy semantic parsing with a relatively small drop in performance.

1 Introduction

Shift-reduce parsing is a class of parsing methods that guarantees a linear number of operations in sentence length. This is a desired property for practical applications that require processing large amounts of text or real-time response. Recently, such techniques were used to build state-of-the-art syntactic parsers, and have demonstrated the effectiveness of deep neural architectures for decision making in linear-time dependency parsing (Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016). In contrast, semantic parsing often relies on algorithms with polynomial number of operations, which results in slow parsing times unsuitable for practical applications. In this paper, we apply shift-reduce parsing to semantic parsing. Specifically, we study transferring a learned Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000) from a dynamic-programming CKY model to a shift-reduce neural network architecture.

We focus on the feed-forward architecture of Chen and Manning (2014), where each parsing step is a multi-class classification problem. The state of the parser is represented using simple feature embeddings that are passed through a multilayer perceptron to select the next action. While simple, the capacity of this model to capture interactions between primitive features, instead of relying on sparse complex features, has led to new state-of-the-art performance (Andor et al., 2016). However, applying this architecture to semantic parsing presents learning and inference challenges.

In contrast to dependency parsing, semantic parsing corpora include sentences labeled with the system response or the target formal representation, and omit derivation information. CCG induction from such data relies on latent-variable techniques and requires careful initialization (e.g., Zettlemoyer and Collins, 2005, 2007). Such feature initialization does not directly transfer to a neural network architecture with dense embeddings, and the use of hidden layers further complicates learning by adding a large number of latent variables. We focus on data that includes sentence-representation pairs, and learn from a previously induced log-linear CKY parser. This drastically simplifies learning, and can be viewed as bootstrapping a fast parser from a slow one. While this dramatically narrows down the number of parses per sentence, it does not eliminate ambiguity. In our experiments, we often get multiple correct parses, up to 49K in some cases. We also observe that the CKY parser generates no parses for

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1775–1786,
Austin, Texas, November 1-5, 2016.

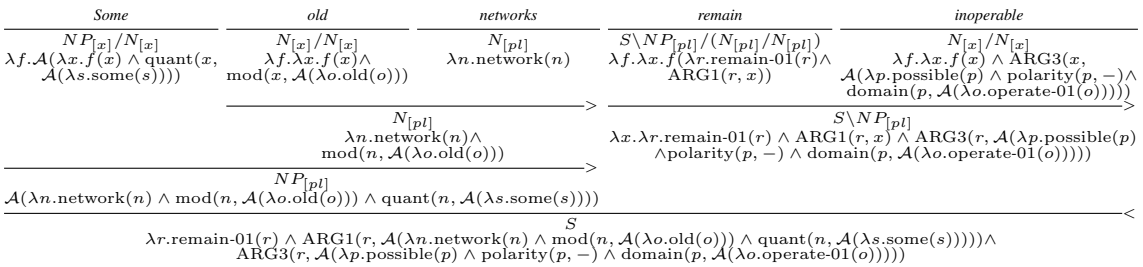


Figure 1: Example CCG tree with five lexical entries, three forward applications (>) and a backward application (<).

a significant number of training sentences. Therefore, we propose an iterative algorithm that automatically selects the best parses for training at each iteration, and identifies partial derivations for best-effort learning, if no parses are available.

CCG parsing largely relies on two types of actions: using a lexicon to map words to their categories, and combining categories to acquire the categories of larger phrases. In most semantic parsing approaches, the number of operations is dominated by the large number of categories available for each word in the lexicon. For example, the lexicon in our experiments includes 1.7M entries, resulting in an average of 146, and up to 2K, applicable actions. Additionally, both operations and parser state have complex structures, for example including both syntactic and semantic information. Therefore, unlike in dependency parsing (Chen and Manning, 2014), we can not treat action selection as multi-class classification, and must design an architecture that can accommodate a varying number of actions. We present a network architecture that considers a variable number of actions, and emphasizes low computational overhead per action, instead focusing computation on representing the parser state.

We evaluate on Abstract Meaning Representation (AMR; Banarescu et al., 2013) parsing. We demonstrate that our modeling and learning contributions are crucial to effectively commit to early decisions during parsing. Somewhat surprisingly, our shift-reduce parser provides equivalent performance to the CKY parser used to generate the training data, despite requiring significantly fewer operations, on average two orders of magnitude less. Similar to previous work, we use beam search, but also, for the first time, report greedy CCG semantic parsing results at a relatively modest 9% decrease in performance, while the source CKY parser with a beam of one demonstrates a 71% decrease. While we focus on semantic parsing, our learning approach makes no task-specific assumptions and has potential for learning efficient models for structured prediction from the output of more expensive ones.[1]

2 Task and Background

Our goal is to learn a function that, given a sentence x, maps it to a formal representation of its meaning z with a linear number of operations in the length of x. We assume access to a training set of N examples $\mathcal{D} = \{(x^{(i)}, z^{(i)})\}_{i=1}^{N}$, each containing a sentence $x^{(i)}$ and a logical form $z^{(i)}$. Since $\mathcal{D}$ does not contain complete derivations, we instead assume access to a CKY parser learned from the same data. We evaluate performance on a test set $\{(x^{(i)}, z^{(i)})\}_{i=1}^{M}$ of M sentences $x^{(i)}$ labeled with logical forms $z^{(i)}$. While we describe our approach in general terms, we apply our approach to AMR parsing and evaluate on a common benchmark (Section 6).

To map sentences to logical forms, we use CCG, a linguistically-motivated grammar formalism for modeling a wide-range of syntactic and semantic phenomena (Steedman, 1996, 2000). A CCG is defined by a lexicon Λ and sets of unary $\mathcal{R}_u$ and binary $\mathcal{R}_b$ rules. In CCG parse trees, each node is a category. Figure 1 shows a CCG tree for the sentence *Some old networks remain inoperable*. For example, $S\backslash NP_{[pl]}/(N_{[pl]}/N_{[pl]})$: $\lambda f.\lambda x.f(\lambda r.\text{remain-01}(r) \wedge \text{ARG1}(r, x))$ is the category of the verb *remain*. The syntactic type $S\backslash NP_{[pl]}/(N_{[pl]}/N_{[pl]})$ indicates that two arguments are expected: first an adjective $N_{[pl]}/N_{[pl]}$ and then a plural noun phrase $NP_{[pl]}$. The final syntactic type will be S. The forward slash / indicates the argument is expected on the right, and the backward slash \ indicates it is expected on the left. The syntactic attribute pl is used to express the plural-

[1]The source code and pre-trained models are available at `http://www.cs.cornell.edu/~dkm/ccgparser`.

ity constraint of the verb. The simply-typed lambda calculus logical form in the category represents semantic meaning. The typing system includes atomic types (e.g., entity e, truth value t) and functional types (e.g., $\langle e, t\rangle$ is the type of a function from e to t). In the example category above, the expression on the right of the colon is a $\langle\langle\langle e,t\rangle,\langle e,t\rangle\rangle,\langle e,\langle e,t\rangle\rangle\rangle$-typed function expecting first an adjectival modifier and then an ARG1 modifier. The conjunction $\wedge$ specifies the roles of remain-01. The lexicon Λ maps words to CCG categories. For example, the lexical entry $remain \vdash S\backslash NP_{[pl]}/(N_{[pl]}/N_{[pl]}) : \lambda f.\lambda x.f(\lambda r.\text{remain-01}(r) \wedge \text{ARG1}(r,x))$ pairs the example category with *remain*. The parse tree in the figure includes four binary operations: three forward applications ($>$) and a backward application ($<$).

3 Neural Shift Reduce Semantic Parsing

Given a sentence $x = \langle x_1, \ldots, x_m\rangle$ with m tokens x_i and a CCG lexicon Λ, let $\textsc{Gen}(x;\Lambda)$ be a function that generates CCG parse trees. We design GEN as a shift-reduce parser, and score decisions using embeddings of parser states and candidate actions.

3.1 Shift-Reduce Parsing for CCG

Shift-reduce parsers perform a single pass of the sentence from left to right to construct a parse tree. The parser configuration[2] is defined with a stack and a buffer. The stack contains partial parse trees, and the buffer the remainder of the sentence to be processed. Formally, a parser configuration c is a tuple $\langle\sigma,\beta\rangle$, where the stack σ is a list of CCG trees $[s_l \cdots s_1]$, and the buffer β is a list of tokens from x to be processed $[x_i \cdots x_m]$.[3] For example, the top-left of Figure 2 shows a parsing configuration with two partial trees on the stack and two words on the buffer (*remain* and *inoperable*).

Parsing starts with the configuration $\langle[], [x_1 \cdots x_m]\rangle$, where the stack is empty and the buffer is initialized with x. In each parsing step, the parser either consumes a word from the buffer and pushes a new tree to the stack, or applies a parsing rule to the trees at the top of the stack. For simplicity, we apply CCG rules to trees, where a rule is applied to the root categories of the argument trees to create a new tree with the arguments as children. We treat lexical entries as trees with a single node. There are three types of actions:[4]

$$\begin{aligned}
\textsc{Shift}(l, \langle\sigma, x_i|\cdots|x_j|\beta\rangle) &= \langle\sigma|g,\beta\rangle \\
\textsc{Binary}(b, \langle\sigma|s_2|s_1,\beta\rangle) &= \langle\sigma|b(s_2,s_1),\beta\rangle \\
\textsc{Unary}(u, \langle\sigma|s_1,\beta\rangle) &= \langle\sigma|u(s_1),\beta\rangle \quad .
\end{aligned}$$

Where $b \in \mathcal{R}_b$ is a binary rule, $u \in \mathcal{R}_u$ is a unary rule, and l is a lexical entry $x_i, \ldots x_j \vdash g$ for the tokens x_i,...,x_j and CCG category g. SHIFT creates a tree given a lexical entry for the words at the top of the buffer, BINARY applies a binary rule to the two trees at the head of the stack, and UNARY applies a unary rule to the tree at head of the stack. A configuration is terminal when no action is applicable.

Given a sentence x, a derivation is a sequence of action-configuration pairs $\langle\langle c_1, a_1\rangle, \ldots, \langle c_k, a_k\rangle\rangle$, where action a_i is applied to configuration c_i to generate configuration c_{i+1}. The result configuration c_{k+1} is of the form $\langle[s],[]\rangle$, where s represents a complete parse tree, and the logical form z at the root category represents the meaning of the complete sentence. Following previous work with CKY parsing (Zettlemoyer and Collins, 2005), we disallow consecutive unary actions. We denote the set of actions allowed from configuration c as $\mathcal{A}(c)$.

3.2 Model

Our goal is to balance computation and model capacity. To recover a rich representation of the configuration, we use a multilayer perceptron (MLP) to create expressive interactions between a small number of simple features. However, since we consider many possible actions in each step, computing activations for multiple hidden layers for each action is prohibitively expensive. Instead, we opt for a computationally-inexpensive action representation computed by concatenating feature embeddings. Figure 2 illustrates our architecture.

Given a configuration c, the probability of an action a is:

$$p(a \mid c) = \frac{\exp\{\phi(a,c)\mathbf{W}_b\mathcal{F}(\xi(c))\}}{\sum_{a'\in\mathcal{A}(c)}\exp\{\phi(a',c)\mathbf{W}_b\mathcal{F}(\xi(c))\}} \quad ,$$

[2] We use the terms *parser configuration* and *parser state* interchangeably.

[3] The head of the stack σ is the right-most entry, and the head of the buffer β is the left-most entry.

[4] We follow the standard notation of $L|x$ indicating a list with all the entries from L and x as the right-most element.

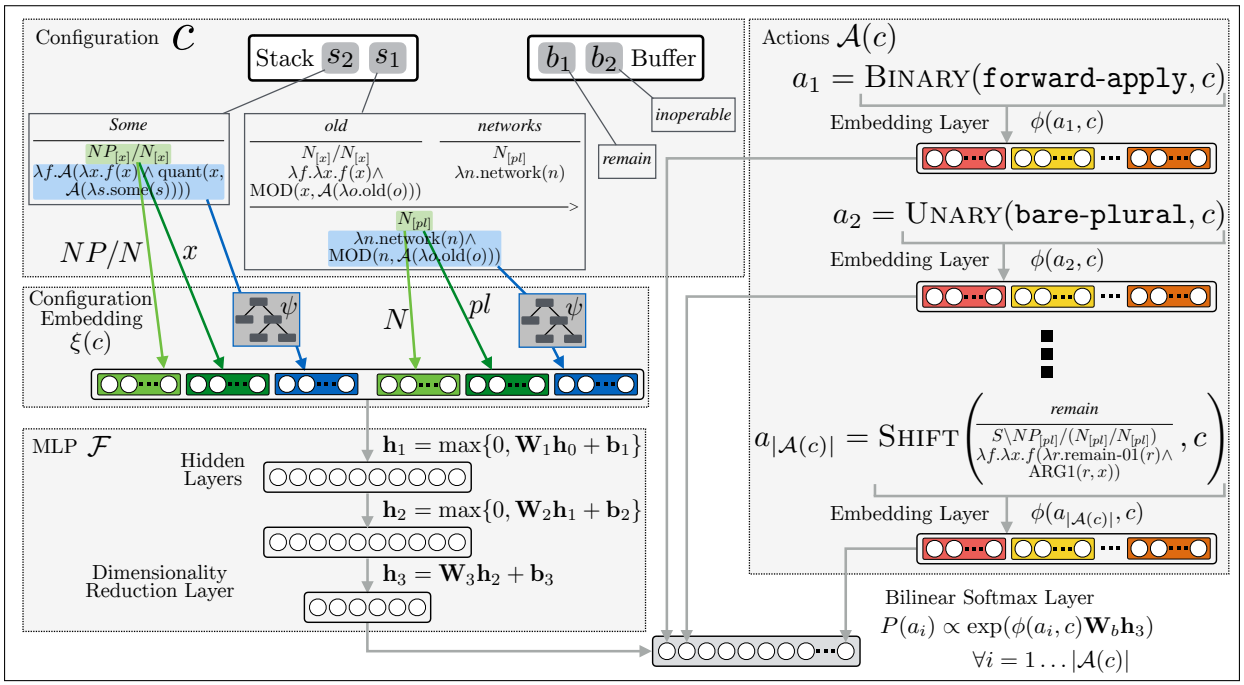


Figure 2: Illustration of scoring the next action given the configuration c when parsing the sentence *Some old networks remain inoperable*. Embeddings of the same feature type are colored the same. The configuration embedding $\xi(c)$ is a concatenation of syntax embeddings (green) and the logical form embedding (blue; computed by ψ) for the top entries in the stack. We then pass $\xi(c)$ through the MLP $\mathcal{F}$. Given the actions $\mathcal{A}(c)$, we compute the embeddings $\phi(a_i, c)$, $i = 1 \ldots |\mathcal{A}(c)|$. The actions and MLP representation are combined with a bilinear softmax layer. The number of concatenated vectors and stack elements used is for illustration. The details are described in Section 3.2.

where $\phi(a, c)$ is the action embedding, $\xi(c)$ is the configuration embedding, and $\mathcal{F}$ is an MLP. $\mathbf{W}_b$ is a bilinear transformation matrix. Given a sentence x and a sequence of action-configuration pairs $\langle\langle c_1, a_1\rangle, \ldots, \langle c_k, a_k\rangle\rangle$, the probability of a CCG tree y is

$$p(y \mid x) = \prod_{i=1\ldots k} p(a_i \mid c_i) \ .$$

The probability of a logical form z is then

$$p(z \mid x) = \sum_{y \in \mathcal{Y}(z)} p(y \mid x) \ ,$$

where $\mathcal{Y}(z)$ is the set of CCG trees with the logical form z at the root.

MLP Architecture $\mathcal{F}$ We use a MLP with two hidden layers parameterized by $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ with a ReLu non-linearity (Glorot et al., 2011). Since the output of $\mathcal{F}$ influences the dimensionality of $\mathbf{W}_b$, we add a linear layer parameterized by $\mathbf{W}_3$ and $\mathbf{b}_3$ to reduce the dimensionality of the configuration, thereby reducing the dimensionality of $\mathbf{W}_b$.

Configuration Embedding $\xi(c)$ Given a configuration $c = \langle[s_l \cdots s_1], [x_i \cdots x_m]\rangle$, the input to $\mathcal{F}$ is a concatenation of syntactic and semantic embeddings, as illustrated in Figure 2. We concatenate embeddings from the top three trees in the stack s_1, s_2, s_3.[5] When a feature is not present, for example when the stack or buffer are too small, we use a tunable null embedding.

Given a tree on the stack s_j, we define two syntactic features: *attribute set* and *stripped syntax*. The attribute feature is created by extracting all the syntactic attributes of the root category of s_j. The stripped syntax feature is the syntax of the root category without the syntactic attributes. For example, in Figure 2, we embed the stripped category N and attribute pl for s_1, and NP/N and x for s_2. The attributes are separated from the syntax to reduce sparsity, and the interaction between them is computed by $\mathcal{F}$. The sparse features are converted to dense embeddings using a lookup table and concatenated. In addition, we also embed the logical form at the root of s_j. Figure 3 illustrates the recursive embedding function ψ.[6] Using a recursive function to embed logical forms is computationally intensive. Due to strong correlation between sentence length and logical form complexity, this computation increases

[5] For simplicity, the figure shows only the top two trees.

[6] The algorithm is provided in the supplementary material.

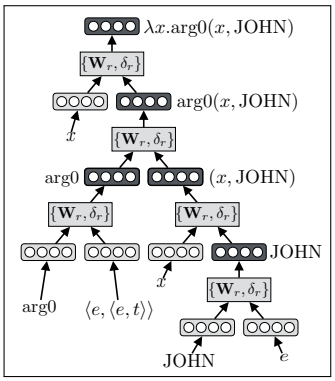


Figure 3: Illustration of embedding the logical form $\lambda x.\text{arg0}(x, \text{JOHN})$ with the recursive embedding function ψ. In each level in ψ, the children nodes are combined with a single-layer neural network parameterized by $\mathbf{W}_r$, δ_r, and the $\tanh$ activation function. Computed embeddings are in dark gray, and embeddings from lookup tables are in light gray. Constants are embedded by combining name and type embeddings, literals are unrolled to binary recursive structures, and lambda terms are combinations of variable type and body embeddings. For example, JOHN is embedded by combining the embeddings of its name and type, the literal $\text{arg0}(x, \text{JOHN})$ is recursively embedded by first embedding the arguments (x, JOHN) and then combining the predicate, and the lambda term is embedded to create the embedding of the entire logical form.

the cost of configuration embedding by a factor linear in sentence length. In Section 6, we experiment with including this option, balancing between potential expressivity and speed.

Action Embedding $\phi(a, c)$ Given an action $a \in \mathcal{A}(c)$, and the configuration c, we generate the action representation by computing sparse features, converting them to dense embeddings via table lookup, and concatenating. If more than one feature of the same type is triggered, we average their embeddings. When no features of a given type are triggered, we use a tunable placeholder embedding instead. The features include all the features used by Artzi et al. (2015), including all conjunctive features, as well as properties of the action and configuration, such as the POS tags of tokens on the buffer.[7]

Discussion Our use of an MLP is inspired by Chen and Manning (2014). However, their architecture is designed to handle only a fixed number of actions, while we observe varying number of actions. Therefore, we adopt a probabilistic model similar to Dyer et al. (2015) to effectively combine the benefits of the two approaches.[8] We factorize the exponent in our objective into action $\phi(a, c)$ and configuration $\mathcal{F}(\xi(c))$ embeddings. While every parse step involves a single configuration, the number of actions is significantly higher. With the goal of minimizing the amount of computation per action, we use simple concatenation only for action embedding. However, this requires retaining sparse conjunctive action features since they are never combined through hidden layers similar to configuration features.

3.3 Inference

To compute the set of parse trees $\textsc{Gen}(x; \Lambda)$, we perform beam search to recover the top-k parses. The beam contains configurations. At each step, we expand all configurations with all actions, and keep only the top-k new configurations. To promote diversity in the beam, given two configurations with the same *signature*, we keep only the highest scoring one. The signature includes the previous configuration in the derivation, the state of the buffer, and the root categories of all stack elements. Since all features are computed from these elements, this optimization does not affect the max-scoring tree. Additionally, since words are assigned structured categories, a key problem is unknown words or word uses. Following Zettlemoyer and Collins (2007), we use a two-pass parsing strategy, and allow skipping words controlled by the term γ in the second pass. The term γ is added to the exponent of the action probability when words are skipped. See the supplementary material for the exact form.

Complexity Analysis The shift-reduce parser processes the sentence from left to right with a linear number of operations in sentence length. We define an operation as applying an action to a configuration. Formally, the number of operations for a sentence of length m is bounded by $O(4mk(|\lambda| + |\mathcal{R}_b| + |\mathcal{R}_u|))$, where $|\lambda|$ is the number of lexical entries per token, k is the beam size, $\mathcal{R}_b$ is the set of binary rules, and $\mathcal{R}_u$ the set of unary rules. In comparison, the number of operations for the CKY parser, where an operation is applying a rule to a single cell or two adjacent cells in the chart, is bounded by $O(m|\lambda| + m^3k^2|\mathcal{R}_b| + m^2b|\mathcal{R}_u|)$. For sentence

[7] See the supplementary material for feature details.

[8] We experimented with an LSTM parser similar to Dyer et al. (2015). However, performance was not competitive. This direction remains an important avenue for future work.

length 25, the mean in our experiments, the shift-reduce parser performs 100 time fewer operations. See the supplementary material for the full analysis.

4 Learning

We assume access to a training set of N examples $\mathcal{D} = \{(x^{(i)}, z^{(i)})\}_{i=1}^{N}$, each containing a sentence $x^{(i)}$ and a logical form $z^{(i)}$. The data does not include information about the lexical entries and CCG parsing operations required to construct the correct derivations. We bootstrap this information from a learned parser. In our experiments we use a learned dynamic-programming CKY parser. We transfer the lexicon Λ directly from the input parser, and focus on estimating the parameters θ, which include feature embeddings, hidden layer matrices, and bias terms. The main challenge is learning from the noisy supervision provided by the input parser. In our experiments, the CKY parser fails to correctly parse 40% of the training data, and returns on average 147 max-scoring correct derivations for the rest. We propose an iterative algorithm that treats the choice between multiple parse trees as latent, and effectively learns from partial analysis when no correct derivation is available.

The learning algorithm (Algorithm 1) starts by processing the data using the CKY parser (lines 3 - 4). For each sentence $x^{(i)}$, we collect the max-scoring CCG trees with $z^{(i)}$ at the root. The CKY parser often contains many correct parses with identical scores, up to 49K parses per sentence. Therefore, we randomly sample and keep up to 1K trees. This process is done once, and the algorithm then runs for T iterations. At each iteration, given the sets of parses from the CKY parser $\mathcal{Y}$, we select the max-probability parse according to our current parameters θ (line 10) and add all the shift-reduce decisions from this parse to $\mathcal{D}_A$ (line 12), the action data set that we use to estimate the parameters. We approximate the $\arg\max$ with beam search using an oracle computed from the CKY parses.[9] CONFGEN aggregates the configuration-action pairs from the highest scoring derivation. Parse selection depends on θ and this choice will gradually converge as the parameters improve. The action data set is used to compute the ℓ_2-regularized negative log-likelihood objective

[9] Our oracle is non-deterministic and incomplete (Goldberg and Nivre, 2013).

Algorithm 1 The learning algorithm.

Input: Training set $\mathcal{D} = \{(x^{(i)}, z^{(i)})\}_{i=1}^{N}$, learning rate μ, regularization parameter ℓ_2, and number of iterations T.

Definitions: GENMAXCKY(x, z) returns the set of max-scoring CKY parses for x with z at the root. SCORE(y, θ) scores a tree y according to the parameters θ (Section 3.2). CONFGEN(x, y) is the sequence of action-configuration pairs that generates y given x (Section 3.1). BP$(\Delta_{\mathcal{J}})$ takes the objective $\mathcal{J}$ and back-propagates the error $\nabla\mathcal{J}$ through the computation graph for the sample used to compute the objective. ADAGRAD(Δ) applies a per-feature learning rate to the gradient Δ (Duchi et al., 2011).

Output: Model parameters θ.

```
1:  » Get trees from CKY parser.
2:  Y ← []
3:  for i = 1 to N do
4:      Y[i] = GENMAXCKY(x^(i), z^(i))
5:  for t = 1 to T do
6:      » Pick max-scoring trees and create action dataset.
7:      D_A = ∅
8:      for i = 1 to N do
9:          if Y[i] ≠ ∅ then
10:             A ← CONFGEN(x^(i),
11:                     arg max_{y∈Y[i]} SCORE(y, θ))
12:             for ⟨c, a⟩ ∈ A do
13:                 D_A ← D_A ∪ {⟨c, a⟩}
14:     » Back-propagate the loss through the network.
15:     for ⟨c, a⟩ ∈ D_A do
16:         J =def − log p(a | c) + (ℓ2/2) θ^T θ
17:         Δ ← BP(∇J)
18:         θ ← θ − μ ADAGRAD(Δ)
19: return θ
```

$\mathcal{J}$ (line 16) and back-propagate the error to compute the gradient (line 17). We use AdaGrad (Duchi et al., 2011) to update the parameters θ (line 18).

4.1 Learning from Partial Derivations

The input parser often fails to generate correct parses. In our experiments, this occurs for 40% of the training data. In such cases, we can obtain a forest of partial parse trees $\mathcal{Y}_p$. Each partial tree $y \in \mathcal{Y}_p$ corresponds to a span of tokens in the sentence and is scored by the input parser. In practice, the spans are often overlapping. Our goal is to generate high quality configuration-action pairs $\langle c, a \rangle$ from $\mathcal{Y}_p$. These pairs will be added to $\mathcal{D}_A$ for training. While extracting actions a is straightforward, generating configurations c requires reconstructing the stack σ from an incomplete forest of partial trees $\mathcal{Y}_p$. Figure 4 illustrates our proposed process. Let CKYSCORE(y) be the CKY score of the partial tree y. To reconstruct σ, we select non-overlapping par-

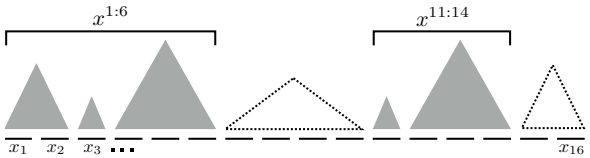


Figure 4: Partial derivation selection for learning (Section 4.1). The dotted triangles represent skipped spans in the sentence, where no high quality partial trees were found. Dark triangles represent the selected partial trees. We identify two contiguous spans, 1-6 and 11-14, and generate two synthetic sentences for training: the tokens are treated as complete sentences and actions and stack state are generated from the partial trees.

tial trees Y that correspond to the entire sentence by solving $\arg\max_{Y \subseteq \mathcal{Y}_p} \text{CKYSCORE}(y)$ under two constraints: (a) no two trees from Y correspond to overlapping tokens, and (b) for each token in x, there exists $y \in Y$ that corresponds to it. We solve the $\arg\max$ using dynamic programming. The generated set Y approximates an intermediate state of a shift-reduce derivation. However, $\mathcal{Y}_p$ often does not contain high quality partial derivation for all spans. To skip low quality partial trees and spans that have no trees, we generate empty trees y_e for every span, where $\text{CKYSCORE}(y_e) = 0$, and add them to $\mathcal{Y}_p$. If the set of selected partial trees Y includes empty trees, we divide the sentence to separate examples and ignore these parts. This results in partial and approximate stack reconstruction. Finally, since $\mathcal{Y}_P$ is noisy, we prune from it partial trees with a root that does not match the syntactic type for this span from an automatically generated CCGBank (Hockenmaier and Steedman, 2007) syntactic parse.

Our complete learning algorithm alternates between epochs of learning with complete parse trees and learning with partial derivations. In epochs where we use partial derivations, we use a modified version of Algorithm 1, where lines 9-10 are updated to use the above process.

5 Related work

Our approach is inspired by recent results in dependency parsing, specifically by the architecture of Chen and Manning (2014), which was further developed by Weiss et al. (2015) and Andor et al. (2016). Dyer et al. (2015) proposed to encode the parser state using an LSTM recurrent architecture, which has been shown generalize well between languages (Ballesteros et al., 2015; Ammar et al., 2016). Our network architecture combines ideas from the two threads: we use feature embeddings and a simple MLP to score actions, while our probability distribution is similar to the LSTM parser.

The majority of CCG approaches for semantic parsing rely on CKY parsing with beam search (e.g., Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010, 2011; Artzi and Zettlemoyer, 2011, 2013; Artzi et al., 2014; Matuszek et al., 2012; Kushman and Barzilay, 2013). Semantic parsing with other formalisms also often relied on CKY-style algorithms (e.g., Liang et al., 2009; Kim and Mooney, 2012). With a similar goal to ours, Berant and Liang (2015) designed an agenda-based parser. In contrast, we focus on a method with linear number of operations guarantee.

Following the work of Collins and Roark (2004) on learning for syntactic parsers, Artzi et al. (2015) proposed an early update procedure for inducing CCG grammars with a CKY parser. Our partial derivations learning method generalizes this method to parsers with global features.

6 Experimental Setup

Task and Data We evaluate on AMR parsing with CCG. AMR is a general-purpose meaning representation, which has been used in multiple tasks (Pan et al., 2015; Liu et al., 2015; Sawai et al., 2015; Garg et al., 2016), We use the newswire portion of AMR Bank 1.0 release (LDC2014T12), which displays some of the fundamental challenges in semantic parsing, including long newswire sentences with a broad array of syntactic and semantic phenomena. We follow the standard train/dev/test split of 6603/826/823 sentences. We evaluate with the SMATCH metric (Cai and Knight, 2013). Our parser is incorporated into the two-stage approach of Artzi et al. (2015). The approach includes a bi-directional and deterministic conversion between AMR and lambda calculus. Distant references, for example such as introduced by pronouns, are represented using Skolem IDs, globally-scoped existentially-quantified unique IDs. A derivation includes a CCG tree, which maps the sentence to an *underspecified logical form*, and a constant mapping, which maps underspecified elements to their fully specified form. The key to the approach is the underspecified logical forms, where distant references and most relations are not fully specified, but instead represented

AMR	Underspecified Logical Form	Logical Form
$(c/\text{conclude-02}$	$\mathcal{A}_1(\lambda c.\text{conclude-02}(c) \wedge$	$\mathcal{A}_1(\lambda c.\text{conclude-02}(c) \wedge$
:ARG0 (l/lawyer)	$\text{ARG0}(c, \mathcal{A}_2(\lambda l.\text{lawyer}(l))) \wedge$	$\text{ARG0}(c, \mathcal{A}_2(\lambda l.\text{lawyer}(l))) \wedge$
:ARG1 $(a/\text{argument}$	$\text{ARG1}(c, \mathcal{A}_3(\lambda a.\text{argument}(a) \wedge$	$\text{ARG1}(c, \mathcal{A}_3(\lambda a.\text{argument}(a) \wedge$
:poss $l)$	$\text{poss}(a, \mathcal{R}(\textbf{ID}))))) \wedge$	$\text{poss}(a, \mathcal{R}(2)))) \wedge$
:time $(l2/late))$	$\textbf{REL}(c, \mathcal{A}_4(\lambda l2.late(l2))))$	$\text{time}(c, \mathcal{A}_4(\lambda l2.late(l2))))$

Figure 5: AMR for the sentence *the lawyer concluded his arguments late*. In Artzi et al. (2015), The AMR (left) is deterministically converted to the logical form (right). The underspecified logical form is the result of the first stage, CCG parsing, and contains two placeholders (bolded): ID for a reference, and REL for a relation. To generate the final logical form, the second stage resolves ID to the identifier of the *lawyer* (2), and REL to the relation time. We focus on a model for the first stage and use an existing model for the second stage.

as placeholders. Figure 5 shows an example AMR, its lambda calculus conversion, and its underspecified logical form. (Artzi et al., 2015) use a CKY parser to identify the best CCG tree, and a factor graph for the second stage. We integrate our shift-reduce parser into the two-stage setup by replacing the CKY parser. We use the same CCG configuration and integrate our parser into the join probabilistic model. Formally, given a sentence x, the probability of an AMR logical form z is

$$p(z \mid x) = \sum_u p(z \mid u, x) \sum_{y \in \mathcal{Y}(u)} p(y \mid x) \ ,$$

where u is an underspecified logical form, $\mathcal{Y}(u)$ is the set of CCG trees with u at the root. We use our shift-reduce parser to compute $p(y \mid x)$ and use the pre-trained model from Artzi et al. (2015) for $p(z \mid u, x)$. Following Artzi et al. (2015), we disallow configurations that will not result in a valid AMR, and design a heuristic post-processing technique to recover a single logical form from terminal configurations that include multiple disconnected partial trees on the stack. We use the recovery technique when no complete parses are available.

Tools We evaluate with the SMATCH metric (Cai and Knight, 2013). We use EasyCCG (Lewis and Steedman, 2014) for CCGBank categories (Section 4.1). We implement our system using Cornell SPF (Artzi, 2016), and the deeplearning4j library.[10] The setup of Artzi et al. (2015) also includes the Illinois NER (Ratinov and Roth, 2009) and Stanford CoreNLP POS Tagger (Manning et al., 2014).

Parameters and Initialization We minimize our loss on a held-out 10% of the training data to tune our parameters, and train the final model on the full data. We set the number of epochs $T = 3$, regularization coefficient $\ell_2 = 10^{-6}$, learning rate $\mu = 0.05$, skipping term $\gamma = 1.0$. We set the dimensionality of feature embeddings based on the vocabulary size of the feature type. The exact dimensions are listed in the supplementary material. We use 65 ReLU units for $\mathbf{h}_1$ and $\mathbf{h}_2$, and 50 units for $\mathbf{h}_3$. We initialize θ with the initialization scheme of Glorot and Bengio (2010), except the bias term for ReLu layers, which we initialize to 0.1 to increase the number of active units on initialization. During test, we use the vector $\mathbf{0}$ as embedding for unseen features. We use a beam of 512 for testing and 2 for CONFGEN (Section 4).

[10] http://deeplearning4j.org/

Parser	P	R	F
CKY (Artzi et al., 2015)	67.2	65.1	66.1
Greedy CKY	64.1	11.29	19.19
SR (complete model)	67.0	63.4	65.3
w/o semantic embedding	67.1	63.3	65.1
w/o partial derivation learning	66.0	62.2	64.0
Ensemble SR (syntax)	68.2	64.1	66.0
Ensemble SR (syntax, semantics)	68.1	63.9	65.9
SR with CKY model	52.5	49.36	50.88

Table 1: Development SMATCH results.

Parser	P	R	F
JAMR[11]	67.8	59.2	63.2
CKY (Artzi et al., 2015)	66.8	65.7	66.3
Shift Reduce	68.1	64.2	66.1
Wang et al. (2015a)[13]	72.0	67.0	70.0

Table 2: Test SMATCH results.[12]

Model Ensemble For our final results, we marginalize the output over three models $\mathcal{M}$ using $p(z \mid x, \theta, \Lambda) = \frac{1}{|M|} \sum_{m \in \mathcal{M}} p(z \mid m, x, \theta, \Lambda)$.

7 Results

Table 1 shows development results. We trained each model three times and report the best performance. We observed a variance of roughly 0.5 in these runs. We experimented with different features for configuration embedding and with removing learning with partial derivations (Section 4.1). The com-

plete model gives the best single-model performance of 65.3 F1 SMATCH, and we observe the benefits for semantic embeddings and learning from partial derivations. Using partial derivations allowed us to learn 370K more features, 22% of observed embeddings. We also evaluate ensemble performance. We observe an overall improvement in performance. However, with multiple models, the benefit of using semantic embeddings vanishes. This result is encouraging since semantic embeddings can be expensive to compute if the logical form grows with sentence length. We also provide results for running a shift-reduce log-linear parser $p(a \mid c) \propto \exp\{\mathbf{w}^T \phi_{\text{CKY}}(a, c)\}$ using the input CKY model. We observe a significant drop in performance, which demonstrates the overall benefit of our architecture.

Figure 6 shows the development performance of our best performing ensemble model for different beam sizes. The performance decays slowly with decreasing beam size. Surprisingly, our greedy parser achieves 59.77 SMATCH F1, while the CKY parser with a beam of 1 achieves only 19.2 SMATCH F1 (Table 1). This allows our parser to trade-off a modest drop in accuracy for a significant improvement in runtime.

Table 2 shows the test results using our best performing model (ensemble with syntax features). We compare our approach to the CKY parser of Artzi et al. (2015) and JAMR (Flanigan et al., 2014).[11,12] We also list the results of Wang et al. (2015b), who demonstrated the benefit of auxiliary analyzers and is the current state of the art.[13] Our performance is comparable to the CKY parser of (Artzi et al., 2015), which we use to bootstrap our system. This demonstrates the ability of our parser to match the performance of a dynamic-programming parser, which executes significantly more operations per sentence.

Finally, Figure 7 shows our parser runtime relative to sentence length. In this analysis, we focus on runtime, and therefore use a single model.

[11] JAMR results are taken from Artzi et al. (2015).

[12] Pust et al. (2015), Flanigan et al. (2014), and Wang et al. (2015b) report results on different sections of the corpus. These results are not comparable to ours.

[13] Our goal is to study the effectiveness of our model transfer approach and architecture. Therefore, we avoid using any resources used in (Wang et al., 2015b) that are not used in the CKY parser we compare to.

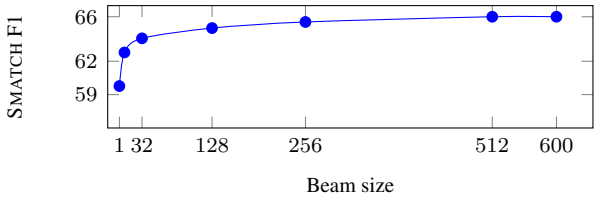


Figure 6: The effect of beam size on model performance.

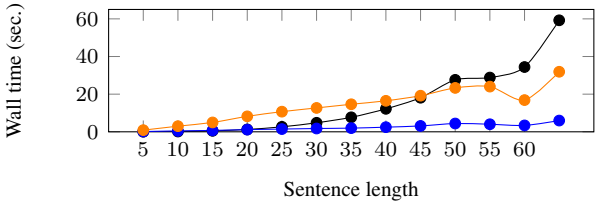


Figure 7: Wall-time performance of shift reduce parser with only syntax features (blue), with syntax and semantic features (orange) and the CKY parser of Artzi et al. (2015) (black).

We compare two versions of our system, including and excluding semantic embeddings, and the CKY parser of Artzi et al. (2015). We run both parsers with 16 cores and 122GB memory. The shift-reduce parser is three times faster on average, and up to ten times faster on long sentences. Since our parser is currently using CPUs, future work focused on GPU porting is likely to see further improvements.

8 Conclusion

Our parser design emphasizes a balance between model capacity and the ability to combine atomic features against the computational cost of scoring actions. We also design a learning algorithm to transfer learned models and learn neural network models from ambiguous and partial supervision. Our model shares many commonalities with transition-based dependency parsers. This makes it a good starting point to study the effectiveness of other dependency parsing techniques for semantic parsing, for example global normalization (Andor et al., 2016) and bidirectional LSTM feature representations (Kiperwasser and Goldberg, 2016).

Acknowledgments

This research was supported in part by gifts from Google and Amazon. The authors thank Kenton Lee for technical advice, and Adam Gibson and Alex Black of Skymind for help with Deeplearning4j. We also thank Tom Kwiatkowski, Arzoo Katiyar, Tianze Shi, Vlad Niculae, the Cornell NLP Group, and the reviewers for helpful advice.

References

Ammar, W., Mulcaire, G., Ballesteros, M., Dyer, C., and Smith, N. A. (2016). Many languages, one parser. *Transactions of the Association for Computational Linguistics*.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. *CoRR*.

Artzi, Y. (2016). Cornell SPF: Cornell semantic parsing framework. *ArXiv e-prints*.

Artzi, Y., Das, D., and Petrov, S. (2014). Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y., Lee, K., and Zettlemoyer, L. (2015). Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. S. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. S. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1.

Ballesteros, M., Dyer, C., and Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with LSTMs.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop*.

Berant, J. and Liang, P. (2015). Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the Conference of the Association of Computational Linguistics*.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the Conference of the Association of Computational Linguistics*.

Garg, S., Galstyan, A., Hermjakob, U., and Marcu, D. (2016). Extracting biomolecular interactions using semantic parsing of biomedical text. In *Proceedings of the Conference on Artificial Intelligence*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Goldberg, Y. and Nivre, J. (2013). Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1.

Hockenmaier, J. and Steedman, M. (2007). CCGBank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*.

Kim, J. and Mooney, R. J. (2012). Unsupervised PCFG induction for grounded language learning

with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4.

Kushman, N. and Barzilay, R. (2013). Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

Kwiatkowski, T., Zettlemoyer, L. S., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kwiatkowski, T., Zettlemoyer, L. S., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lewis, M. and Steedman, M. (2014). A* CCG parsing with a supertag-factored model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liang, P., Jordan, M., and Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics the International Joint Conference on Natural Language Processing*.

Liu, F., Flanigan, J., Thomson, S., Sadeh, N., and Smith, N. A. (2015). Toward abstractive summarization using semantic representations. In *Proceedings of the North American Association for Computational Linguistics*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Matuszek, C., FitzGerald, N., Zettlemoyer, L. S., Bo, L., and Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Pan, X., Cassidy, T., Hermjakob, U., Ji, H., and Knight, K. (2015). Unsupervised entity linking with Abstract Meaning Representation. In *Proceedings of the North American Association for Computational Linguistics*.

Pust, M., Hermjakob, U., Knight, K., Marcu, D., and May, J. (2015). Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning*.

Sawai, Y., Shindo, H., and Matsumoto, Y. (2015). Semantic structure analysis of noun phrases using abstract meaning representation. In *Proceedings of the annual meeting on Association for Computational Linguistics*.

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Wang, C., Xue, N., and Pradhan, S. (2015a). Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Wang, C., Xue, N., Pradhan, S., and Pradhan, S. (2015b). A transition-based algorithm for AMR parsing. In *Proceedings of the North American Association for Computational Linguistics*.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. In *Proceedings of the annual meeting on Association for Computational Linguistics*.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured clas-

sification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. S. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Syntactic Parsing of Web Queries

Xiangyan Sun
Fudan University

Haixun Wang
Facebook

Yanghua Xiao*
Fudan University

Zhongyuan Wang
Microsoft Research

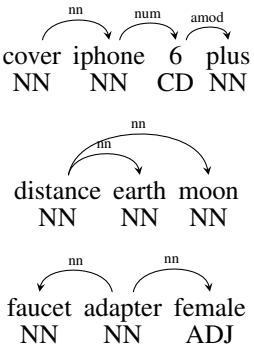
Abstract

Syntactic parsing of web queries is important for query understanding. However, web queries usually do not observe the grammar of a written language, and no labeled syntactic trees for web queries are available. In this paper, we focus on a query's clicked sentence, i.e., a well-formed sentence that i) contains all the tokens of the query, and ii) appears in the query's top clicked web pages. We argue such sentences are semantically consistent with the query. We introduce algorithms to derive a query's syntactic structure from the dependency trees of its clicked sentences. This gives us a web query treebank without manual labeling. We then train a dependency parser on the treebank. Our model achieves much better UAS (0.86) and LAS (0.80) scores than state-of-the-art parsers on web queries.

1 Introduction

Syntactic analysis is important in understanding a sentence's grammatical constituents, parts of speech, syntactic relations, and semantics. In this paper, we are concerned with the syntactic structure of a short text. The challenge is that short texts, for example, web queries, do not observe grammars of written languages (e.g., users often overlook capitalization, function words, and word order when creating a web query), and applying parsers trained on standard treebanks on queries leads to poor results.

Syntactic structures are valuable for query understanding. Consider the following web queries and their syntactic structures we would like to construct:



The syntactic structure of query `cover iphone 6 plus` tells us that the head token is `cover`, indicating its intent is to shop for the cover of an iphone, instead of iphones. With this knowledge, search engines show ads of iphone covers instead of iphones. For `distance earth moon`, the head is `distance`, indicating its intent is to find the distance between the earth and the moon. For `faucet adapter female`, the intent is to find a female faucet adapter. In summary, correctly identifying the head of a query helps identify its intent, and correctly identifying the modifiers helps rewrite the query (e.g., dropping non-essential modifiers).

Syntactic parsing of web queries is challenging for at least two reasons. First, grammatical signals from function words and word order are not available. Query `distance earth moon` is missing function words *between* (preposition), *and* (coordinator), and *the* (determiner) in conveying the intent

* Correspondence author. This paper was supported by National Key Basic Reserach Program of China under No.2015CB358800, by National NSFC(No.61472085, 61171132, 61033010, U1509213), by Shanghai Municipal Science and Technology Commission foundation key project under No.15JC1400900.

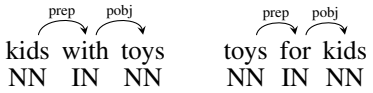
Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1787–1796,
Austin, Texas, November 1-5, 2016.

distance between the earth and the moon. Also, it is likely that queries {`distance earth moon`, `earth moon distance`, `earth distance moon`, $\cdots$} have the same intent, which means they should have the same syntactic structure. Second, there is no labeled dependency trees (treebank) for web queries, nor is there a standard for constructing such dependency trees. It will take a tremendous amount of time and effort to come up with such a standard and a treebank for web queries.

In this paper, we propose an end-to-end solution from treebank construction to syntactic parsing for web queries. Our model achieves a UAS of 0.830 and an LAS of 0.747 on web queries, which is dramatic improvement over state-of-the-art parsers trained from standard treebanks.

2 Our Approach

The biggest challenge of syntactic analysis of web queries is that they do not contain sufficient grammatical signals required for parsing. Indeed, web queries can be very ambiguious. For example, `kids toys` may mean either `toys for kids` or `kids with toys`, for which the dependency relationships between `toys` and `kids` are totally opposite.

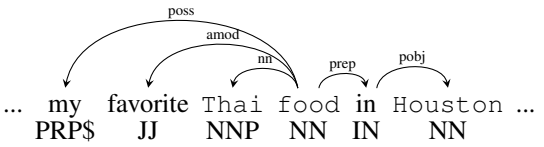


In view of this, why is syntactic parsing of web queries a legitimate problem? We have shown some example syntactic structures for 3 queries in Section 1. How do we know they are the correct syntactic structures for the queries? We answer these questions here.

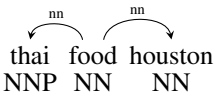
2.1 Derive syntax from semantics

In many cases, humans can easily determine the syntax of a web query because its intent is easy to understand. For example, for `toys kids`, we are pretty sure as a web query, its intent is to look for toys for kids, instead of the other way around. Thus, `toys` should be the head of the query, and `kids` should be its modifier. In other words, when the semantics of a query is understood, we can often recover its syntax.

We may then manually annotate web queries. Specifically, given a query, a human annotator forms a sentence that is consistent with the meaning he comes up for the query. Then, from the sentence's syntactic structure (which is well understood and can be derived by a parser), the annotator derives the syntactic structure of the query. For example, for query `thai food houston`, the annotator may formulate the following sentence:



Then we may project the dependency tree of the sentence to the query:



The above approach has two issues. First, `food` and `houston` are not directly connected in the dependency tree of the sentence. We connected them in the query, but in general, it is not trivial to infer synatx of the query from sentences in a consistent way. There is no linguistic standard for doing this. Second, annotation is very costly. A treebank project takes years to accomplish.

2.2 Semantics of a web query

To avoid human annotation, we derive syntactic understanding of the query from semantic understanding of the query. Our goal is to decide for any two tokens $x, y \in q$, whether there is a dependency arc between x and y, and if yes, what the dependency is.

Context-free signals. One approach to determine the dependency between x and y is to directly model $P(e|x, y)$, where e denotes the dependency ($x \rightarrow y$ or $x \leftarrow y$). It is context-free because we do not condition on the query where x and y appear in.

To acquire $P(e|x, y)$, we may consider annotated corpora such as Google's syntactic ngram (Goldberg and Orwant, 2013). For any x and y, we count the number of times that x is a dependent of y in the corpus. One disadvantage of this approach is that web queries and normal text differ significantly in distribution. Another approach (Wang et al., 2014) is to use search log to estimate $P(e|x, y)$, where x and y are nouns. Specifically, we find queries of pattern x PREP y, where PREP is a preposition {of, in, for, at, on, with, $\cdots$}. We have $P(x \rightarrow y|x, y) =$

$\frac{n_{x,y}}{n_{x,y}+n_{y,x}}$ where $n_{x,y}$ denotes the number of times pattern x PREP y appears in the search log. The disadvantage is that the simple pattern only gives dependency between two nouns.

Context-sensitive signals. The context-free approach has two major weaknesses: (1) It is risky to decide the dependency between two tokens without considering the context. (2) Context-free signals do not reveal the type of dependency, that is, it does not reveal the linguistic relationship between the head and the modifier.

To take context into consideration, which means estimating $P(e|x, y, q)$ for any two tokens $x, y \in q$, we are looking at the problem of building a parser for web queries. This requires a training dataset (a treebank). In this work, we propose to automatically create such a treebank. The feasibility is centered on the following assumption: The intent of q is contained in or consistent with the semantics of its **clicked sentences**. We call sentence s a clicked sentence of q if i) s appears in a top clicked page for q, and ii) s contains all tokens in q. For instance, assume sentence $s =$ "`... my favorite Thai food in Houston ...`" appears in one of the most frequently clicked pages for query $q =$ `thai food houston`, then s is a clicked sentence of q. It follows from the above assumption that the dependency between any two tokens in q are likely to be the same as the dependency between their corresponding tokens in s. This allows to create a treebank if we can project the dependency from sentences to queries. However, since x and y may not be directly connected by a dependency edge in s, we need a method to derive the dependency between $x, y \in q$ from the (indirect) dependency between $x, y \in s$. We propose such a method in Section 3.

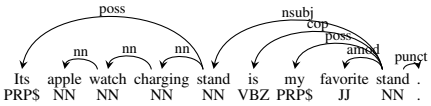
3 Treebank for Web Queries

We create a web query treebank by projecting dependency from clicked sentences to queries.

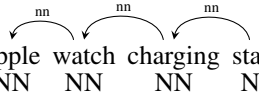
3.1 Inferring a dependency tree

A query q may have multiple clicked sentences. We describe here how we project dependency to q from such a sentence s. We describe how we aggregate dependencies from multiple sentences in Sec 3.2.

Under our assumption, each token $x \in q$ must appear in sentence s. But x may appear multiple times in s (especially when x is a function word). As an example, for query `apple watch stand`, we may get the following sentence:



Sentence s contains token `stand` twice, but only one subtree contains each token in q exactly once.



We use the following heuristics to derive a dependency tree for query q from sentence s.

1. Let T_s denote all the subtrees of the dependency tree of s.
2. Find the minimum subtree $t \in T_s$ such that each $x \in q$ has one and only one match $x' \in t$.
3. Derive dependency tree $t_{q,s}$ for q from t as follows. For any two tokens x and y in q:
   (a) if there is an edge from x' to y' in t, we create a same edge from x to y in $t_{q,s}$.
   (b) if there is a path[1] from x' to y' in t, we create an edge from x to y in $t_{q,s}$, and label it temporarily as *dep*.

We note the following. First, we argue that if the dependency tree of s has a subtree that contains each token in q once and only once, then it is very likely that the subtree expresses the same semantics as the query. On the other hand, if we cannot find such a subtree, it is an indication that we cannot derive reasonable dependency information from the sentence.

Second, it's possible x' and y' are not connected directly in s but through one or more other tokens. Thus, we do not know the label of the derived edge. We will decide on the label in Sec 3.3.

Third, we want to know whether it is meaningful to connect x and y in q while x' and y' are not directly connected in s. We evaluated a few hundreds

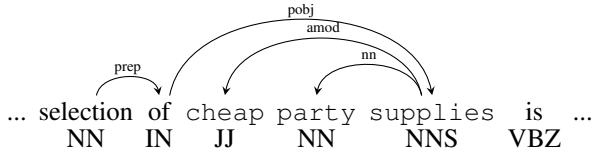
[1] A path consists of edges of the same direction.

of query-sentence pairs. Among the cases where dependency trees for queries can be derived successfully, we found that x' and y' are connected in 5 possible ways (Table 1). We describe them in details next.

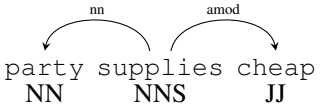
directly connected	46%
connected via function words	24%
connected via modifiers	24%
connected via a head noun	4%
connected via a verb	2%

Table 1: Dependency Projection

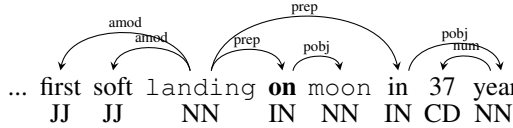
Directly connected. In this case, we copy the edge and its label directly. Consider query `party supplies cheap`'s clicked sentence below:



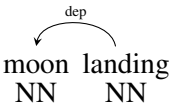
Here both (`party`, `supplies`) and (`supplies`, `cheap`) are directly connected. The query inherits the dependencies, but note that tokens `supplies` and `cheap` have different word orders in q and s:



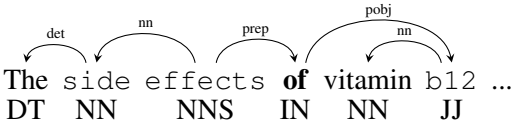
Connected via function words. It is quite common prepositions are omitted in a query. Consider query `moon landing`'s clicked sentence:



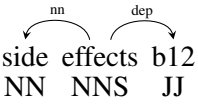
We can derive the following dependency tree:



For query `side effects b12`, suppose we have the following sentence:



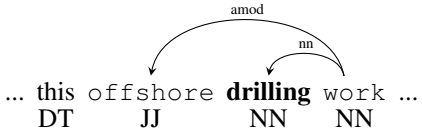
The derived dependency tree should be:



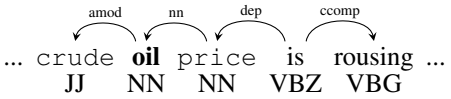
For these two cases, we need to introduce a derived edge for the query, which will be resolved later to a specific dependency label.

Connected via modifiers. Many web queries are noun compounds. Their clicked sentences may have more modifiers. Depending on the bracketing, we may or may not have direct dependencies.

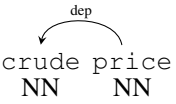
For `offshore work` and its clicked sentence below, missing `drilling` in the query does not cause any problem: `offshore` and `work` are still directly connected in the dependency tree.



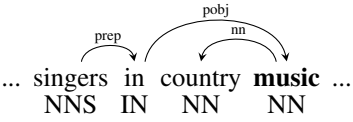
But not for `crude price` and its clicked sentence. Still, there is a path: `crude` ← `oil` ← `price`.



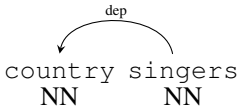
In this case, we create a dependency between `crude` and `oil` in the query and give it a temporary label *dep*. We will resolve it to a specific label later.



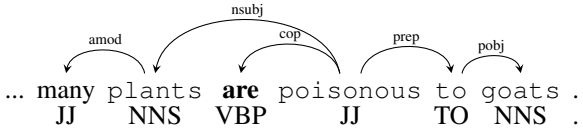
Connected via a head noun. In some cases, the head of a noun compound is missing. Consider `country singers` and its clicked sentence:



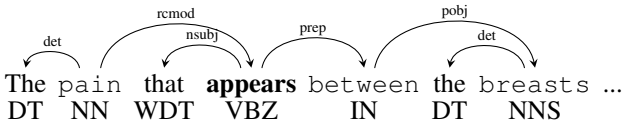
Clearly they mean the same thing, but the head (`music`) of the noun compound is missing in the query. Still, a path exists from `singers` to `country`, and we create a dependency:



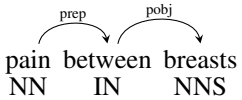
Connected via a verb. One common case is the omission of copular verbs. Consider `plants poisonous to goats` and its clicked sentence:



Here, the missing `are` does not cause any problem. But for query `pain between breasts` and its clicked sentence:



we need to introduce a derived edge, and it leads to:



3.2 Inferring a unique dependency tree

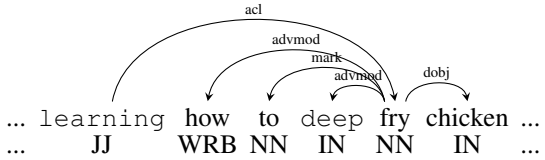
A query corresponds to multiple clicked sentences. From each sentence, we derive a dependency tree. These dependency trees may not be the same, because i) dependency parsing for sentences is not perfect; ii) queries are ambiguous; or iii) some queries do not have well-formed clicked sentences.

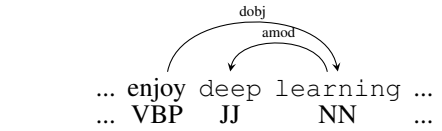
To choose a unique dependency tree for a query q, we define a scoring function f to measure the quality of a dependency tree t_q derived from q's clicked sentence s:

$$f(t_q, s) = \sum_{(x \to y) \in t_q} -\alpha dist(x, y) + \log \frac{count(x \to y)}{count(x \leftarrow y)} \quad (1)$$

where $(x \to y)$ is an edge in the tree t_q, $count(x \to y)$ is the occurrence count of the edge $x \to y$ in the entire query dataset, $dist(x, y)$ is the distance of words x and y on the original sentence parsing tree, and α is a parameter to adjust the importance between the two measures (its value is empirically determined).

The first term of the scoring function measures the *compactness* of the query tree. Consider two clicked sentences for query `deep learning`:





In the first sentence, `deep` and `learning` are indirectly connected through `fry` so the total distance measure is 2. In the second query, the distance is 1. Therefore, query aligned with the second sentence is better than the first sentence.

The second term of the scoring function measures the *global consistency* among head modifier directions. For a word pair (x, y), if in the dataset, the number of edges $x \to y$ dominates the number of edges $x \leftarrow y$, then the latter is likely to be incorrect.

One important thing to note is word order. Word order may influence the head-modifier relations between two words. For example, `child of` and `of child` should definitely have different head-modifier relations. Therefore, we treat two words of different order as two different word pairs.

Table 2 shows some examples of conflicting dependency edges and their corresponding occurrence count in queries and sentences.

Correct	Wrong	Query	Sentence
side ← effects	side → effects	1110:1	11257:17
benefits → of	benefits ← of	144:63	5228:0
Full ← Movie	Full → Movie	128:5	1585:27
coconut ← oil	coconut → oil	91:10	1507:46
credit ← card	credit → card	96:2	4394:60

Table 2: Examples of globally inconsistent head modifier relations

3.3 Label refinement

In Section 3.1, some dependencies are derived with a placeholder label *dep*. Before we use the data to train a parser, we must resolve *dep* to a true label, otherwise they introduce inconsistency in the training data. For example, consider a simple query `crude price`. From clicked sentences that contain `crude oil price`, we de-

rive crude$\xleftarrow{\text{dep}}$price, but from those that contain crude price, we derive crude$\xleftarrow{\text{amod}}$price.

To resolve *dep*, we resort to majority vote first. For any $x \xleftarrow{\text{dep}} y$, we count the occurrence of $x \xleftarrow{\text{label}} y$ in the training data for each concrete label. If the frequency of a certain label is dominating by a predetermined threshold (10 times more frequent than any other label), then we resolve *dep* to that label.

With our training data, the above process is able to resolve about 90% dependencies. We can simply discard queries that contain unresolvable dependencies. However, such queries still contain useful information, for example, the direction of this edge, and the directions and labels of all the other edges. We develop a bootstrapping method to preserve such useful information. First, we train a parser on data without *dep* labels. This skips about 10% queries in our experiments. Second, we use the parser to predict the unknown label. If the prediction is consistent with the annotation except for the *dep* label, we use the predicted label. Third, we add the resolved queries into the training data and train a final parser. Experiments show the bootstrapping approach improves the quality of the parser.

4 Dependency Parsing

We train a parser from the web query treebank data. We also try to incorporate context-free head-modifier signals into parsing. To make it easier to incorporate such signals, we adopt a neural network approach to train our POS tagger and parser.

4.1 Neural network POS tagger and parser

We first train a neural network POS tagger for web queries. For each word in the sentence, we construct features out of a fixed context window centered at that word. The features include the word itself, case (whether the first letter, any letter, or every letter in the word, is in uppercase), prefix, and suffix (we recognize a pre-defined set of prefixes and suffixes, for the rest we use a special token "UNK"). For the word feature, we use pre-trained word2vec embeddings. For word case and prefix/suffix, we use random initialization for the embeddings. The accuracy of the trained POS tagger is similar to that of (Ganchev et al., 2012), which outperforms POS taggers trained on PTB data.

Buffer features
$b_1.wt$, $b_2.wt$, $b_3.wt$
Stack features
$s_1.wt$, $s_2.wt$, $s_3.wt$
Tree features
$lc_1(s_1).wtl$, $lc_2(s_1).wtl$, $rc_1(s_1).wtl$, $rc_2(s_1).wtl$
$lc_1(lc_1(s_1)).wtl$, $rc_1(rc_1(s_1)).wtl$
$lc_1(s_2).wtl$, $lc_2(s_2).wtl$, $rc_1(s_2).wtl$, $rc_2(s_2).wtl$
$lc_1(lc_1(s_2)).wtl$, $rc_1(rc_1(s_2)).wtl$

Table 3: The feature templates. $s_i(i = 1, 2, ...)$ denote the i^{th} top element of the stack, $b_i(i = 1, 2, ...)$ denote the i^{th} element on the buffer, $lc_k(s_i)$ and $rc_k(s_i)$ denote the kth leftmost and rightmost children of s_i, w denotes words, t denotes POS tag, l denotes label.

We use the arc standard transition based dependency parsing system (Nivre, 2004). The architecture of the neural network dependency parser is similar to that of (Chen and Manning, 2014) designed for parsing sentences. The features used in parsing are shown in Table 3.

4.2 Context free features

In Section 2.2, we discussed context-free signals $P(e|x, y)$ and context-sensitive signals $P(e|x, y, q)$. Previous work (Wang et al., 2014) uses context-free signals for syntactic analysis of a query. Our approach outperforms the context-free approach.

An interesting question is, will context-free signals further improve our approach? The rationale is that although context-sensitive signals $P(e|x, y, q)$ are more accurate in predicting the dependency between x and y, such signals are also very sparse. Do context-free signals $P(e|x, y)$ provide backoff information in parsing?

It is not straightforward to include $P(e|x, y)$ in the neural network model. The head-modifier relations $P(e|x, y)$ may exist between any pair of tokens in the input query. Essentially, it is a pairwise graphical model and it is difficult to directly incorporate the signals in transition based dependency parsing.

We treat context-free signals as *prior knowledge*. We train head-modifier embeddings for each token, and use such embeddings as pre-trained embeddings. Specifically, we use an approach similar to training word2vec embeddings but focusing on head

modifier relationships instead of co-occurrence relationships. More specifically, we train an one hidden layer neural network classifier to determine whether two words have head-modifier relations. The input of the neural network is the concatenation of the embeddings of two words. The output is whether the two words form a proper head-modifier relationship. We obtain a large set of head-modifier data from text corpus by mining "h PREP m" pattern in search log where h and m are nouns. Then, for each known head modifier pair h and m, we use (h, m) as positive example and (m, h) as negative example. For each word, we also choose a few random words as negative examples. During the training process, the gradients are back propagated to the word embeddings. After training, the embeddings should contain sufficient information to recover head modifier relations between any word pairs.

But we did not observe improvement over the existing neural network that are trained on context sensitive treebank data alone. The head-modifier embeddings has about 3% advantage in UAS over randomized embeddings. However, using pretrained word2vec embeddings, we also achieve 3% advantage. Thus, it seems that context-sensitive signals plus the generalizing power of embeddings contain all the context-free signals already.

5 Experiments

In this section, we start with some case studies. Then we describe data and compare models.

In experiments, we use the standard UAS (unlabeled attachment score) and LAS (labeled attachment score) score for measuring the quality of dependency parsing. They are calculated as:

$$UAS = \frac{\#\ correct\ arc\ directions}{\#\ total\ arcs} \quad (2)$$

$$LAS = \frac{\#\ correct\ arc\ directions\ and\ labels}{\#\ total\ arcs} \quad (3)$$

5.1 Case Study

We compare dependency trees produced by our QueryParser and Stanford Parser (Chen and Manning, 2014) for some web queries (Stanford Parser is trained from the standard PTB treebank). Table 4 shows that Stanford Parser heavily relies on grammar signals such as function words and word order, while QueryParser relies more on the semantics of the query. For instance, in the 1st example, QueryParser identifies `toys` as the head, regardless of the word order, while Stanford parser always assumes the last token as the head. In the 2nd example, the semantics of the query is a school (`vanguard school`) at a certain location (`lake wales`). QueryParser captures the semantics and correctly identifies `school` as the head (root) of the query, while Stanford parser treats the entire query as a single noun compound (likely inferred from the POS tags).

5.2 Clicked Sentences

For training data, we use one-month Bing query log (between July 25, 2015 and August 24, 2015). From the log, we obtain web query q and its top clicked URLs $\{url_1, url_2, ..., url_m\}$. From the urls, we retrieve the clicked HTML document, and find sentences $\{s_1, s_2, ..., s_n\}$ that contain all words (regardless to their order of occurrence) in q. Then we extract query-sentence tuples $(q, s, count)$ to serve as our training data to generate a web query treebank. The size (# of distinct query-sentence pairs) of the raw clicked sentences is 390,225,806.

5.3 Web Query Treebank

We evaluate the 3 steps of treebank generation. After each step, we sample 100 queries from the result and manually compute their UAS and LAS scores. We also count the number of total query instances in each step. The results are shown in Table 5.

- **Inferring a dependency tree**: For each (query, sentence) pair, we project dependency from the sentence to the query. The number of instances shown in Table 5 are the input number of (query, sentence) pairs. It shows that we obtain dependency trees for only 31% of the queries, while the rest do not satisfy our filtering criterion. This however is not a concern. By sacrificing recall in this process, we ensure high precision. Given that query log is large, precision is more important.

- **Inferring a unique dependency tree**: In this step, we group (query, sentence) pairs by unique queries. Using the method in Section

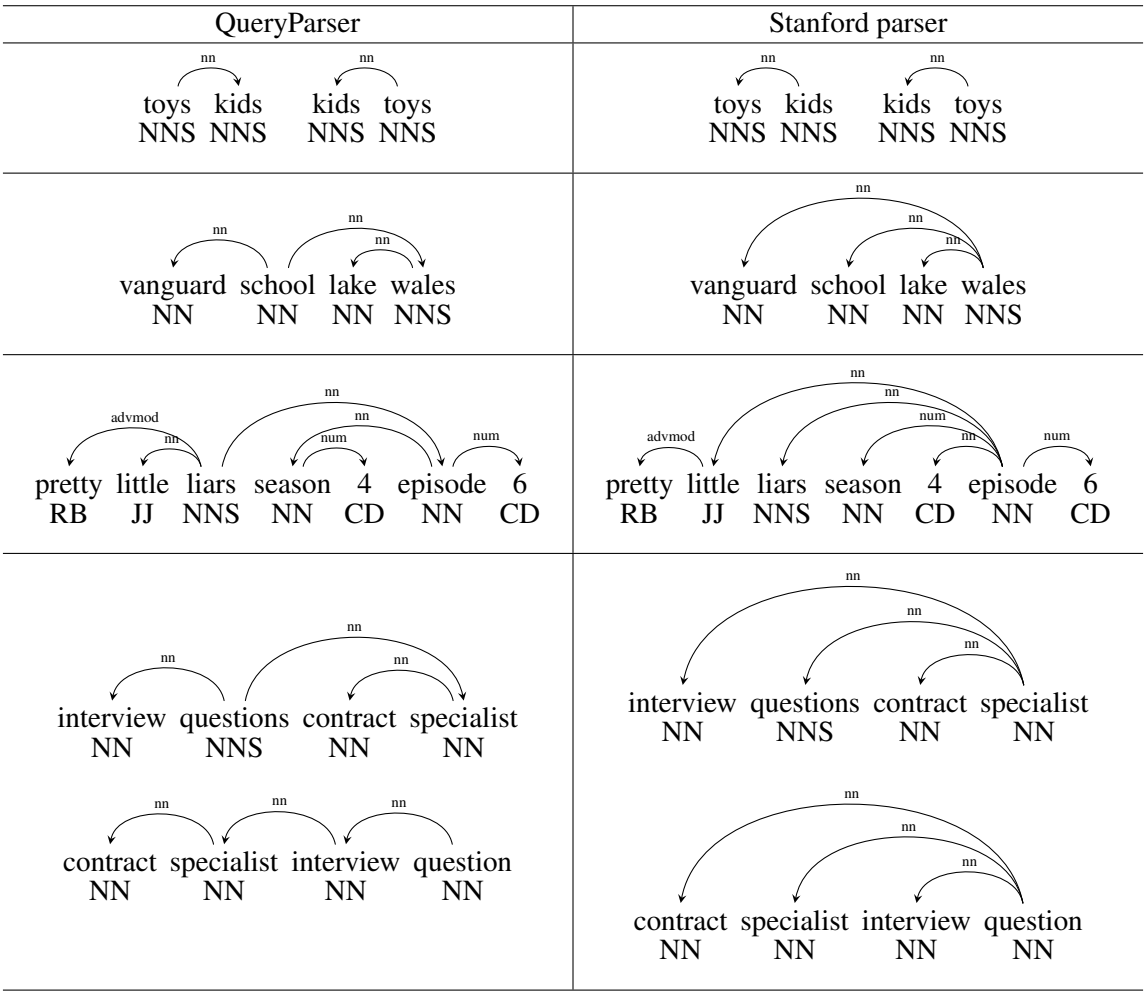


Table 4: Case study of parsers.

3.2, each group produces one or zero dependency trees. The number of instances in Table 5 corresponds to the number of different query groups. The overall success rate is high. This is expected as the filtering process uses majority voting, and we already have high precision parsing trees after the first step.

- **Label refinement**: Dependency labels are refined using the methodology in Section 3.3. It shows that with majority voting and bootstraping, we are able to keep all the input.

5.4 Parser Performance

We compare QueryParser against three state-of-the-art parsers: Stanford parser, which is a transition based dependency parser based on neural network, MSTParser (McDonald et al., 2005), which is a graph based dependency parser based on minimum spanning tree algorithms, and LSTMParser (Dyer et al., 2015), which is a transition based dependency parser based on stack long short-term memory cells. Here, QueryParser is trained from our web query treebank, while Stanford Parser and MSTParser are trained from standard PTB treebanks.

For comparison, we manually labeled 1,000 web queries to serve as a ground truth dataset[2]. We produce POS tags for the queries using our neural network POS tagger. To specifically measure the ability of QueryParser in parsing queries with no explicit syntax structure, we split the entire dataset **All** into two parts: **NoFunc** and **Func**, which correspond to queries without any function word, and queries with at least one function word. The number of queries

[2]https://github.com/wishstudio/queryparser

Step	Total Instances	Produced Instances	Success Rate	UAS	LAS
Inferring a dependency tree	3986300	1229860	31%	0.906	0.851
Inferring a unique tree	716261	680857	95%	0.910	0.851
Label refinement	680857	680857	100%	0.917	0.855

Table 5: Training dataset generation statistics

System	**All** (n=1000)		**NoFunc** (n=900)		**Func** (n=100)	
	UAS	LAS	UAS	LAS	UAS	LAS
Stanford	0.694	0.602	0.670	0.568	0.834	0.799
MSTParser	0.699	0.616	0.683	0.691	0.799	0.766
LSTMParser	0.700	0.608	0.679	0.578	0.827	0.790
QueryParser + label refinement	0.829	0.769	0.824	0.761	0.858	0.818
QueryParser + word2vec	0.843	0.788	0.843	0.784	0.838	0.812
QueryParser + label refinement + word2vec	**0.862**	**0.804**	**0.858**	**0.795**	**0.883**	**0.854**

Table 6: Parsing performance on web queries

of the two datasets are 900 and 100, respectively.

Table 6 shows the results. We use 3 versions of QueryParser. The first two use random word embedding for initialization, and the first one does not use label refinement. From the results, it can be concluded that QueryParser consistently outperformed competitors on query parsing task. Pretrained word2vec embeddings improve performance by 3-5 percent, and the postprocess of label refinement also improves the performance by 1-2 percent.

Table 6 also shows that conventional depencency parsers trained on sentence dataset relies much more on the syntactic signals in the input. While Stanford parser and MSTParser have similar performance to our parser on **Func** dataset, the performance drops significantly on **All** and **NoFunc** dataset, when the majority of input has no function words.

6 Related Work

Some recent work (Ganchev et al., 2012; Barr et al., 2008) investigated the problem of syntactic analysis for web queries. However, current study is mostly at postag rather than dependency tree level. Barr et al. (2008) showed that applying taggers trained on traditional corpora on web queries leads to poor results. Ganchev et al. (2012) propose a simple, efficient procedure in which part-of-speech tags are transferred from retrieval-result snippets to queries at training time. But they do not reveal syntactic structures of web queries.

More work has focused on resolving simple relations or structures in queries or short texts, particularly entity-concept relations (Shen et al., 2006; Wang et al., 2015; Hua et al., 2015), entity-attribute relations (Pasca and Van Durme, 2007; Lee et al., 2013), head-modifier relations (Bendersky et al., 2010; Wang et al., 2014). Such relations are important but not enough. The general dependency relations we focus on is an important addition to query understanding.

On the other hand, there is extensive work on syntactic analysis of well-formed sentences (De Marneffe et al., 2006). Recently, a lot of work (Collobert et al., 2011; Vinyals et al., 2015; Chen and Manning, 2014; Dyer et al., 2015) started using neural network for this purpose. In this work, we use similar neural network architecture for web queries.

7 Conclusion

Syntactic analysis of web queries is extremely important as it reveals actional signals to many downstream applications, including search ranking, ads matching, etc. In this work, we first acquire well-formed sentences that contain the semantics of the query, and then infer the syntax of the query from the sentences. This essentially creates a treebank for web queries. We then train a neural network dependency parser from the treebank. Our experiments show that we achieve significant improvement over traditional parsers on web queries.

References

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1021–1030, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Bendersky, Donald Metzler, and W Bruce Croft. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 31–40. ACM.

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

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

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependeny parsing with stack long short-term memory. In *Proc. ACL*.

Kuzman Ganchev, Keith Hall, Ryan McDonald, and Slav Petrov. 2012. Using search-logs to improve query tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 238–242, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2015. Short text understanding through lexical-semantic analysis. In *International Conference on Data Engineering (ICDE)*.

Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. 2013. Attribute extraction and scoring: A probabilistic approach. In *International Conference on Data Engineering (ICDE)*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

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

Marius Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI*, volume 7, pages 2832–2837.

Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138. ACM.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Zhongyuan Wang, Haixun Wang, and Zhirui Hu. 2014. Head, modifier, and constraint detection in short texts. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 280–291. IEEE.

Zhongyuan Wang, Kejun Zhao, Haixun Wang, Xiaofeng Meng, and Ji-Rong Wen. 2015. Query understanding through knowledge-based conceptualization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.

Unsupervised Text Recap Extraction for TV Series

Hongliang Yu and **Shikun Zhang** and **Louis-Philippe Morency**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
{yuhongliang, shikunz, morency}@cs.cmu.edu

Abstract

Sequences found at the beginning of TV shows help the audience absorb the essence of previous episodes, and grab their attention with upcoming plots. In this paper, we propose a novel task, text recap extraction. Compared with conventional summarization, text recap extraction captures the duality of summarization and plot contingency between adjacent episodes. We present a new dataset, *TVRecap*, for text recap extraction on TV shows. We propose an unsupervised model that identifies text recaps based on plot descriptions. We introduce two contingency factors, *concept coverage* and *sparse reconstruction*, that encourage recaps to prompt the upcoming story development. We also propose a multi-view extension of our model which can incorporate dialogues and synopses. We conduct extensive experiments on *TVRecap*, and conclude that our model outperforms summarization approaches.

1 Introduction

According to a study by FX Networks, in U.S., the total number of ongoing scripted TV series hit a new high of 409 on broadcast, cable, and streaming in 2015[1]. Such a large number indicates there are more shows than anyone can realistically watch. To attract prospective audiences as well as help current viewers recall the key plot when airing new episodes, some TV shows add a clip montage, which is called a recap sequence, at the beginning of new episodes or seasons. Recaps not only help the audience absorb the essence of previous episodes, but also grab people's attention with upcoming plots. However, creating those recaps for every newly aired episode is labor-intensive and time-consuming. To our advantage, there are many textual scripts freely available online which describe the events and actions happening during the TV show episodes[2]. These textual scripts contain *plot descriptions* of the events, *dialogues* of the actors, and sometimes also the *synopsis* summarizing the whole episode.

These abundant textual resources enable us to study a novel, yet challenging task: automatic text recap extraction, illustrated in Figure 1. The goal of text recap extraction is to identify segments from scripts which both summarize the current episode and prompt the story development of the next episode. This unique task brings new technical challenges as it goes beyond summarizing prior TV episodes, by introducing a concept of plot contingency to the upcoming TV episode. It differs from conventional summarization techniques which do not consider the interconnectivity between neighboring episodes. Text recaps should capture the duality of summarization and plot contingency between neighboring episodes. To our knowledge, no dataset exists to study this research topic.

In this paper, we present an unsupervised model to automatically extrapolate text recaps of TV shows from plot descriptions. Since we assume recaps should cover the main plot of the current episode and also prompt the story development of the next episode, our model jointly optimizes these two ob-

[1] http://tinyurl.com/jugyyu2

[2] http://www.simplyscripts.com/tv_all.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1797–1806,
Austin, Texas, November 1-5, 2016.

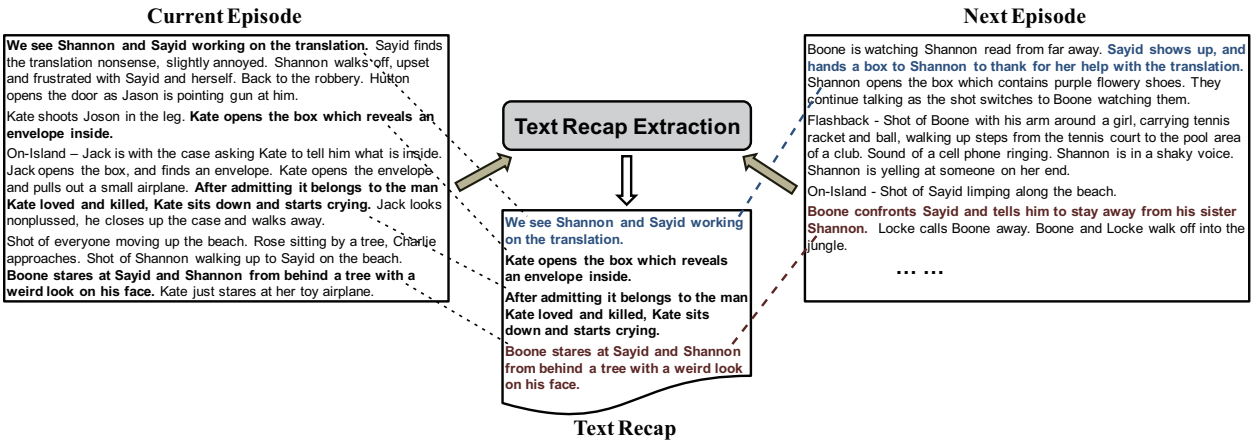


Figure 1: Illustration of text recap extraction. The system extracts sentences from the current episode. The text recap sentences in black summarize the current episode, while colored sentences motivate the next episode.

jectives. To summarize the current episode, our model exploits coverage-based summarization techniques. To connect to the next episode, we devise two types of plot contingency factors between adjacent episodes. These factors implement the coverage and reconstruction assumptions to the next episode. We also show how our model can be extended to integrate dialogues and synopses when available.

We introduce a new dataset[3], named *TVRecap* for text recap extraction which consists of TV series with textual scripts, including descriptions, dialogues and synopses. The dataset enables us to study whether contingency-based methods which exploit relationships between adjacent episodes can improve summarization-based methods.

The rest of this paper is organized as follows. In Section 2, we discuss related work and the motivation for our work. In Section 3, we introduce our new dataset for text recap extraction. Section 4 explains our proposed model for text recap extraction, and Section 5 expands the model by incorporating synopses and dialogues. In Section 6 and 7, we present our experimental results and analyses, and finally conclude our work in Section 8.

2 Related Work

In this section, we discuss three related research topics. Text summarization is an relevant task that aims to create a summary that retains the most important points of the original document. Then we discuss the evaluation metrics of text summarization. Finally, we discuss the video description which is complementary to our work.

Generic Text Summarization Alogrithms Text summarization is widely explored in the news domain (Hong and Nenkova, 2014; McKeown, 2005). Generally, there are two approaches: *extractive* and *abstractive* summarization.

Extractive summarization forms a summary by choosing the most representative sentences from the original corpus. The early system LEAD (Wasson, 1998) was pioneering work. It selected leading text of the document as the summary, and was applied in news searching to help online customers focus their queries on the beginning of news documents. He et al. (2012) assumed that summarization should consist of sentences that could best reconstruct the original document. They modeled relationship among sentences by forming an optimization problem. Moreover, Sipos et al. (2012) and Lin and Bilmes (2010) studied multi-document summarization using coverage-based methods. Among them, Lin and Bilmes (2010) proposed to approximate the optimal solution of a class of functions by exploiting submodularity.

Abstractive summarization automatically create new sentences. For example, compared with the sentence-level analysis in extractive summarization, Bing et al. (2015) explored fine-grained syntactic units, i.e. noun/verb phrases, to represent concepts in input documents. The informative phrases were

[3] http://multicomp.cs.cmu.edu

then used to generate sentences.

In this paper, we generalize the idea of text summarization to text recap extraction. Instead of summarizing a given document or collection, our model emphasizes plot contingency with the next episode.

Summarization Applications Summarization techniques are not restricted to informative resources (e.g. news), applications in broader areas are gaining attention (Aparício et al., 2016). As the prevailance of online forums, Misra et al. (2015) developed tools to recognize arguments from opinionated conversations, and group them across discussions. In entertainment industry, Sang and Xu (2010) proposed a character-based movie summarization approach by incorporating scripts into movie analysis. Moreover, recent applications include multimedia artifact generation (Figueiredo et al., 2015), music summarization (Raposo et al., 2015) and customer satisfaction analysis (Roy et al., 2016).

Video Description Generating video descriptions is a task that studies automatic generation of natural language that describes events happening in video clips. Most work uses sequential learning for encoding temporal information and language generation (Guadarrama et al., 2013; Rohrbach et al., 2013, 2015; Donahue et al., 2015). Our work is complementary to video description: the large number of unlabeled videos can be utilized to train end-to-end recap extraction system when video description models can properly output textual descriptions.

Contributions of This Paper In contrast with prior work, the main contributions of this paper are:
(1) We propose a novel problem, text recap extraction for TV series. Our task aims to identify segments from scripts which both summarize the current episode and prompt the story development of the upcoming episode;
(2) We propose an unsupervised model for text recap extraction from descriptions. It models the episode contingency through two factors, next episode summarization and sparse reconstruction;
(3) We introduce a new dataset for TV show recap extraction, where descriptions, dialogues and synopses are provided.

3 The TVRecap Dataset

We collected a new dataset, called *TVRecap*, for text recap extraction on TV series. We gathered and processed scripts, subtitles and synopses from websites[4] as components to build our model upon. We also established ground truth to help future research on this challenging topic. *TVRecap* includes all seasons from the widely-known show "*Lost*" with a total of 106 episodes. Statistics of our dataset are shown in Table 1.

	# sent.	avg. # sent.	# words	avg. # w./s.
description	14,686	138.5	140,684	9.57
dialogue	37,714	355.8	284,514	7.54
synopsis	453	4.27	7,868	17.36
recap	619	17.19	5,892	9.52

Table 1: Statistics of *TVRecap*.

This section describes how textual scripts and synopses are processed, and how we automatically define the ground truth of text recap annotations.

Descriptions, Dialogues and Synopses A script for one TV series episode is a sequence of dialogues interleaved with descriptions (marked by square brackets). We automatically split the script into descriptions and dialogues. For each episode, We also downloaded the synopsis, a human-written paragraph summarizing the main plot of the episode. Figure 2 shows examples of a script and a synopsis from our *TVRecap* dataset.

LOCKE: Two players. Two sides. One is light... one is dark. Walt, do you want to know a secret?

[*Claire writing in her diary. Jin approaches and offers her some urchin. She shakes her head, but then gives in and takes some.*]

CLAIRE: No. Thank you. No, it's okay. [*Jin keeps insisting*] No, really. Okay. Thanks.

(a) Script: containing descriptions and dialogues.

Boone steals the decreasing water supply in a misguided attempt to help everyone, but the survivors turn on him. A sleep-deprived Jack chases after what appears to be his deceased father in the forests and eventually discovers caves with fresh water. Jack comes to terms with his role as leader. In flashbacks, Jack goes to Australia to retrieve his deceased father.

(b) Synopsis.

Figure 2: Example of a script (including descriptions and dialogues) and a synopsis.

[4] `http://lostpedia.wikia.com/` and `https://www.wikipedia.org/`

All plot descriptions and dialogues are time-aligned automatically using the subtitle files[5]. We first aligned the dialogue sentences from the script with the subtitle files which contain time-stamps (in milliseconds) of the spoken dialogues. Then we estimated time-stamps of description sentences using surrounding dialogues.

Since descriptions sometimes contain words not relevant to the event, we manually post-processed all descriptions and recap sentences as follows: (1) remove trivial sentences such as "*music on*", (2) remove introductory terms like "*Shot of*", (3) complete missing grammatical components (like omitted subjects) of sentences when possible.

Text Recap Annotations The goal of our ground truth annotation is to identify the text descriptions associated with the TV show recap. We performed this annotation task in three steps.

First, we automatically extracted the recap sequence, which is a montage of important scenes from previous episodes to inform viewers of what has happened in the show, from the TV show video. These recap sequences, if available, are always shown at the beginning of TV episodes. We automatically separated video recap sequences from full-length video files by detecting a lengthy appearance of black frames in the first several minutes of the episode. Second, we located the frames of the recap sequences in the videos of previous episodes, and recorded their time-stamps. Finally, the recap annotations are automatically identified by comparing the video time-stamps with the text description time-stamps. A description is annotated as part of the recap if at least 4 frames from the video recap are present during this description.

4 Our Text Recap Extraction Model

In our Text Recap Extraction Model (TREM), we assume a good text recap should have two characteristics: (a) it covers the main plot of the current episode, and (b) it holds plot contingency with the next episode. Under the first assumption, the text recap can be seen as a summarization that retains the most important plots. Under assumption (b), the text recap should capture the connections between two consecutive episodes.

[5] http://www.tvsubtitles.net/

Formally, the system is given E episodes from a specific TV show, where each episode contains textual descriptions. We define these descriptions as $D = \{D^1, \cdots, D^E\}$, where D^i is the set of descriptions of episode i. D^i is composed of descriptive sentences as $D^i = \{d^i_1, \cdots, d^i_{|D^i|}\}$, where d^i_j is the j-th sentence. The goal of our task is to find text recaps $R = \{R^1, \cdots, R^{E-1}\}$ where the components of R^i are selected from D^i with a length budget (constraint on the number of sentences) $|R^i| \leq K$.

In our TREM model, the text recap R^i of the i-th episode is optimized by:

$$\begin{aligned} \max_{R^i \subset D^i} \quad & \mathcal{F}(R^i) = \mathcal{S}(R^i, D^i) + \mathcal{M}(R^i, D^{i+1}) \\ \text{s.t} \quad & |R^i| \leq K, \end{aligned} \tag{1}$$

where $\mathcal{S}(R^i, D^i)$ measures how well R^i summarizes D^i, and $\mathcal{M}(R^i, D^{i+1})$ quantifies the level of connectivity between the text recap of the current episode and the plot description of the next episode. By using $\mathcal{M}(\cdot,\cdot)$, we expect to produce text recaps with better plot contingency with the upcoming story.

In the following sections, we demonstrate in details: (1) the definition of the summarization function $\mathcal{S}(\cdot,\cdot)$; (2) two factors that derive the contingency function $\mathcal{M}(\cdot,\cdot)$ based on different hypotheses.

4.1 Plot Summarization

In this section, we discuss the summarization component of our model's objective function. Our model is inspired by the coverage-based summarization (Lin and Bilmes, 2010), whose key idea is to find a proxy that approximates the information overlap between the summary and the original document. In this work, any text is assumed to be represented by a set of "*concepts*" using weights to distinguish their importance. To be more specific, a *concept* is defined as a noun/verb/adjective or noun/verb phrase. In terms of *concepts*, we define the summarization term $\mathcal{S}(R^i, D^i)$ as follows:

$$\mathcal{S}(R^i, D^i) = \sum_{c \in \mathcal{C}(D^i)} z(c, D^i) \max_{r \in R^i} w(c, r), \tag{2}$$

where $\mathcal{C}(D^i) = \{c' | c' \in d^i_j, \forall d^i_j \in D^i\}$ is the concept set of D^i, and $z(c, D^i)$ measures the importance of c in D^i. We use Term Frequency Inverse

Document Frequency (TF-IDF) (Salton and Buckley, 1988) to calculate $z(c, D^i)$. Finally, $w(c, r)$ denotes the relatedness of a concept c to a sentence r.

We use Word2Vec (Mikolov et al., 2013) vectors as the semantic representation of concepts, and define $w(c, r)$ as:

$$w(c, r) = |c| \cdot \max_{c' \in r} \cos(\mathbf{c}, \mathbf{c}'), \tag{3}$$

where bold notations are the Word2Vec representations of c and c'. Note that if c is a phrase, $\mathbf{c}$ is mean pooled by the embeddings of its component words. $|c|$ is the number of words in c.

4.2 Plot Contingency

We model plot contingency on the concept level as well as on the sentence level. Therefore, the component $\mathcal{M}(\cdot, \cdot)$ is decomposed into two factors:

$$\begin{aligned}\mathcal{M}(R^i, D^{i+1}) =& \lambda_s \mathcal{M}_s(R^i, D^{i+1}) + \\ & \lambda_r \mathcal{M}_r(R^i, D^{i+1}).\end{aligned} \tag{4}$$

where $\mathcal{M}_s(R^i, D^{i+1})$ measures how well R^i can summarize the next episode D^{i+1} and $\mathcal{M}_r(R^i, D^{i+1})$ is the factor that quantify the ability of R^i to reconstruct D^{i+1}. $\lambda_s, \lambda_r \geq 0$ are coefficients for $\mathcal{M}_s(\cdot, \cdot)$ and $\mathcal{M}_r(\cdot, \cdot)$ respectively. In the following sections, we define and explain these two factors in details.

4.2.1 Concept Coverage

Following the coverage assumption of Section 4.1, we argue that the text recap should also cover important concepts from the next episode. Therefore, the first contingency factor can be defined in the same form as the summarization component where D^i's in Equation 2 are replaced by D^{i+1}'s:

$$\mathcal{M}_s(R^i, D^{i+1}) = \sum_{c \in \mathcal{C}(D^{i+1})} z(c, D^{i+1}) \max_{r \in R^i} w(c, r). \tag{5}$$

4.2.2 Sparse Reconstruction

As events happening in the current episode can have an impact on the next episode, there exist hidden connections between the descriptive sentences in D^i and D^{i+1}. To be more specific, assuming descriptive sentences from D^{i+1} are dependent on a few sentences in D^i, we aim to infer such hidden contingency. Here we assume that sentence d_j^{i+1} is related to a small number of sentences in D^i.

Let $\boldsymbol{\alpha}_j^{i+1} \in \mathbb{R}^{|D^i|}$ be the indicator that determines which sentences in D^i prompt d_j^{i+1}, and $\mathbf{W}$ be the matrix that transforms these contingent sentences to the embedding space of d_j^{i+1}. Intuitively, our model learns $\mathbf{W}$ by assuming each sentence in D^{i+1} can be reconstructed by contingent sentences from D^i:

$$\mathbf{d}_j^{i+1} \approx \mathbf{W}\mathbf{D}^i \boldsymbol{\alpha}_j^{i+1}, \tag{6}$$

In the equation, we first convert every description sentence to its distributed representation using the pre-trained skip-thought model proposed by Kiros et al. (2015). The sentence embedding is denoted in bold (e.g. $\mathbf{d}_j^i$ for sentence d_j^i). $\mathbf{D}^i = [\mathbf{d}_1^i; \cdots; \mathbf{d}_{|D^i|}^i]$ stacks the vector representations of all sentences in D^i, and $\boldsymbol{\alpha}_j^{i+1}$ linearly combines the contingent sentences.

We propose to jointly optimize $\boldsymbol{\alpha}_j^{i+1}$ and $\mathbf{W}$ by:

$$\begin{aligned}\min_{\{\boldsymbol{\alpha}^{i+1}\}_{i=1}^{E-1}, \mathbf{W}} & \sum_{i,j} \big(\|\mathbf{W}\mathbf{D}^i \boldsymbol{\alpha}_j^{i+1} - \mathbf{d}_j^{i+1}\|_2^2 \\ & + \gamma \|\boldsymbol{\alpha}_j^{i+1}\|_1 \big) + \theta \|\mathbf{W}\|_F^2,\end{aligned} \tag{7}$$

where we denote $\boldsymbol{\alpha}^{i+1} = [\boldsymbol{\alpha}_1^{i+1}; \cdots; \boldsymbol{\alpha}_{|D^{i+1}|}^{i+1}]$. We impose sparsity constraint on $\boldsymbol{\alpha}_j^{i+1}$ with L_1 norm such that only a small fraction of sentences in D^i will be linked to d_j^{i+1}. γ and θ are coefficients of the regularization terms.

Given the optimal $\mathbf{W}^*$ from Equation 7, our main objective is to identify the subset of descriptions in D^i that best capture the contingency between D^i and D^{i+1}. The reconstruction contingency factor can be defined as:

$$\mathcal{M}_r(R^i, D^{i+1}) = \sum_{d \in D^{i+1}} \max_{r \in R^i} \mathbf{r}^\intercal \mathbf{W}^* \mathbf{d}. \tag{8}$$

4.3 Optimization

In this section, we describe our approach to optimize the main objective function expressed in Equations 1 and 7.

Finding an efficient algorithm to optimize a set function like Equation 1 is often challenging. However, it can be easily shown that the objective function of Equation 1 is submodular, since all its components $\mathcal{S}(R^i, D^i), \mathcal{M}_s(R^i, D^{i+1})$ and

$\mathcal{M}_r(R^i, D^{i+1})$ are submodular with respect to R^i. According to Lin and Bilmes (2011), there exists a simple greedy algorithm for monotonic submodular function maximization where the solution is guaranteed to be close to the real optimum. Specifically, if we denote R^i_{greedy} as the approximation optimized by greedy algorithm and R^{i*} as the best possible solution, then $\mathcal{F}(R^i_{greedy}) \geq (1 - \frac{1}{e}) \cdot \mathcal{F}(R^{i*})$, where $\mathcal{F}(\cdot)$ is the objective function of Equation 1 and $e \approx 2.718$ denotes the natural constant. The greedy approach is shown in Algorithm 1.

Algorithm 1 Text Recap Extraction

Input: Vectorized sentence representations $\{\mathbf{D}^i\}_{i=1}^E$, parameters $\lambda_s, \lambda_r, \theta, \gamma$, budget K, optimal $\mathbf{W}^*$ for Equation 7.
Output: Text recaps $\{R^i\}_{i=1}^E$.

```
for i = 1, ..., E
    Initialize R^i ← ∅;
    REPEAT
        r* ← arg max F(R^i ∪ {r});
        R^i ← R^i ∪ {r*};
    UNTIL |R^i| ≥ K
end
```

Algorithm 1 requires the optimal $\mathbf{W}^*$ learned from the adjacent episode pairs in Equation 7. We utilize the algorithm that iteratively updates $\mathbf{W}$ and $\boldsymbol{\alpha}$ given the current solution. At each iteration, each variable ($\mathbf{W}$ or $\{\boldsymbol{\alpha}^{i+1}\}$) is updated by fixing the other. At t-th iteration, $\mathbf{W}^{(t)}$ is computed as the solution of ridge regression (Hoerl and Kennard, 1970):

$$\mathbf{W}^{(t)} = \mathbf{D}\mathbf{X}^\intercal(\mathbf{X}\mathbf{X}^\intercal + \theta\mathbf{I})^{-1}, \tag{9}$$

where $\mathbf{D}$ and $\mathbf{X}$ stack all $\mathbf{d}_j^{i+1}$ and $\mathbf{x}_j^{i+1} \triangleq \mathbf{D}^i\boldsymbol{\alpha}_j^{i+1}, \forall i = 1, \cdots, E-1, j = 1, \cdots, |D^i|$. Fixing $\mathbf{W}$, each $\boldsymbol{\alpha}_j^{i+1}$ can be solved separately by general sparse coding algorithms as stated in Mairal et al. (2009). Algorithm 2 shows the optimization process of Equation 7.

5 Multi-View Recap Extraction

In addition to plot descriptions, there are also dialogues and plot synopses available for TV shows. Descriptions, dialogues and synopses can be seen as three different views of the same TV show episode.

Algorithm 2 Reconstruction Matrix Optimization

Input: Vectorized sentence representations $\{\mathbf{D}^i\}_{i=1}^E$, θ and γ.
Output: Contingency matrix $\mathbf{W}$.

```
Initialize W^(0) ← I and α_j^{i+1(0)} ← 0, ∀i, j;
Initialize iteration step t ← 0;
REPEAT
    t ← t + 1;
    W^(t) is updated according to Equation 9;
    ∀i, j, α_j^{i+1,(t)} ← sparse_coding(W^(t));
UNTIL ||W^(t) − W^(t−1)||_F^2 ≤ ε
```

Previously, we build TREM using plot descriptions. In this section, we expand our TREM model to incorporate plot synopses and dialogues. We define text synopses and dialogues as $S = \{S^1, \cdots, S^E\}$ and $T = \{T^1, \cdots, T^E\}$, where S^i and T^i are the set of sentences from synopses and dialogues of the i-th episode.

Dialogues In TV shows, a lot of useful information is presented via actors' dialogues which motivates us to extend our TREM model to include dialogues. Both views can be used to identify recap segments which are assumed to be summative and contingent. Denote the neighboring dialogues of R^i as $N(R^i) = \{t \in T^i \big| \exists r \in R^i, \text{ s.t. } |\text{time}(t) - \text{time}(r)| < \delta\}$, we extend the optimization objective (Equation 1) into:

$$\begin{aligned}\mathcal{F}(R^i) = &\big(\mathcal{S}(R^i, D^i) + \mathcal{S}(N(R^i), T^i)\big) \\ &+ \big(\mathcal{M}(R^i, D^{i+1}) + \mathcal{M}(N(R^i), T^{i+1})\big).\end{aligned} \tag{10}$$

Synopses Since a synopsis is a concise summary of each episode, we can treat plot summarization as text alignment where R^i is assumed to match the content of S^i. Therefore, the summarization term can be redefined by substituting D^i with S^i:

$$\mathcal{S}(R^i, S^i) = \sum_{c \in \mathcal{C}(S^i)} z(c, S^i) \max_{r \in R^i} w(c, r). \tag{11}$$

Similarly, the contingency component can be modified to include connections from synopses to detailed descriptions. For Equation 8, we substitute

	ROUGE-1	ROUGE-2	ROUGE-SU4
ILP-Ext (Banerjee et al., 2015)	0.308	0.112	0.091
ILP-Abs (Banerjee et al., 2015)	0.361	0.158	0.120
Our approach TREM	**0.405**	**0.207**	**0.148**
w/o SR	0.393	0.189	0.144
w/o CC	0.383	0.171	0.132
w/o SR&CC (summarization only)	0.374	0.168	0.129

Table 2: Experimental results on different methods using descriptions. Contingency-based methods generally outperforms summarization-based methods.

D^{i+1} to S^{i+1} where our model only focuses on high-level storyline:

$$\mathcal{M}_r(R^i, S^{i+1}) = \sum_{s \in S^{i+1}} \max_{r \in R^i} \mathbf{r}^\intercal \mathbf{W}^* \mathbf{s}. \quad (12)$$

6 Experimental Setup

We designed our experiments to evaluate whether our TREM model, by considering contingency between adjacent episodes, can achieve better results than summarization techniques. Furthermore, we want to examine how each contingency factor as proposed in Section 4.2 contributes to the system performance. As our model can integrate multiple views, we want to dissect the effects of using different combinations of three views.

6.1 Comparison Models

To answer the research questions presented above, we compare the following methods in our experiments.

– **ILP-Ext** and **ILP-Abs** (Banerjee et al., 2015): This summarizer generates sentences by optimizing the integer linear programming problem in which the information content and linguistic quality are defined. Both extractive and abstractive implementations are used in our experiments.

– **TREM**: Our TREM model proposed in Section 4 extracts sentences that can both summarize the current episode and prompt the next episode with two contingency factors.

– **TREM w/o SR**: The TREM model without the sparse reconstruction factor proposed in Section 4.2.2.

– **TREM w/o CC**: The TREM model without the concept coverage factor proposed in Section 4.2.1.

– **TREM w/o SR&CC**: The summarization-only TREM model without contingency factors. In the rest of the paper, we also call it as **TREM-Summ**.

– **Multi-view TREM**: The augmented TREM model with descriptions, dialogues and synopses as proposed in Section 5. Different views and combinations will be tested in our experiments.

6.2 Methodology

Using *TVRecap*, we measure the quality of generated sentences following the standard metrics in the summarization community, ROUGE (Lin and Hovy, 2003).

For the purpose of evaluation, we defined a development and a test set, by randomly selecting 18 adjacent pairs of episodes from all seasons. These episodes were selected to have at least two recap description sentences. The remaining 70 episodes were only used during the learning process of $\mathbf{W}$. After tuning hyper-parameters on development set, we report the comparison results on the test set.

7 Results and Discussion

7.1 Overall Results

Table 2 shows our experimental results comparing TREM and baseline models using descriptions.

In general, contingency-based methods (TREM, TREM w/o SR and TREM w/o CC) outperform summarization-based methods. Our contingency assumptions are verified as adding CC and SC both improve TREM with summarization component only. Moreover, the best result is achieved by the complete TREM model with both contingency factors. It suggests that these two factors, modeling word-level summarization and sentence-level reconstruction, are complementary.

From the summarization-based methods, we can see that our TREM-Summ gets higher ROUGE scores than two ILP approaches. Additionally, we

Target sentence from next episode	Sentences with highest reconstruction value from current episode
Kate is putting water bottles in a pack.	We see three bottles of water. They go into a room with a body bag on a gurney. Kate is going through clothes, as Claire approaches.
Locke is with his knife case, holding a pencil, sitting by a fire.	Boone is coming up to camp and sees Locke sitting by a fire. Locke throws a knife into the ground, just out of Boone's reach. Boone quickly cuts through the ropes and starts running.
In another part of the temple grounds, Miles and Hurley are playing Tic-Tac-Toe by placing leaves in a grid of sticks on the ground.	John contemplates the fabric swatches he is holding. On the beach, Frank covers Locke's body with a tarp. Helen closes the door and brings the case inside to the kitchen.

Table 3: A case study on sparse reconstruction as proposed in Section 4.2.2. Sentences in the first column are reconstructed by sentences in the second column. The first two examples successfully captures related sentences, while the third example fails.

note that the performance of ILP-Ext is poor. This is because ILP-Ext tends to output short sentences, while ROUGE is a recall-oriented measurement.

Model	Current	Next	R-1	R-2	R-SU4
TREM-Summ	des	-	0.374	0.168	0.129
	syn	-	0.369	0.163	0.121
	dial	-	0.354	0.138	0.115
	des+syn	-	0.384	0.172	0.132
	des+dial	-	0.386	0.168	0.135
TREM	des	des	0.405	0.207	0.148
	des	syn	**0.411**	**0.219**	**0.154**
	des	dial	0.375	0.158	0.127
	des	des+syn	0.409	0.210	**0.154**
	des	des+dial	0.395	0.177	0.142

Table 4: Comparison of views in summarization-only TREM and full TREM with contingency factors. "*des*", "*syn*", and "*dial*" are abbreviated for description, synopses and dialogues.

7.2 Multi-view Comparison

As shown in Table 4, The second study examines the effect of different views in both types of methods using the TREM model. In single-view summarization, TREM-Summ with descriptions outperforms methods based on the other two views. In terms of hybrid of views, only ROUGE-1 is significantly improved, while ROUGE-2 and ROUGE-SU4, which focus more on semantic consistency, have little improvement.

In contingency-based methods, we keep the current episode represented as descriptions which obtain the best performance in single-view summarization, and change the views of the next episode. Comparing the model using descriptions with the one fusing descriptions and synopses, we can see that simply adding views does not guarantee higher ROUGE scores. In both TREM-Summ and full TREM, dialogue is inferior to others. It might be because dialogues contain too many trivial sentences. Synopses, however, are relatively short, but provide key plots to summarize the story, and hence achieve the best ROUGE scores.

7.3 Qualitative Study on Sparse Reconstruction

In this section, we give some examples to illustrate the process of sparse reconstruction. Equation 7 assumes that each descriptive sentence can be reconstructed by a few sentences from the previous episode. Table 3 shows three examples of sentences with their top-3 reconstructive sentences, which are defined by values in the indicator vector $\boldsymbol{\alpha}_j^{i+1}$.

7.4 Limitations and Future Work

TREM restrains the contingency within adjacent episodes. However, storylines sometimes proceed through multiple episodes. In our model, with more connectivity terms $\mathcal{M}(R^i, D^j)$ where $i < j$, we can develop more general system with longer dependencies.

While our model and dataset are appropriate for text recap extraction and algorithm comparison, this task can be further applied to multimedia settings, where visual or acoustic information can be included. Therefore, in future work, we plan to expand our work to broader applications where interconnectivity between consecutive instances is crucial, such as educational lectures, news series and book chapters. Specifically, TREM can be integrated with video description results to get an end-to-end system that produces video recaps.

8 Conclusion

In this paper, we explore a new problem of text recap extraction for TV shows. We propose an unsupervised model that identifies recap segments from multiple views of textual scripts. To facilitate the study of this new research topic, we create a dataset called *TVRecap*, which we test our approach on. From the experimental results, we conclude that contingency-based methods improve summarization-based methods at ROUGE measurements by exploiting plot connection between adjacent episodes.

Acknowledgement

This material is based in part upon work partially supported by the National Science Foundation (IIS-1523162). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Marta Aparício, Paulo Figueiredo, Francisco Raposo, David Martins de Matos, Ricardo Ribeiro, and Luís Marujo. 2016. Summarization of films and documentaries based on subtitles and scripts. *Pattern Recognition Letters* 73:7–12.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *24th International Joint Conference on Artificial Intelligence (IJCAI). Buenos Aires, Argentina: AAAI press*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597* .

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2625–2634.

Paulo Figueiredo, Marta Aparício, David Martins de Matos, and Ricardo Ribeiro. 2015. Generation of multimedia artifacts: An extractive summarization-based approach. *arXiv preprint arXiv:1508.03170* .

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2712–2719.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *AAAI*.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *EACL*. pages 712–721.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726* .

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 71–78.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 912–920.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the*

Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, pages 510–520.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 689–696.

Kathleen McKeown. 2005. Text summarization: News and beyond. In *Proceedings of the Australasian Language Technology Workshop*. pages 4–4.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Amita Misra, Pranav Anand, JEF Tree, and MA Walker. 2015. Using summarization to discover argument facets in online idealogical dialog. In *NAACL HLT*. pages 430–440.

Francisco Raposo, Ricardo Ribeiro, and David Martins de Matos. 2015. On the application of generic summarization algorithms to music. *IEEE Signal Processing Letters* 22(1):26–30.

Anna Rohrbach, Marcus Rohrbach, and Bernt Schiele. 2015. The long-short story of movie description. In *Pattern Recognition*, Springer, pages 209–221.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. 2013. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 433–440.

Shourya Roy, Ragunathan Mariappan, Sandipan Dandapat, Saurabh Srivastava, Sainyam Galhotra, and Balaji Peddamuthu. 2016. Qa rt: A system for real-time holistic quality assurance for contact center dialogues. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

Jitao Sang and Changsheng Xu. 2010. Character-based movie summarization. In *Proceedings of the 18th ACM international conference on Multimedia*. ACM, pages 855–858.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 754–763.

Mark Wasson. 1998. Using leading text for news summaries: Evaluation results and implications for commercial summarization applications. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 1364–1368.

On- and Off-Topic Classification and Semantic Annotation of User-Generated Software Requirements

Markus Dollmann and **Michaela Geierhos**
Heinz Nixdorf Institute
University of Paderborn
Fürstenallee 11, 33102 Paderborn, Germany
{dollmann|geierhos}@hni.upb.de

Abstract

Users prefer natural language software requirements because of their usability and accessibility. When they describe their wishes for software development, they often provide off-topic information. We therefore present REaCT[1], an automated approach for identifying and semantically annotating the on-topic parts of requirement descriptions. It is designed to support requirement engineers in the elicitation process on detecting and analyzing requirements in user-generated content. Since no lexical resources with domain-specific information about requirements are available, we created a corpus of requirements written in controlled language by instructed users and uncontrolled language by uninstructed users. We annotated these requirements regarding predicate-argument structures, conditions, priorities, motivations and semantic roles and used this information to train classifiers for information extraction purposes. REaCT achieves an accuracy of 92% for the on- and off-topic classification task and an F_1-measure of 72% for the semantic annotation.

1 Introduction

"Requirements are what the software product, or hardware product, or service, or whatever you intend to build, is meant to do and to be" (Robertson and Robertson, 2012). This intuitive description of requirements has one disadvantage. It is as vague as a requirement that is written by an untrained user. More generally, functional requirements define what a product, system or process, or a part of it is meant to do (Robertson and Robertson, 2012; Vlas and Robinson, 2011). Due to its expressiveness, natural language (NL) became a popular medium of communication between users and developers during the requirement elicitation process (de Almeida Ferreira and da Silva, 2012; Mich et al., 2004). Especially in large ICT projects, requirements, wishes, and ideas of up to thousands of different users have to be grasped (Castro-Herrera et al., 2009). For this purpose, requirement engineers collect their data, look for project-relevant concepts and summarize the identified technical features. However, this hand-crafted aggregation and translation process from NL to formal specifications is error-prone (Goldin and Berry, 1994). Since people are getting tired and unfocused during this monotonous work, the risk of information loss increases. Hence, this process should be automated as far as possible to support requirement engineers.

In this paper, we introduce our approach to identify and annotate requirements in user-generated content. We acquired feature requests for open source software from SourceForge[2], specified by (potential) users of the software. We divided these requests into off-topic information and (on-topic) requirements to train a binary text classifier. This allows an automated identification of new requirements in user-generated content. In addition, we collected requirements in controlled language from the NFR corpus[3] and from web pages with user-story explanations. We annotated the semantically rele-

[1] Requirements Extraction and Classification Tool

[2] https://sourceforge.net

[3] http://openscience.us/repo/requirements/other-requirements/nfr

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1807–1816,
Austin, Texas, November 1-5, 2016.

vant parts of the acquired requirements for information extraction purposes. This will support requirements engineers on requirement analysis and enables a further processing such as disambiguation or the resolution of incomplete expressions.

This paper is structured as follows: In Section 2, we discuss the notion of requirements. Then we provide an overview of previous work (Section 3), before we introduce lexical resources necessary for our method (Section 4). The approach itself is presented in Section 5 before it is evaluated in Section 6. Finally, we conclude this work in Section 7.

2 The Nature of Requirements

Requirement engineers and software developers have to meet users' wishes in order to create new software products. Such descriptions of software functionalities can be expressed in different ways: For example, by using controlled languages or formal methods, clarity and completeness can be achieved. But non-experts can hardly apply them and therefore do not belong to the user group. For this reason, users are encouraged to express their individual requirements for the desired software application in NL in order to improve user acceptance and satisfaction (Firesmith, 2005). In general, software requirements are expressed through active verbs such as *"to calculate"* or *"to publish"* (Robertson and Robertson, 2012). In this work, we distinguish requirements expressed in NL between controlled and uncontrolled language.

A controlled language is a subset of NL, which is characterized by a restricted grammar and/or limited vocabulary (Yue et al., 2010). Requirements in controlled language do not suffer from ambiguity, redundancy and complexity (Yue et al., 2010). That is why these recommendations lead to a desirable input for text processing. Robertson and Robertson (2012) therefore recommend specifying each requirement in a single sentence with one verb. Furthermore, they suggest the following start of record *"The [system/product/process] shall ..."*, which focuses on the functionality and keeps the active form of a sentence. An example therefore is *"The system shall display the Events in a graph by time."* Another type of controlled requirements are user stories. They follow the form *"As a [role], I want [something] so that [benefit]"* and describe software functionalities from the user's perspective (Cohn, 2004). Compared to the previous ones, they do not focus on the technical implementation but concentrate on the goals and resulting benefits. An example therefore is *"As a Creator, I want to upload a video from my local machine so that any users can view it."*

We also consider uncontrolled language in this work because requirements are usually specified by users that have not been instructed for any type of formulation. Requirements in uncontrolled language do not stick to grammar and/or orthographic rules and may contain abbreviations, acronyms or emoticons. There is no restriction how to express oneself. An example therefore is *"Hello, I would like to suggest the implementation of expiration date for the master password :)"*.

In the following, the word "requirement" is used for a described functionality. We assume that its textualization is written within a single English sentence. Requirements are specified in documents like the *Software Requirements Specification* (SRS). We refer to SRS and other forms (e.g. e-mails, memos from workshops, transcripts of interviews or entries in bug-trackers) as requirement documentations.

3 Previous Work

It is quite common that requirement engineers elicit requirements together with users in interviews, group meetings, or by using questionnaires (Mich, 1996). Researchers developed (semi-) automated and collaborative approaches to support requirement engineers in this process (Ankori and Ankori, 2005; Castro-Herrera et al., 2009). Besides the elicitation in interaction with the users, an identification of requirements from existing sources is possible. For example, John and Dörr (2003) used documentations from related products to derive requirements for a new product. Vlas and Robinson (2011) used unstructured, informal, NL feature requests from the platform SourceForge to collect requirements for open source software. They presented a rule-based method to identify and classify requirements according to the quality criteria of the *McCall's Quality Model* (McCall, 1977). Analogous to their work, we want to automatically detect requirements in user-generated content. While they applied a rule-based

method, we plan to identify requirements in user-generated content with a machine learning approach. Since those approaches automatically identify patterns for this classification task, we expect a higher recall and more reliable results.

Goldin and Berry (1994) identified so-called abstractions (i.e. relevant terms and concepts related to a product) of elicited requirements for a better comprehension of the domain and its restrictions. Their tool *AbstFinder* is based on the idea that the significance of terms and concepts is related to the number of mentions in the text. However, in some cases, there is only a weak correlation between the term frequencies and their relevance in documents. This problem can be reduced by a statistical corpus analysis, when the actual term frequency is similar to the expected (Sawyer et al., 2002; Gacitua et al., 2011). This approach eliminates corpus specific stopwords and misleading frequent terms. In our work, we intent to perform a content analysis of the previously detected requirements. However, instead of only identifying significant terms and concepts, we capture the semantically relevant parts of requirements such as conditions, motivations, roles or actions (cf. Figure 1).

In addition to the identification of abstractions, there are methods to transform NL requirements into graphical models (e.g. in *Unified Modeling Language*) (Harmain and Gaizauskas, 2003; Ambriola and Gervasi, 2006; Körner and Gelhausen, 2008). A systematic literature review, done by Yue et al. (2010), aims at the modeling of requirements by comparing transformation techniques in such models. Unlike those techniques, we aim to keep the expressive aspect of the original textual requirements and semantically annotate them for filtering purposes. These results can be further used for different NLP tasks such as disambiguation, resolution of vagueness or the compensation of underspecification.

The semantic annotation task of this work is similar to semantic role labeling (SLR). According to Jurafsky and Martin (2015), the goal of SLR is understanding events and their participants, especially being able to answer the question *who did what to whom* (and perhaps also *when and where*). In this work, we seek to adapt this goal to the requirements domain, where we want to answer the question *what actions should be done by which component* (and perhaps also *who wants to perform that action*, *are there any conditions*, *what is the motivation for performing this action* and *is there a priority assigned to the requirement*).

4 Gathering and Annotation of Controlled and Uncontrolled Requirements

There are benchmarks comparing automated methods for requirement engineering (Tichy et al., 2015). However, none of the published datasets is sufficient to train a text classifier, since annotated information is missing. For our purposes, we need a data set with annotated predicate-argument structures, conditions, priorities, motivations and semantic roles. We therefore created a semantically annotated corpus by using the categories shown in Figure 1, which represent all information bits of a requirement. Since the approach should be able to distinguish between (on-topic) requirements and off-topic comments, we acquired software domain-specific off-topic sentences, too.

Therefore, we acquired requirements in controlled language from the system's and the user's perspective. While requirements from the system's perspective are describing technical software functionalities, the requirements from the user's perspective express wishes for software, to fulfill user needs. For instance, the NFR corpus[4] covers the system's perspective of controlled requirements specifications. It consists of 255 functional and 370 non-functional requirements whereof we used the functional subset to cover the system's perspective. Since we could not identify any requirement corpus that describes a software at user's request, we acquired 304 user stories from websites and books that describe how to write user stories.

However, these requirements in controlled language have not the same characteristics as uncontrolled requirements descriptions. For the acquisition of uncontrolled requirements, we adapted the idea of Vlas and Robinson (2011) that is based on feature requests gathered from the open-source software platform SourceForge[5]. These feature requests

[4] https://terapromise.csc.ncsu.edu/repo/requirements/nfr/nfr.arff

[5] https://sourceforge.net

are created by users that have not been instructed for any type of formulation. Since these requests do not only contain requirements, we split them into sentences and manually classified them in requirements and off-topic information. Here, we consider social communication, descriptions of workflows, descriptions of existing software features, feedback, salutations, or greetings as off-topic information. In total, we gathered 200 uncontrolled on-topic sentences (i.e. requirements) and 492 off-topic ones.

Then we analyzed the acquired requirements in order to identify the different possible semantic categories to annotate their relevant content in our requirements corpus (cf. Figure 1):

- **component**
  - refinement of component
- **action**
  - argument of action
- condition
- priority
- motivation
- role
- **object**
  - refinement of object
- **sub-action**
  - argument of sub-action
- sub-priority
- sub-role
- **sub-object**
  - refinement of sub-object

Figure 1: Semantic categories in our software requirements corpus used for annotation purposes

The categories *component* or *role*, *action* and *object* are usually represented by subject, predicate and object of a sentence. In general, a description refers to a *component*, either to a product or system itself or to a part of the product/system. *Actions* describe what a component should accomplish and affect. Actions have an effect on *Objects*. The authors of the requirements can refine the description of components and objects, which is covered by the categories *refinement of component* and *refinement of object*. For each action, users can set a certain *priority*, describe their *motivation* for a specific functionality, state *conditions*, and/or even define some semantic *roles*. Apart from the component and the object, additional arguments of the action (predicate of a sentence) are annotated with *argument of action*. In some cases, requirements contain sub-requirements in subordinate clauses. The annotators took this into account when using the predefined sub-categories. An example of an annotated requirement is shown in Figure 2.

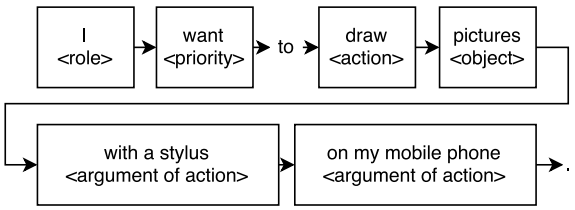


Figure 2: Annotated requirement sample

Two annotators independently labeled the categories in the requirements. We define one of the annotation set as gold standard and the other as candidate set. We will use the gold standard for training and testing purposes in Section 5 and 6 and the candidate set for calculating an inter-annotator agreement. In total, our gold standard consists of 3,996 labeled elements (i.e. clauses, phrases, and even modality). The frequency distribution is shown in Table 1.

Semantic Category	CR	UR	Total
component	241	84	325
refinement of component	6	16	22
action	526	204	730
argument of action	180	104	284
condition	94	39	133
priority	488	209	697
motivation	33	19	52
role	406	42	448
object	540	195	735
refinement of object	155	48	203
sub-action	76	40	116
argument of sub-action	27	14	41
sub-priority	22	16	38
sub-role	22	11	33
sub-object	78	37	115
refinement of sub-object	16	8	24
Total	2,910	1,086	3,996

Table 1: Number of annotated elements per category in our gold standard (CR=controlled requirements, UR=uncontrolled requirements)

The inter-annotator agreement in multi-token annotations is commonly evaluated by using F_1-score (Chinchor, 1998). The two annotators achieve an agreement of 80%, whereby the comparison was invoked from the gold standard.

Many information extraction tasks use the IOB encoding[6] for annotation purposes. When using the IOB encoding, the first token of an element is split into its head (first token) and its tail (rest of the element). That way, its boundaries are labeled with B (begin) and I (inside). This allows separating successive elements of the same category. Thus, we use the IOB encoding during the annotation step. However, we want to discuss a drawback of this notation: When applying text classification approaches in information extraction tasks with IOB encoding, the number of classes reduplicates and this reduces the amount of training data per class. During our annotation process, successive elements of the same semantic category only occurred in the case of *argument of the action* and *argument of the sub-action*. When we disregard the IOB encoding, we can easily split up (sub-)actions by keywords such as *"in"*, *"by"*, *"from"*, *"as"*, *"on"*, *"to"*, *"into"*, *"for"*, and *"through"*. So if we use IO encoding, it can be easily transformed to the IOB encoding. The only difference between IOB and IO encoding is that it does not distinguish between the head and tail of an element and therefore does not double the number of classes.

5 REaCT – A Two-Stage Approach

Requirement documentations are the input of our system. Figure 3 illustrates the two-stage approach divided in two separate classification tasks. First, we apply an on-/off-topic classification to decide whether a sentence is a requirement or irrelevant for the further processing (cf. Section 5.1). Then, the previously identified requirements were automatically annotated (Section 5.2). As a result, we get filtered and semantically annotated requirements in XML or JSON.

The models for on-/off-topic classification and semantic annotation are trained on the gathered requirements (cf. Section 4). We split up the gold standard on sentence level in a ratio of 4:1 in a training set of 607 requirements and test set of 152 requirements. Furthermore, we used 10-fold cross validation on the training set for algorithm configuration and feature engineering (cf. Section 5.1 and Section 5.2). Finally, our approach is evaluated on the test set (cf. Section 6).

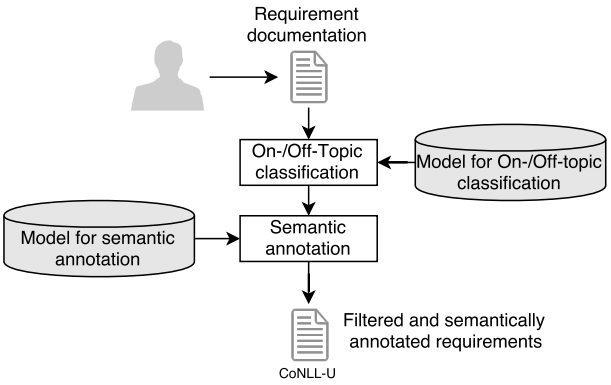


Figure 3: Processing pipeline of the two-stage approach

5.1 On-/Off-Topic Classification Task

User requirement documentations often contain off-topic information. Therefore, we present a binary text classification approach that distinguishes between requirements and off-topic content. Thus, we trained different classification algorithms and tested them using various features and parameter settings. We compared the results to select the best algorithm together with its best-suited parameter values and features.

5.1.1 Features

To differentiate requirements from off-topic content, the sentences will be transformed in numerical feature vectors using a bag-of-words approach with different settings[7]. The features for the transformation are listed along with their possible parameter settings in Table 2. We can choose whether the feature should be taken from word or character n-grams (a.1). For both versions, the unit can range between $[n, m]$ (a.2), which can be specified by parameters. Here, we consider all combinations of $n = [1, 5]$ and $m = [1, 5]$ (where $m \geq n$). If the feature should be build from word n-grams, stopword detection is possible (a.3). Additionally, terms can be ignored that reach a document frequency below or above a given

[6] I (inside), O (outside) or B (begin) of an element

[7] Parameters; to be chosen during algorithm configuration

threshold (e.g. domain-specific stopwords) (a.4 and a.5). Another threshold can be specified to only consider the top features ordered by term frequency (a.6). Besides, it is possible to re-weight the units in the bag-of-words model in relation to the inverse document frequency (IDF) (a.7). Moreover, the frequency vector can be reduced to binary values (a.8), so that the bag-of-words model only contains information about the term occurrence but not about its calculated frequency. We also consider the length of a sentence as feature (b). Furthermore, the frequency of the part-of-speech (POS) tags (c) and the dependencies between the tokens (d) can be added to the feature vector[8]. These two features are optional (c.1 and d.1). This set of features covers the domain-specific characteristics and should enable the identification of the requirements.

#	Feature/Parameter	Possible Values
a	Bag of words	
a.1	analyzer	word, char
a.2	ngram_range	(1,1),(1,2), ...,(5,5)
a.3	stop_words	True, False
a.4	max_df	[0,8,1,0]
a.5	min_df	[0.0,0.5]
a.6	max_features	int or None
a.7	use_idf	True, False
a.8	binary	True, False
b	Length of the sentence	
c	Dependencies	
c.1	use_dep	True, False
d	Part of speech	
d.1	use_pos	True, False

Table 2: Features for on-/off-topic classification together with their corresponding parameters

5.1.2 Selected Algorithms

We selected the following algorithms from the *scikit-learn* library[9] for binary classification: decision tree (`DecisionTreeClassifier`), Naive Bayes (`BernoulliNB` and `MultionmialNB`), support vector machine (`SVC` and `NuSVC`) as well as ensemble methods (`BaggingClassifier`, `RandomForestClassifier`, `ExtraTreeClassifier` and `AdaBoostClassifier`). Finally, after evaluating these algorithms, we chose the best one for the classification task (cf. Section 6).

5.2 Semantic Annotation Task

For each identified requirement, the approach should annotate the semantic components (cf. Figure 1). Here, we use text classification techniques on token level for information extraction purposes. The benefit is that these techniques can automatically learn rules to classify data from the annotated elements (cf. Section 4). Each token will be assigned to one of the semantic categories presented in Figure 1 or the additional class *O* (outside according IOB notation).

We decided in favor of IO encoding during classification to reduce the drawback described in Section 4. We finally convert the classification results into the IOB encoding by labeling the head of each element as begin and the tail as inside. By using the keywords listed in Section 4 as separators, we further distinguish the beginning and the inner parts of arguments.

5.2.1 Features

In the second classification step, we had to adapt the features to token level. The goal of feature engineering is to capture the characteristics of the tokens embedded in their surrounding context. We divided the features in four groups: orthographic and semantic features of the token, contextual features, and traceable classification results.

Orthographic features of a token are its graphematic representation (a) and additional flags that decide if a token contains a number (b), is capitalized (c), or is somehow uppercased (d) (cf. Table 3). For the graphematic representation, we can choose between the token or the lemma (a.1). Another orthographic feature provides information about the length of the token (e). Furthermore, we can use the pre- and suffix characters of the token as features (f and g). Their lengths are configurable (f.1 and g.1).

[8] We use spaCy (`https://spacy.io`) for POS tagging and dependency parsing

[9] `http://scikit-learn.org`

#	Feature/Parameter	Possible Values
a	Graphematic representation	
a.1	use_lemma	True, False
b	Token contains a number	
c	Token is capitalized	
d	Token is somehow uppercased	
e	Length of the token	
f	Prefix of the token	
f.1	length_prefix	[0,5]
g	Suffix of the token	
g.1	length_suffix	[0,5]

Table 3: Orthographic features for semantic annotation

Furthermore, we consider the relevance (h), the POS tag (i) and the WordNet ID of a token (j) as its semantic features (cf. Table 4). By checking the stopword status of a token, we can decide about its relevance. Besides, the POS tag of each token is used as feature. When applying the POS information, we can choose between the *Universal Tag Set*[10] (consisting of 17 POS tags) and the *Penn Treebank Tag Set*[11] (including of 36 POS tags) (i.1). Another boolean feature tells us whether the token appears in *WordNet*[12]. We use this feature as indicator for component or object identification.

#	Feature/Parameter	Possible Values
h	Relevance	
i	Part-of-speech tag	
i.1	extended_tagset	True, False
j	WordNet ID	

Table 4: Semantic features for semantic annotation

As contextual features, we use sentence length (k), index of the token in the sentence (l), as well as the tagging and dependency parsing information of the surrounding tokens (m, n and o) (cf. Table 5). Thus, the POS tags sequences of the n previous and the next m token are considered, where n and m are defined during algorithm configuration (l.1 and n.1). Moreover, it can be specified if each POS tag should be stored as a single feature or should be concatenated (e.g. NOUN+VERB+NOUN) (l.2 and n.2).

#	Feature/Parameter	Possible Values
k	Sentence length	
l	Index of the token	
m	Previous part-of-speech tags	
l.1	n_pos_prev	[0,15]
l.2	conc_prev_pos	True, False
n	Subsequent part-of-speech tags	
n.1	n_pos_succ	[0,15]
n.2	conc_succ_pos	True, False
o	Dependencies	

Table 5: Contextual features for semantic annotation

The classification task is carried out from left to right in the sentence. This enables the consideration of previous classification results (cf. Table 6). We implemented two slightly different variants that can be combined on demand: Firstly, we can define a fixed number of previous classification results as independent or concatenated features (i.e. a sliding window (p)). Secondly, the number of token already assigned to a particular class may be a valuable information (q). This is especially of interest for the hierarchical structure of the categories: For instance, a *sub-object* should only occur if an *object* has already been identified. These two features are optional (p.1 and q.1). The size of the sliding window will be specified during algorithm configuration (p.2).

#	Feature/Parameter	Possible Values
p	Sliding windows	
p.1	conc_prev_labels	True, False
p.2	window_size	[0,10]
q	Number of previous labels per category	
q.1	use_prev_labels	True, False

Table 6: Traceable classification results for semantic annotation

[10] http://universaldependencies.org/u/pos/

[11] http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[12] https://wordnet.princeton.edu

5.2.2 Selected Algorithms

In addition to the classifiers we already used in the on-/off-topic classification task, we considered three sequential learning algorithms: conditional random fields (`FrankWolfeSSVM`) from the *PyStruct*-library[13], multinomial hidden markov model (`MultinomialHMM`) as well as structured perceptron from the *seqlearn*-library[14]. We could not estimate feasible parameter settings for the `NuSVC` classifier, so that this classifier was ignored. We chose the algorithm with the best results on the test set for annotating the requirements (cf. Section 6).

6 Evaluation

As mentioned in Section 5, the data was separated in a ratio of 4:1 in a training and a test set. We trained all classifiers on the training set with their defined settings from the automated algorithm configuration. Subsequently, we evaluated these classifiers on the test set. Our results are shown in Table 7 that lists the accuracy for the best classifier per algorithm family of the on-/off-topic classification task. The `ExtraTreeClassifier` performs best on the test data with an accuracy of 92%. The accuracy was calculated with the following formula:

$$accuracy = \frac{\#true\ positives + \#true\ negatives}{\#classified\ requirements}$$

The `ExtraTreeClassifier` is an implementation of *Extremely Randomized Trees* (Geurts et al., 2006). We achieved the best result when using character n-grams as features in the model with a fixed length of 4. Thereby, we considered term occurrence instead of term frequency and IDF. Before creating the bag-of-words model, the approach removes stopwords. Furthermore, the frequency of the POS tags and their dependencies are used as features. In total, the `ExtraTreeClassifier` used 167 estimators based on entropy in the ensemble (algorithm-specific parameters).

[13] `https://pystruct.github.io`

[14] `https://github.com/larsmans/seqlearn`

Classifier	Accuracy
AdaBoostClassifier	0.87
ExtraTreeClassifier	0.92
MultinomialNB	0.89
NuSVC	0.90

Table 7: Accuracy of best classifiers per algorithm family in the on-/off-topic classification task after algorithm configuration

Table 8 shows the values for precision, recall, and F_1 of the `ExtraTreeClassifier`. In brief, the introduced approach detects requirements in user-generated content with an average F_1-score of 91%.

Class	Precision	Recall	F_1
off-topic info	0.94	0.85	0.89
requirements	0.89	0.96	0.93
Avg.	0.92	0.91	0.91

Table 8: Evaluation results for the on-/off-topic classification with the `ExtraTreeClassifier`

Table 9 provides an overview of the results of the semantic annotation task. To determine the F_1-score, the agreement of the predicted and the a priori given annotations is necessary to count an element as true positive.

Again, the `ExtraTreeClassifier` achieves the best F_1-score of 72%. We gained the best results by using 171 randomized decision trees based on entropy (algorithm-specific parameters). As features, we took the POS tags from *Universal Tag Set* for the twelve previous and the three following tokens. Traceable classification results are taken into account by a sliding window of size 1. Besides, we validate if a class label has already been assigned. For each considered token, the four prefix and the two suffix characters as well as the graphematic representation of the token are applied as features.

The sequential learning algorithms (`FrankWolfeSSVM`, `MultinomialHMM` and `StructuredPerceptron`) perform worse than the other classifiers. We assume that this is due to the small amount of available training data. However, the methods depending on decision trees, especially the ensemble methods (`RandomForestClassifier`, `Bagging-`

Classifier and ExtraTreeClassifier), perform significantly better.

Classifier	F_1
AdaBoostClassifier	0.33
ExtraTreeClassifier	0.72
FrankWolfeSSVM	0.50
MultinomialNB	0.64
SVC	0.70

Table 9: F_1-scores of best classifiers per algorithm family in the semantic annotation task after algorithm configuration

In Table 10, we provide detailed results achieved with the ExtraTreeClassifier for the different semantic categories. The recognition of main aspects (component, action and object) reached F_1-scores of 73%, 80% and 68%. The semantic categories, that have only a few training examples, are more error-prone (e.g. sub-action or sub-object).

Semantic Category	Precision	Recall	F_1
component	0.71	0.75	0.73
ref. of component	0.17	0.14	0.15
action	0.78	0.82	0.80
arg. of action	0.49	0.62	0.54
condition	0.88	0.61	0.72
priority	0.96	0.96	0.96
motivation	0.67	0.29	0.40
role	0.93	0.86	0.89
object	0.63	0.74	0.68
ref. of object	0.69	0.51	0.59
sub-action	0.46	0.44	0.45
arg. of sub-action	0.33	0.29	0.31
sub-priority	0.44	0.57	0.50
sub-role	0.40	0.80	0.53
sub-object	0.35	0.33	0.34
ref. of sub-object	0.67	0.33	0.44
Avg.	0.72	0.73	0.73

Table 10: Evaluation results for the semantic annotation with the ExtraTreeClassifier

7 Conclusion and Future Work

Requirement engineers and software developers have to meet users' wishes to create new software products. The goal of this work was to develop a system that can identify and analyze requirements expressed in natural language. These are written by users unlimited in their way of expression. Our system REaCT achieves an accuracy of 92% in distinguishing between on- and off-topic information in the user-generated requirement descriptions. The text classification approach for semantic annotation reaches an F_1-score of 72% – a satisfying result compared to the inter-annotator agreement of 80%. One possibility to improve the quality of the semantic annotation is to expand the training set. Especially the sequential learning techniques need more training data. Besides, this would have a positive impact on those semantic categories that only contain a small number of annotated elements.

Developers and requirement engineers can facilely identify requirements written by users for products in different scenarios by applying our approach. Moreover, the semantic annotations are useful for further NLP tasks. User-generated software requirements adhere to the same quality standards as software requirements that are collected and revised by experts: They should be complete, unambiguous and consistent (Hsia et al., 1993). Since there was no assistant system to check the quality for many years (Hussain et al., 2007) we plan to extend the provided system in order to provide some quality analysis of the extracted information. We have already developed concepts to generate suggestions for non-experts, how to complete or clarify their requirement descriptions (Geierhos et al., 2015). Based on these insights, we want to implement a system for the resolution of vagueness and incompleteness of NL requirements.

Acknowledgments

Special thanks to our colleagues Frederik S. Bäumer and David Kopecki for their support during the semantic annotation of the requirements. This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre „On-The-Fly Computing“ (SFB 901).

References

Vincenzo Ambriola and Vincenzo Gervasi. 2006. On the Systematic Analysis of Natural Language Requirements with CIRCE. *Automated Software Engineering*, 13(1):107–167.

Ronit Ankori and Ronit Ankori. 2005. Automatic requirements elicitation in agile processes. In *Proceedings of the 2005 IEEE International Conference on Software - Science, Technology and Engineering*, pages 101–109. IEEE.

Carlos Castro-Herrera, Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. 2009. A recommender system for requirements elicitation in large-scale software projects. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1419–1426. ACM.

Nancy A. Chinchor, editor. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7) Named Entity Task Definition*, Fairfax, VA.

Mike Cohn. 2004. *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Redwood City, CA, USA.

David de Almeida Ferreira and Alberto Rodrigues da Silva. 2012. RSLingo: An information extraction approach toward formal requirements specifications. In *Model-Driven Requirements Engineering Workshop*, pages 39–48. IEEE.

Donald G. Firesmith. 2005. Are Your Requirements Complete? *Journal of Object Technology*, 4(2):27–43, February.

Ricardo Gacitua, Pete Sawyer, and Vincenzo Gervasi. 2011. Relevance-based abstraction identification: technique and evaluation. *Requirements Engineering*, 16(3):251–265.

Michaela Geierhos, Sabine Schulze, and Frederik Simon Bäumer. 2015. What did you mean? Facing the Challenges of User-generated Software Requirements. In *Proceedings of the 7th International Conference on Agents and Artificial Intelligence*, pages 277–283, 10 - 12 January. Lisbon. ISBN: 978-989-758-073-4.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Leah Goldin and Daniel M. Berry. 1994. AbstFinder, A Prototype Abstraction Finder for Natural Language Text for Use in Requirements Elicitation: Design, Methodology, and Evaluation. *Automated Software Engineering*, 4(4):375–412.

H.M. Harmain and R. Gaizauskas. 2003. CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. *IEEE International Conference on Software - Science, Technology & Engineering*, 10(2):157–181.

Pei Hsia, Alan Davis, and David Kung. 1993. Status Report: Requirements Engineering. *IEEE Software*, 10(6):75–79, November.

Ishrar Hussain, Olga Ormandjieva, and Leila Kosseim. 2007. Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier. In *Proceedings of the 7th International Conference on Quality Software*, QSIC '07, pages 209–218. IEEE.

Isabel John and Jörg Dörr. 2003. Elicitation of Requirements from User Documentation. In *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation of Software Quality*, pages 17–26. Springer.

Daniel Jurafsky and James H Martin. 2015. Semantic role labeling. In *Speech and Language Processing*. 3rd ed. draft edition.

Sven J. Körner and Tom Gelhausen. 2008. Improving Automatic Model Creation using Ontologies. In *Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering*, pages 691–696. Knowledge Systems Institute.

Jim McCall. 1977. McCall's Quality Model. `http://www.sqa.net/softwarequalityattributes.html`.

Luisa Mich, Mariangela Franch, and Pier Luigi Novi Inverardi. 2004. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9(2):151–151.

Luisa Mich. 1996. NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. *Natural Language Engineering*, 2:161–187.

James Robertson and Suzanne Robertson. 2012. *Mastering the Requirements Process*. Getting Requirements Right. Addison-Wesley Publishing, New York, NY, USA.

Peter Sawyer, Paul Rayson, and Roger Garside. 2002. REVERE: support for requirements synthesis from documents. *Information Systems Frontiers*, 4(3):343–353.

Walter F. Tichy, Mathias Landhäußer, and Sven J. Körner. 2015. nlrpBENCH: A Benchmark for Natural Language Requirements Processing. In *Multikonferenz Software Engineering & Management 2015*. GI.

Radu Vlas and William N. Robinson. 2011. A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects. In *Proceedings of the 44th Hawaii International Conference on System Sciences*, pages 1–10. IEEE.

Tao Yue, Lionel C. Briand, and Yvan Labiche. 2010. A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering*, 16(2):75–99.

Deceptive Review Spam Detection via Exploiting Task Relatedness and Unlabeled Data

Zhen Hai[†] **Peilin Zhao**[‡] **Peng Cheng**[§] **Peng Yang**[*] **Xiao-Li Li** [†] **Guangxia Li**[¶]
[†]Institute for Infocomm Research, A*STAR, Singapore, {*haiz,xlli*}*@i2r.a-star.edu.sg*
[‡]Ant Financial, Hangzhou, China, *peilin.zpl@alipay.com*
[§]SCSE, Nanyang Technological University, Singapore, *pcheng1@ntu.edu.sg*
[*]Tencent AI Lab, Shenzhen, China, *henryppyang@tencent.com*
[¶]SCST, Xidian University, Xi'an, China, *gxli@xidian.edu.cn*

Abstract

Existing work on detecting deceptive reviews primarily focuses on feature engineering and applies off-the-shelf supervised classification algorithms to the problem. Then, one real challenge would be to manually recognize plentiful ground truth spam review data for model building, which is rather difficult and often requires domain expertise in practice. In this paper, we propose to exploit the relatedness of multiple review spam detection tasks and readily available unlabeled data to address the scarcity of labeled opinion spam data. We first develop a multi-task learning method based on logistic regression (MTL-LR), which can boost the learning for a task by sharing the knowledge contained in the training signals of other related tasks. To leverage the unlabeled data, we introduce a graph Laplacian regularizer into each base model. We then propose a novel semi-supervised multi-task learning method via Laplacian regularized logistic regression (SMTL-LLR) to further improve the review spam detection performance. We also develop a stochastic alternating method to cope with the optimization for SMTL-LLR. Experimental results on real-world review data demonstrate the benefit of SMTL-LLR over several well-established baseline methods.

1 Introduction

Nowadays, more and more individuals and organizations have become accustomed to consulting user-generated reviews before making purchases or online bookings. Considering great commercial benefits, merchants, however, have tried to hire people to write undeserving positive reviews to promote their own products or services, and meanwhile to post malicious negative reviews to defame those of their competitors. The fictitious reviews and opinions, which are deliberately created in order to promote or demote targeted entities, are known as *deceptive opinion spam* (Jindal and Liu, 2008; Ott et al., 2011).

By formulating deceptive opinion spam detection as a classification problem, existing work primarily focuses on extracting different types of features and applies off-the-shelf supervised classification algorithms to the problem (Jindal and Liu, 2008; Ott et al., 2011; Feng et al., 2012; Chen and Chen, 2015). Then, one weakness of previous work lies in the demand of manually recognizing a large amount of ground truth review spam data for model training. Unlike other forms of spamming activities, such as email or web spam, deceptive opinion spam, which has been deliberately written to sound authentic, is more difficult to be recognized by manual read. In an experiment, three undergraduate students were (randomly) invited to identify spam reviews from nonspam ones in hotel domain. As shown in Table 1, their average accuracy is merely 57.3% (Ott et al., 2011). Then, given a limited set of labeled review data for a domain, e.g., hotel, it is almost impossible to build a robust classification model for detecting deceptive spam reviews in reality.

In this work, we deal with the problem of detecting a textual review as *spam* or not, i.e., *nonspam*. We consider each deceptive review spam detection problem within each domain, e.g., detecting

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1817–1826,
Austin, Texas, November 1-5, 2016.

	Judge-1	Judge-2	Judge-3
Accuracy	61.9%	56.9%	53.1%
F-spam	48.7%	30.3%	43.6%
F-nonspam	69.7%	68.8%	59.9%

Table 1: Performance of human judges for review spam detection in hotel domain (Ott et al., 2011), where F-spam/F-nonspam means F-score for spam/nonspam label.

spam hotel/restuarnt reviews from hotel/restaurnat domain, to be a different task. Previous studies have empirically shown that learning multiple related tasks simultaneously can significantly improve performance relative to learning each task independently, especially when only a few labeled data per task are available (Caruana, 1997; Bakker and Heskes, 2003; Argyriou et al., 2006). Thus, given the limited labeled review data for each domain, we formulate the review spam detection tasks for multiple domains, e.g., hotel, restaurant, and so on, as a multi-task learning problem.

We develop a multi-task learning method via logistic regression (MTL-LR) to address the problem. One key advantage of the method is that it allows to boost the learning for one review spam detection task by leveraging the knowledge contained in the training signals of other related tasks. Then, there is often a large quantity of review data freely available online. In order to leverage the unlabeled data, we introduce a graph Laplacian regularizer into each base logistic regression model. We extend MTL-LR, and propose a novel semi-supervised multi-task learning model via Laplacian regularized logistic regression (SMTL-LLR) to further boost the review spam detection performance under the multi-task learning setting. Moreover, to cope with the optimization problem for SMTL-LLR, we also develop a stochastic alternating optimization method, which is computationally efficient.

To the best of our knowledge, this is the first work that generalizes opinion spam detection from independent single-task learning to symmetric multi-task learning setting. By symmetric, we mean that the setting seeks to improve the performance of all learning tasks simultaneously. In this sense, it is different from transfer learning (Pan and Yang, 2010), where the objective is to improve the performance of a target task using information from source tasks. Under this new setting, we can exploit the commonality shared by related review spam detection tasks as well as readily available unlabeled data, and then alleviate the scarcity of labeled spam review data. Experimental results on real-world review data demonstrate the superiority of SMTL-LLR over several representative baseline methods.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 introduces the proposed methods and stochastic alternating optimization algorithm. Then, in Section 4, we present the experimental results in detail, and conclude this paper in Section 5.

2 Related Work

Previous work typically formulates deceptive opinion spam detection as a classification problem, and then presents different types of features to train supervised classification algorithms for the problem. Jindal and Liu (2008) first studied opinion spam detection problem. They built the ground truth review data set by treating the duplicate reviews in a given corpus as spam reviews and the rest as nonspam reviews. They presented review, product, and reviewer related features, and then trained logistic regression (LR) model on the features for finding fake review spam. Ott et al. (2011) created the ground truth review data via a crowd-sourcing service called Amazon Mechanical Turk[1]. They presented three different types of features for opinion spam detection, i.e., genre identification features, psycholinguistic deception features, and standard n-gram text features. They found that the supervised support vector machines (SVM) trained on the textual n-gram features can achieve good performance. Feng et al. (2012) presented syntactic stylometry features and trained SVM model for deception detection, while Chen and Chen (2015) built the SVM classifier on a diversity of features, such as content and thread features, for opinion spam detection in web forum. In addition, Li et al. (2014) employed a feature-based additive model to explore the general rule for deceptive opinion spam detection. Generally, in order to build robust supervised review spam detection models, we have to manually recognize large-scale ground truth spam data. But this could be very ex-

[1]https://www.mturk.com

pensive, and often requires domain expertise.

Though a large amount of unlabeled review data are freely available online, very limited work has been done on developing semi-supervised methods for review spam detection. Li et al. (2011) used a two-view co-training method (Blum and Mitchell, 1998) for semi-supervised learning to identify fake review spam. One limitation of the work is that it needs additional reviewer information when building model. Given a corpus of textual reviews, the reviewer related view may not be always available in reality. Moreover, the co-training method is not intrinsically geared to learning from the unlabeled review data, instead, simply makes use of the unlabeled reviews within a fully supervised learning framework, negating the semi-supervised learning benefit. For some particular scenarios, the available training data could be only a partially labeled set of positive examples, e.g., spam reviews, and a large set of unlabeled reviews. Positive unlabeled learning (PU) (De Comite et al., 1999; Liu et al., 2002) may be then used for deceptive review spam detection (Hernandez et al., 2013). However, this clearly contrasts with our problem, where our training data contains a complete labeled set of positive (spam) and negative (nonspam) reviews besides the unlabeled set of review data.

In addition, instead of detecting spam reviews directly, considerable efforts have been made to recognize review spammers, i.e., online users who have written spam reviews. Lim et al. (2010) studied different types of spamming behavioral indicators, and then used a regression method on the indicators for finding review spammers. Wang et al. (2012) investigated the relationships among reviewers, reviews, and stores, and developed a social review graph based method to identify online store spammers. Mukherjee et al. (2013) developed an author spamicity Bayesian model to exploit the observed behavioral footprints for spammer detection. In reality, a group of online users may work together to create spam reviews. Mukherjee et al. (2012) developed a group spam ranking algorithm to detect spammer groups.

Multi-task learning is a learning paradigm that seeks to boost generalization performance by learning a task together with other tasks at the same time while using a shared representation (Caruana, 1997). Most majority of existing work on multi-task learning does not infer actual task relations from training data automatically, instead, they typically make the assumptions that the relations are existent or are given as prior knowledge (Thrun and O'Sullivan, 1996; Bakker and Heskes, 2003; Evgeniou and Pontil, 2004; Argyriou et al., 2006; Liu et al., 2009). To better fit the multi-task learning model to real-world data, Zhang and Yeung (2010) proposed a convex regularization formulation named multi-task relation learning (MTRL), which can learn real relationships between tasks under a multi-task learning framework.

In this work, we focus on detecting online deceptive review spam. We formulate review spam detection for multiple domains (e.g., hotel and restaurant) as a multi-task learning problem. Following the convex framework of MTRL, we first develop a multi-task learning method via logistic regression (MTL-LR). We employ logistic regression as base classification model, because: 1) It is a robust model that does not have configuration parameters to tune; 2) It can be straightforwardly extended, and be efficiently trained using convex optimization techniques (Hoi et al., 2006; Minka, 2003); and 3) It has been shown effective for large-scale text classification and fake review detection problems (Hoi et al., 2006; Jindal and Liu, 2008). Then, to leverage the large volume of unlabeled review data, we extend the base logistic regression model, and incorporate a graph Laplacian regularizer into it. We thus develop a new semi-supervised multi-task learning paradigm via Laplacian regularized logistic regression, which is able to further boost the performance for review spam detection.

3 Methodology

3.1 Multi-task Learning via Logistic Regression

Given m review domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$, we accordingly have m review spam detection tasks $\mathcal{T}_1, \ldots, \mathcal{T}_m$, which share a common feature space with d dimensions. For the task $\mathcal{T}_i$ in the domain $\mathcal{D}_i$, there is a small labeled set of l_i review examples $\mathcal{L}_i = \{(\mathbf{x}_1^i, y_1^i), \ldots, (\mathbf{x}_{l_i}^i, y_{l_i}^i)\}$, where $\mathbf{x}_j^i \in \mathbb{R}^d$ is the vectorial representation of the review j in the labeled set $\mathcal{L}_i$, and $y_j^i \in \{+1, -1\}$ refers to the *spam*

(+1) or *nonspam* (−1) label of the review.

When there is only one review spam detection task, for example, $\mathcal{T}_i$, we can use logistic regression (LR) model to learn a supervised classifier based on the labeled set $\mathcal{L}_i$. The objective function of LR for single-task learning is

$$P^i_{LR}(\mathbf{w}_i) = \frac{1}{l_i}\sum_{j=1}^{l_i}\ln(1+\exp(-y^i_j\mathbf{w}_i^\top\mathbf{x}^i_j)) + \frac{\lambda}{2}\|\mathbf{w}_i\|^2,$$

where $\mathbf{w}_i \in \mathbb{R}^d$, $\lambda > 0$ refers to regularization parameter.

Once the model is learned from solving the optimization problem, given a test review instance $\mathbf{x}_{j'}$ of the task $\mathcal{T}_i$, we can employ the model to predict it as *spam*, i.e., $\hat{y}_{j'} = 1$, with probability

$$Prob(\hat{y}_{j'} = 1) = \frac{1}{1+\exp(-\mathbf{w}_i^\top\mathbf{x}^i_{j'})}.$$

Now we have m review spam detection tasks for multiple domains, and we would learn m supervised classification models simultaneously. To achieve this, we introduce a covariance matrix Ω to represent the correlations among the m review spam detection tasks, where Ω_{ij} refers to the relation/covariance between a pair of tasks $\mathcal{T}_i$ and $\mathcal{T}_j$. Since Ω is a task covariance matrix, we require it to satisfy the constraint $\Omega \succeq 0$, i.e., positive semidefinite. We also restrict $Tr(\Omega) = 1$ without of loss of generality, since for any covariance matrix $Tr(\Sigma) \neq 1$, we can use $\frac{\Sigma}{Tr(\Sigma)}$ as Ω. If the covariance matrix is given as prior knowledge, then we introduce a supervised multi-task learning (MTL) framework via logistic regression as follows

$$P^\Omega_{MTL}(\mathbf{W}) = \sum_{i=1}^{m}\frac{1}{l_i}\sum_{j=1}^{l_i}\ln(1+\exp(-y^i_j\mathbf{w}_i^T\mathbf{x}^i_j)) + \frac{\lambda}{2}Tr(\mathbf{W}\mathbf{W}^\mathbf{T}) + \frac{\beta}{2}Tr(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T),$$

where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$, and $\beta > 0$ is a regularization parameter.

Under this multi-task learning setting, the first term refers to the sum of all the average empirical loss, the second term refers to the regularizer used to avoid over-fitting, and the last term is introduced to leverage the shared knowledge from multiple learning tasks according to their relationships.

In reality, the covariance matrix may be not provided a priori. We then present the following multi-task learning model, which can learn the model parameters W and Ω automatically from training review data

$$P_{MTL}(\mathbf{W}, \mathbf{\Omega}) = \sum_{i=1}^{m}\frac{1}{l_i}\sum_{j=1}^{l_i}\ln(1+\exp(-y^i_j\mathbf{w}_i^T\mathbf{x}^i_j)) + \frac{\lambda}{2}Tr(\mathbf{W}\mathbf{W}^\mathbf{T}) + \frac{\beta}{2}Tr(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)$$
$$s.t. \quad \mathbf{\Omega} \succeq 0, \quad Tr(\mathbf{\Omega}) = 1,$$

If we have only one review spam detection task, i.e., $m = 1$, then it is straightforward to verify that the above multi-task learning formulation would be reduced to single-task objective function of logistic regression.

3.2 Semi-supervised Multi-task Learning via Laplacian Regularized Logistic Regression

Generally, for a given review domain $\mathcal{D}_i$, there is a large set of unlabeled reviews $\mathcal{U}_i = \{\mathbf{x}^i_{l_i+1}, \ldots, \mathbf{x}^i_{n_i}\}$ in addition to the labeled review set $\mathcal{L}_i$. Then, for each review spam detection task $\mathcal{T}_i$, we constracut a weighted neighborhood graph $\mathcal{G}_i = (V_i, E_i)$ based on both labeled and unlabeled review sets $\mathcal{L}_i$ and $\mathcal{U}_i$. V refers to the set of data points, each of which stands for a review example $\mathbf{x}^i_j$ ($j : 1, \ldots, n_i$) from either $\mathcal{L}_i$ or $\mathcal{U}_i$. E refers to the set of weighted edges. Specifically, if a review example/point $\mathbf{x}^i_j$ is among the K nearest neighbors of the review point $\mathbf{x}^i_k$, we put an edge linking the two examples, and vice versa. We also assign an adjacent weight score s^i_{jk} to the edge, which represents the similarity or closeness between the two reviews. Once the neighborhood graph $\mathcal{G}_i$ has been built for each task, a Laplacian regularizer can be then constrcted on the graph to extend the regular logistic regression model.

Considering the similarity matrix S_i that corresponds to the graph $\mathcal{G}_i$ for the task $\mathcal{T}_i$, it is expected that a good model would also minimize the follow-

ing objective

$$\sum_{jk} s^i_{jk}(\mathbf{w}_i^\top \mathbf{x}^i_j - \mathbf{w}_i^\top \mathbf{x}^i_k)^2,$$

This objective implies that $\mathbf{w}_i^\top \mathbf{x}^i_j$ should be close to $\mathbf{w}_i^\top \mathbf{x}^i_k$ if the similarity s^i_{jk} is large. The objective can be simplified as

$$\begin{aligned}&\sum_{jk} s^i_{jk}(\mathbf{w}_i^\top \mathbf{x}^i_j - \mathbf{w}_i^\top \mathbf{x}^i_k)^2\\ &= \mathrm{Tr}(\mathbf{w}_i^\top X_i(D_i - S_i)X_i^\top \mathbf{w}_i)\\ &= \mathrm{Tr}(\mathbf{w}_i^\top X_i L_i X_i^\top \mathbf{w}_i),\end{aligned}$$

where $D_i = diag(D^i_{jj})$ is a diagonal matrix, $D^i_{jj} = \sum_k s^i_{jk}$, and $L_i = D_i - S_i$ refers to the graph Laplacian matrix.

Then, given both labeled review set $\mathcal{L}_i$ and unlabeled set $\mathcal{U}_i$ for the task T_i, we extend the basic logistic regression by incorporating the graph Laplacian regularizer into its learning framework, and develop a new semi-supervised Laplacian regularized logistic regression (LLR) model. The objective function of LLR for semi-supervised single-task learning is given below

$$\begin{aligned}&P^i_{LLR}(\mathbf{w}_i)\\ &= \frac{1}{l_i}\sum_{j=1}^{l_i} \ln(1+\exp(-y^i_j \mathbf{w}_i^\top \mathbf{x}^i_j))\\ &+\frac{\lambda}{2}\|\mathbf{w}_i\|^2 + \frac{\gamma}{2}\mathrm{Tr}(\mathbf{w}_i^\top X_i L_i X_i^\top \mathbf{w}_i),\end{aligned}$$

where $\lambda > 0$ and $\gamma > 0$ are regularization parameters.

The semi-supervised formulation of LLR balances several desires. The first term is used to minimize the loss of the model on the labeled review data, the second term is used to minimize the complexity of the model, and the last term refers to the Laplacian regularizer, which is introduced to make the prediction of the model smooth on the whole review data set.

Next, based on the objective function of the above LLR model, we extend the supervised multi-task learning framework, and propose a novel semi-supervised multi-task learning paradigm via Laplacian regularized logistic regression (SMTL-LLR) as follows

$$\begin{aligned}&P_{SMTL}(\mathbf{W}, \mathbf{\Omega})\\ &= \sum_{i=1}^{m}\frac{1}{l_i}\sum_{j=1}^{l_i}\ln(1+\exp(-y^i_j \mathbf{w}_i^T \mathbf{x}^i_j))\\ &+\frac{\lambda}{2}Tr(\mathbf{W}\mathbf{W}^\top) + \frac{\beta}{2}Tr(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top)\\ &+\frac{\gamma}{2}\sum_{i=1}^{m}\frac{1}{n_i}Tr(\mathbf{w}_i^T X_i L_i X_i^T \mathbf{w}_i)\\ &s.t., \quad \mathbf{\Omega} \succeq 0, \quad Tr(\mathbf{\Omega}) = 1.\end{aligned}$$

Under this new semi-supervised unified framework, our proposed SMTL-LLR model can leverage the large amount of unlabeled review data in addition to the labeled ones to learn multiple review spam detection models simultaneously, and then, what is learned for one task can help other related tasks be learned better. In contrast, previous single-task learning based review spam detection models, which are trained independently, and are typically built on a limited set of labeled review data, cannot benefit from this.

3.3 Stochastic Alternating Method

There are two parameters $\mathbf{W}$ and Ω in the objective function of the proposed SMTL-LLR model. It is not easy to optimize the objective function against the two parameters at the same time. We then develop a stochastic alternating method to cope with the optimization problem for SMTL-LLR, i.e., alternatively updating one parameter by fixing the other. In particular, we initialize $\mathbf{W}$ with the values randomly chosen from $[0, 1]$, and initialize $\mathbf{\Omega}$ as a diagonal matrix, where $\mathbf{\Omega}_{ii} = \frac{1}{m}$. For each iteration, the key update steps for the two parameters are described as follows

- *Step 1*: Update $\mathbf{W}$ while $\mathbf{\Omega}$ is fixed.

$$\mathbf{W} \leftarrow \operatorname*{argmin}_{\mathbf{W}} P_{SMTL}(\mathbf{W}, \mathbf{\Omega})$$

- *Step 2*: Update $\mathbf{\Omega}$ while $\mathbf{W}$ is fixed.

$$\Omega \leftarrow \operatorname*{argmin}_{\Omega} P_{SMTL}(\mathbf{W}, \mathbf{\Omega})$$

3.3.1 Updating W While Fixing Ω

For *Step 1* of the alternating optimization method, we introduce a stochastic gradient descent method to efficiently update the parameter $\mathbf{W}$, while Ω is fixed. Formally, given a learning task T_i, we randomly choose a subset or mini-batch of reviews $A_b^i = \{(\mathbf{x}_j^i, y_j^i)|j \in [l_i]\}$ from the labeled set $\mathcal{L}_i$ in a particular iteration, where $[l_i]$ denotes $\{1, \ldots, l_i\}$ and $|A_b^i| = r \ll l_i$. Based on the subset of labeled reviews A_b^i, we can construct an unbiased estimate of the objective function

$$\begin{aligned}
&P_{SMTL}(\mathbf{W}, \mathbf{\Omega}, \{A_b^i\}_{i=1}^m) \\
&= \sum_{i=1}^{m} \frac{1}{r} \sum_{j \in A_b^i} \ln(1 + \exp(-y_j^i \mathbf{w}_i^T \mathbf{x}_j^i)) \\
&+ \frac{\lambda}{2} Tr(\mathbf{W}\mathbf{W}^{\mathbf{T}}) + \frac{\beta}{2} Tr(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \\
&+ \frac{\gamma}{2} \sum_{i=1}^{m} \frac{1}{n_i} Tr(\mathbf{w}_i^T X_i L_i X_i^T \mathbf{w}_i)
\end{aligned}$$

We can then obtain an unbiased stochastic gradient of the objective

$$\begin{aligned}
&\nabla_{\mathbf{W}} P_{SMTL}(\mathbf{W}, \mathbf{\Omega}, \{A_b^i\}_{i=1}^m) \\
&= [\mathbf{g}_b^1, \ldots, \mathbf{g}_b^m] + \lambda \mathbf{W} + \beta \mathbf{W}\Omega^{-1} \\
&+ [\gamma \frac{1}{n_1} X_1 L_1 X_1^T \mathbf{w}_1, \ldots, \gamma \frac{1}{n_m} X_m L_m X_m^T \mathbf{w}_m],
\end{aligned}$$

where

$$\mathbf{g}_b^i = \frac{1}{r} \sum_{j \in A_b^i} \frac{-y_j^i \mathbf{x}_j^i}{1 + \exp(y_j^i \mathbf{w}_i^T \mathbf{x}_j^i)}.$$

Next, the model parameter $\mathbf{W}$ can be updated via stochastic gradient descent method

$$\mathbf{W}_{t+\frac{1}{2}} = \mathbf{W}_t - \eta_t \nabla_{\mathbf{W}} P_{SMTL}(\mathbf{W}, \mathbf{\Omega}, \{A_b^i\}_{i=1}^m)$$

where $\eta_t > 0$ refers to learning rate in iteration *t*.

Note that, after each update step for the parameter $\mathbf{W}$, we perform a scaling process by forcing the solution

$$\|\mathbf{W}_{t+1}\|_F \leq \sqrt{2m \ln(2)/\lambda},$$

and then have the following update rule

$$\mathbf{W}_{t+1} = \min(1, \frac{\sqrt{2m \ln(2)/\lambda}}{\|\mathbf{W}_{t+\frac{1}{2}}\|_F}) \mathbf{W}_{t+\frac{1}{2}}.$$

We provide a straightforward theoretical analysis, which shows an upper bound of the norm of the optima solution $\mathbf{W}_*$, and explains why we perform the above scaling step. Using the fact that

$$P_{SMTL}(\mathbf{W}_*) \leq P_{SMTL}(0),$$

we thus have

$$\begin{aligned}
\frac{\lambda}{2} \|\mathbf{W}_*\|_F^2 &\leq P_{SMTL}(\mathbf{W}_*) \\
&\leq P_{SMTL}(0) = m \ln(2).
\end{aligned}$$

The fist inequality is guaranteed by

$$\ln(1 + \exp(-y_j^i \mathbf{w}_i^\top \mathbf{x}_j^i)) > 0,$$

$$Tr(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top) \geq 0,$$

and

$$\mathrm{Tr}(\mathbf{w}_i^\top X_i L_i X_i^\top \mathbf{w}_i) \geq 0.$$

3.3.2 Updating Ω While Fixing W

The second step of the stochastic alternating method is equivalent to solving the following optimization problem

$$\begin{aligned}
&\min_{\Omega} \ Tr(W\Omega^{-1}W^\top) \\
&s.t., \ \mathbf{\Omega} \succeq 0, \ Tr(\mathbf{\Omega}) = 1.
\end{aligned}$$

This convex formulation enjoys the following closed-form solution (Zhang and Yeung, 2010)

$$\mathbf{\Omega} = \frac{(\mathbf{W}^\top \mathbf{W})^{\frac{1}{2}}}{Tr((\mathbf{W}^\top \mathbf{W})^{\frac{1}{2}})}.$$

It is obviously observed that Ω models the correlations between each pair of the tasks or the models.

Algorithm 1 summarizes the stochastic alternating optimization method for SMTL-LLR. Given labeled and unlabeled review data for multiple review domains, we run the algorithm for P alternating loops. Within each loop p, we update the model parameter $\mathbf{W}$ for T iterations via stochastic gradient descent method, where *B* is number of mini-batches; after that, we update the task covariance matrix $\mathbf{\Omega}$ once based on new $\mathbf{W}$. The procedure is performed iteratively until it is converged. Then, multiple optimized review spam detection models and task covariance matrix would be learned finally.

Algorithm 1 Stochastic Alternating Method

Input:
Labeled and unlabeled review data for multiple tasks
Initial learning rate η_0, hyper-parameter δ
Regularization parameters λ, β, γ
Initialization:
Initialize $\mathbf{W}$ with values randomly chosen from [0, 1]
Initialize $\mathbf{\Omega} = diag(1/m, \ldots, 1/m)$
for $p = 1, \ldots, P$ **do**
  $\widetilde{\mathbf{W}}_1 = \mathbf{W}$
  for $t = 1, \ldots, T$ **do**
    Learning rate $\eta_t = \frac{\eta_0}{1+\eta_0 \delta t}$
    Randomly shuffle reviews in the training set
    for $b = 1, \ldots, B$ **do**
      Compute $\nabla_{\mathbf{W}} P_{SMTL}(\mathbf{W}, \mathbf{\Omega}, \{A_b^i\}_{i=1}^m)$
      Update $\widetilde{\mathbf{W}}_{t+\frac{1}{2}} = \widetilde{\mathbf{W}}_t$
        $-\eta_t \nabla_{\mathbf{W}} P_{SMTL}(\mathbf{W}, \mathbf{\Omega}, \{A_b^i\}_{i=1}^m)$
      $\widetilde{\mathbf{W}}_{t+1} = \min(1, \frac{\sqrt{2m\ln(2)/\lambda}}{\|\widetilde{\mathbf{W}}_{t+\frac{1}{2}}\|_F}) \widetilde{\mathbf{W}}_{t+\frac{1}{2}}$
    end for
  end for
  Update $\mathbf{W} = \widetilde{\mathbf{W}}_{T+1}$
  Update $\mathbf{\Omega} = \frac{(\mathbf{W}^\top \mathbf{W})^{\frac{1}{2}}}{Tr((\mathbf{W}^\top \mathbf{W})^{\frac{1}{2}})}$
end for
Output: $\mathbf{W}$ and $\mathbf{\Omega}$

In addition, we also rely on the stochastic alternating method to optimize the proposed MTL-LR method. Differently, we need to remove all the terms related to unlabeled data, i.e., discarding the Laplacian regularization term from the objective function and gradient.

4 Experiments

In this section, we evaluate the proposed multi-task learning methods MTL-LR and SMTL-LLR for review spam detection, and demonstrate the improved effectiveness of the methods over other well-established baselines.

4.1 Data Sets

Due to big challenge in manually recognizing deceptive reviews, there are limited benchmark opinion spam data in this field. We used three ground truth data sets from the review domains, *doctor*[2], *hotel*[3], and *restaurant*[4], respectively, to evaluate the proposed methods, which were created by following the similar rules used in (Ott et al., 2011). Then, for each ground truth review data set, we randomly collected a large number of unlabeled reviews (10,000), which were written about the same entities or domain. Table 2 shows some data statistics, where the last column computes the ratio of labeled reviews to unlabeled ones.

	Spam/Nonspam	Unlabeled	Ratio
Doctor	200/200	10,000	4.0%
Hotel	300/300	10,000	6.0%
Restaurant	200/200	10,000	4.0%

Table 2: Some statistics of review data sets.

4.2 Experimental Setup

We followed previous work (Mihalcea and Strapparava, 2009; Ott et al., 2011), and leveraged text unigram and bigram term-frequency features to train our models for review spam detection. This problem setting is quite useful, for example, when user behavior data are sparse or even not available in practical applications.

Supervised classification models, such as logistic regression (LR) and support vector machines (SVM), have been used to identify fake review spam (Jindal and Liu, 2008; Ott et al., 2011). We compared our methods with the two models. Semi-supervised positive-unlabeled (PU) learning was employed for review spam detection, then we chose one representative PU learning method (Liu et al., 2002) to evaluate our models. We did not compare our methods with the two-view co-training method, which was used for fake review detection (Li et al., 2011), because the *reviewer view* data are not available in the ground truth review sets. Instead, we selected a well-known semi-supervised transductive SVM (TSVM) (Joachims, 1999) to evaluate our models. Different from the proposed methods, we trained each of above baselines in a single domain, because they are single-task learning methods. Moreover, we also compared our methods with one well-established multi-task learning baseline MTRL (Zhang and Yeung, 2010), which has not been used

[2]https://www.ratemds.com

[3]https://www.tripadvisor.com

[4]http://www.yelp.com

for review spam detection problem.

It is important to specify appropriate values for the parameters in the proposed methods. In our setting, we used the learning rates η_t that asymptotically decrease with iteration numbers (Bottou, 2012). Following previous work (Ott et al., 2011; Chen and Chen, 2015), we conducted five-fold cross-validation experiments, and determined the values of the regularization and hyper parameters via a grid-search method.

4.3 Experimental Results

Table 3 reports the spam and nonspam review detection accuracy of our methods SMTL-LLR and MTL-LR against all other baseline methods. In terms of 5% significance level, the differences between SMTL-LLR and the baseline methods are considered to be statistically significant.

	Doctor	Hotel	Restaurant	Average
SMTL-LLR	85.4%	88.7%	87.5%	87.2%
MTL-LR	83.1%	86.7%	85.7%	85.2%
MTRL	82.0%	85.4%	84.7%	84.0%
TSVM	80.6%	84.2%	83.8%	82.9%
LR	79.8%	83.5%	83.1%	82.1%
SVM	79.0%	83.5%	82.9%	81.8%
PU	68.5%	75.4%	74.0%	72.6%

Table 3: Spam and nonspam review detection results in the doctor, hotel, and restaurant review domains.

Under symmetric multi-task learning setting, our methods SMTL-LLR and MTL-LR outperform all other baselines for identifying spam reviews from nonspam ones. MTL-LR achieves the average accuracy of 85.2% across the three domains, which is 3.1% and 3.4% better than LR and SVM trained in the single task learning setting, and 1.2% higher than MTRL. Training with a large quantity of unlabeled review data in addition to labeled ones, SMTL-LLR improves the performance of MTL-LR, and achieves the best average accuracy of 87.2% across the domains, which is 3.2% better than that of MTRL, and is 4.3% better than TSVM, a semi-supervised single task learning model. PU gives the worst performance, because learning only with partially labeled positive review data (spam) and unlabeled data may not generalize as well as other methods.

4.4 Performance versus Unlabeled Data Size

Figure 1 plots SMTL-LLR accuracy versus unlabeled data sizes from *0* to *10,000*, where *0* corresponds to using only labeled data to build the model, i.e., MTL-LR. Note that we first randomly sampled *2,000* unlabeled reviews to build the first set, and then created the second set by appending another randomly selected set of *2,000* reviews to the previous one. We repeated the process until all the unlabeled review data sets were created.

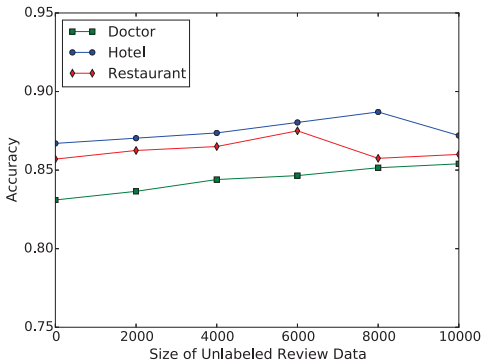


Figure 1: Accuracy versus Unlabeled Data Size.

We observed that learning from unlabeled reviews does help to boost the performance of MTL-LR, which was trained with labeled data alone. The performance of SMTL-LLR improves when training with more and more unlabeled review data. This is because the useful patterns learned from unlabeled data perhaps supports SMTL-LLR to generalize better. But continuing to learn from much more unlabeled reviews may even harm the performance. One explanation is that appending more unlabeled data may also incur noisy information to learning process. Interestingly, the performance of SMTL-LLR keeps increasing on the doctor domain, when training with more and more unlabeled reviews up to *10,000*. From above observations, we conclude that an elaborately selected set of high-quality unlabeled review data may help SMTL-LLR to learn better.

4.5 Task Correlation

Based on the covariance matrix (Ω) learned from the review spam detection tasks, we obtained the correlation between each pair of tasks for doctor, hotel, and restaurant domains, as shown in Table 4. The review spam detection tasks are highly correlated with each other for hotel and restaurant domains (0.772).

This is reasonable due to the large amount of commonality shared between the two domains. We can see that the tasks are also positively correlated between hotel and doctor, as well as between doctor and restaurant domains.

	Doctor	Hotel	Restaurant
Doctor	1.0	0.688	0.638
Hotel	0.688	1.0	0.772
Restaurant	0.638	0.772	1.0

Table 4: Task correlations.

4.6 Shared Text Features among Tasks

Table 5 lists top weighted shared text features among the review spam detection tasks for doctor, hotel, and restaurant domains. Generally, review spammers demonstrate similar motivations when creating deceptive review spam, i.e., promoting their own products/services or defaming those of their competitors. Though different aspects or entities can be commented on across different domains, we find that many features or expressions are indeed shared among the three review domains. As we know, deceptive reviewers normally write up reviews for making money, thus they prefer choosing exaggerated language in their lies, no matter which domains they are working with. As shown in the first row for spam category, they tend to exaggerate their sentiments using the words like "definitely", "sure", "highly", and so on.

In contrast, truthful reviewers contribute reviews for sharing their true feelings or personal anecdotes. They are willing to write up detailed factual experiences, for example, about the doctors they visited or delicious foods they enjoyed. Their reviews thus tend to contain language patterns in past tense, such as "went", "did", and "took" shown in the second row.

5 Conclusions

We have coped with the problem of detecting deceptive review spam. Given the limited labeled review data for individual domains, we formulated it as a multi-task learning problem. We first developed a multi-task learning method via logistic regression (MTL-LR), which allows to boost the learning for one task by sharing the knowledge contained in the training signals of other related tasks. To leverage the unlabeled data, we introduced a graph Laplacian regularizer into each base model, and proposed a semi-supervised multi-task learning model via Laplacian regularized logistic regression (SMTL-LLR). Moreover, to deal with the optimization problem, we developed a stochastic alternating method. Experimental results on real-world review data demonstrated the superiority of SMTL-LLR over several well-established baseline methods.

Labels	Features
Spam	staff, friendly,comfortable, really, right, experience, best, way, amazing, check, away, staff friendly, definitely, sure, highly recommend
Nonspam	good, just, like, went, did, people, excellent, took, wonderful, things, day, fantastic, know, going, nice

Table 5: Top weighted shared text features for spam/nonspam category across the three review domains.

For future work, we plan to create much more ground truth review data from other review domains and different applications like forums or microblogs, and further test our proposed models for deceptive opinion spam detection. We also plan to incorporate our model into a practical opinion mining system, in this way, more reliable opinion and sentiment analysis results can be then expected.

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2006. Multi-task feature learning. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 41–48, Vancouver, British Columbia, Canada.

Bart Bakker and Tom Heskes. 2003. Task clustering and gating for bayesian multitask learning. *The Journal of Machine Learning Research*, 4:83–99.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Léon Bottou. 1997. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436.

Rich Caruana. 1997. Multitask learning. In *Machine Learning*, pages 41–75.

Yu-Ren Chen and Hsin-Hsi Chen. 2015. Opinion spam detection in web forum: A real case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 173–183, Republic and Canton of Geneva, Switzerland.

Francesco De Comite, Francois Denis, Remi Gilleron, and Fabien Letouzey. 1999. Positive and Unlabeled Examples Help Learning. In *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence, pages 219–230, Tokyo, Japan. Springer Verlag.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, New York, NY, USA.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, pages 171–175.

D. Hernandez, R. Guzman, M. Montes-y-Gomez, and P. Rosso 2013. Using PU-learning to detect deceptive opinion spam. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 38–45, Atlanta, Georgia, USA.

Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web*, pages 633–642, New York, NY, USA.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 219–230, Palo Alto, California, USA.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, page 2488.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard H. Hovy. 2014. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1566–1576, Baltimore, MD, USA.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 939–948.

Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 387–394.

Jun Liu, Shuiwang Ji, and Jieping Ye. 2009. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312, Stroudsburg, PA, USA.

Thomas P. Minka. 2003. A comparison of numerical optimizers for logistic regression. Technical report, CMU Technical Report.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st International Conference on World Wide Web*, pages 191–200.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 632–640.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 309–319, Portland, Oregon.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October.

S. Thrun and J. O'Sullivan. 1996. Discovering structure in multiple learning tasks: The TC algorithm. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, San Mateo, CA. Morgen Kaufmann.

Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. 2012. Identify online store review spammers via social review graph. *ACM Trans. Intell. Syst. Technol.*, 3(4):61:1–61:21, September.

Yu Zhang and Dit-Yan Yeung. 2010. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 733–442, Catalina Island, CA, USA.

Regularizing Text Categorization with Clusters of Words

Konstantinos Skianis **François Rousseau** **Michalis Vazirgiannis**

LIX, École Polytechnique, France
kskianis@lix.polytechnique.fr

Abstract

Regularization is a critical step in supervised learning to not only address overfitting, but also to take into account any prior knowledge we may have on the features and their dependence. In this paper, we explore state-of-the-art structured regularizers and we propose novel ones based on clusters of words from LSI topics, word2vec embeddings and graph-of-words document representation. We show that our proposed regularizers are faster than the state-of-the-art ones and still improve text classification accuracy. Code and data are available online[1].

1 Introduction

Harnessing the full potential in text data has always been a key task for the NLP and ML communities. The properties hidden under the inherent high dimensionality of text are of major importance in tasks such as text categorization and opinion mining.

Although simple models like bag-of-words manage to perform well, the problem of overfitting still remains. Regularization as proven in Chen and Rosenfeld (2000) is of paramount importance in Natural Language Processing and more specifically language modeling, structured prediction, and classification. In this paper we build upon the work of Yogatama and Smith (2014b) who introduce prior knowledge of data as a regularization term. One of the most popular structured regularizers, the group lasso (Yuan and Lin, 2006), was proposed to avoid large L2 norms for groups of weights.

[1] https://goo.gl/mKqvro

In this paper, we propose novel linguistic structured regularizers that capitalize on the clusters learned from texts using the word2vec and graph-of-words document representation, which can be seen as group lasso variants. The extensive experiments we conducted demonstrate these regularizers can boost standard bag-of-words models on most cases tested in the task of text categorization, by imposing additional unused information as bias.

2 Background & Notation

We place ourselves in the scenario where we consider a prediction problem, in our case text categorization, as a loss minimization problem, i. e. we define a loss function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}, y)$ that quantifies the loss between the prediction $h_{\boldsymbol{\theta},b}(\boldsymbol{x})$ of a classifier parametrized by a vector of feature weights $\boldsymbol{\theta}$ and a bias b, and the true class label $y \in \boldsymbol{\mathcal{Y}}$ associated with the example $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$. Given a training set of N data points $\{(\boldsymbol{x}^i, y^i)\}_{i=1...N}$, we want to find the optimal set of feature weights $\boldsymbol{\theta}^*$ such that:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i)}_{\textbf{empirical risk}} \tag{1}$$

In the case of logistic regression with binary predictions ($\boldsymbol{\mathcal{Y}} = \{-1, +1\}$), $h_{\boldsymbol{\theta},b}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} + b$ and $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}, y) = e^{-y h_{\boldsymbol{\theta},b}(\boldsymbol{x})}$ (log loss).

2.1 Regularization

Only minimizing the empirical risk can lead to overfitting, that is, the model no longer learns the underlying pattern we are trying to capture but fits the

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1827–1837,
Austin, Texas, November 1-5, 2016.

noise contained in the training data and thus results in poorer generalization (e. g., lower performances on the test set). For instance, along with some feature space transformations to obtain non-linear decision boundaries in the original feature space, one could imagine a decision boundary that follows every quirk of the training data. Additionally, if two hypothesis lead to similar low empirical risks, one should select the "simpler" model for better generalization power, simplicity assessed using some measure of model complexity.

Loss+Penalty Regularization takes the form of additional constraints to the minimization problem, i. e. a budget on the feature weights, which are often relaxed into a penalty term $\Omega(\boldsymbol{\theta})$ controlled via a Lagrange multiplier λ. We refer to the book of Boyd and Vandenberghe (2004) for the theory behind convex optimization. Therefore, the overall expected risk (Vapnik, 1991) is the weighted sum of two components: the empirical risk and a regularization penalty term, expression referred to as "Loss+Penalty" by Hastie et al. (2009). Given a training set of N data points $\{(\boldsymbol{x}^i, y^i)\}_{i=1\dots N}$, we now want to find the optimal set of feature weights $\boldsymbol{\theta}^*$ such that:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underbrace{\underbrace{\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i)}_{\textbf{empirical risk}} + \underbrace{\lambda\Omega(\boldsymbol{\theta})}_{\textbf{penalty term}}}_{\textbf{expected risk}} \quad (2)$$

L_1 and L2 regularization The two most used penalty terms are known as L_1 regularization, a. k. a. *lasso* (Tibshirani, 1996), and L2 regularization, a. k. a. *ridge* (Hoerl and Kennard, 1970) as they correspond to penalizing the model with respectively the L_1 and L2 norm of the feature weight vector $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i) + \lambda \sum_{j=1}^{p} |\boldsymbol{\theta}_j| \quad (3)$$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i) + \lambda \sum_{j=1}^{p} \boldsymbol{\theta}_j{}^2 \quad (4)$$

Prior on the feature weights L_1 (resp. L2) regularization can be interpreted as adding a Laplacian (resp. Gaussian) prior on the feature weight vector. Indeed, given the training set, we want to find the most likely hypothesis $h^* \in \boldsymbol{\mathcal{H}}$, i. e. the one with *maximum a posteriori* probability:

$$\begin{aligned}
h^* &= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\mathbb{P}(h|\{(\boldsymbol{x}^i, y^i)\}_{i=1\dots N})\right) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\frac{\mathbb{P}(\{y^i\}_i|\{\boldsymbol{x}^i\}_i, h)\,\mathbb{P}(h|\{\boldsymbol{x}^i\}_i)}{\mathbb{P}(\{y^i\}_i|\{\boldsymbol{x}^i\}_i)}\right) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\mathbb{P}(\{y^i\}_i|\{\boldsymbol{x}^i\}_i, h)\,\mathbb{P}(h|\{\boldsymbol{x}^i\}_i)\right) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\mathbb{P}(\{y^i\}_i|\{\boldsymbol{x}^i\}_i, h)\,\mathbb{P}(h)\right) \quad (5) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\prod_{i=1}^{N} \left(\mathbb{P}(y^i|\boldsymbol{x}^i, h)\right)\mathbb{P}(h)\right) \quad (6) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmax}} \left(\sum_{i=1}^{N} \left(\log \mathbb{P}(y^i|\boldsymbol{x}^i, h)\right) + \log \mathbb{P}(h)\right) \\
&= \underset{h\in\boldsymbol{\mathcal{H}}}{\operatorname{argmin}} \left(\underbrace{\sum_{i=1}^{N} \left(-\log \mathbb{P}(y^i|\boldsymbol{x}^i, h)\right)}_{\textbf{empirical risk}} \underbrace{- \log \mathbb{P}(h)}_{\textbf{penalty term}}\right)
\end{aligned}$$

For the derivation, we assumed that the hypothesis h does not depend on the examples alone (Eq. 5) and that the N training labeled examples are drawn from an i.i.d. sample (Eq. 6). In that last form, we see that the loss function can be interpreted as a negative log-likelihood and the regularization penalty term as a negative log-prior over the hypothesis. Therefore, if we assume a multivariate Gaussian prior on the feature weight vector of mean vector $\mathbf{0}$ and covariance matrix $\Sigma = \sigma^2 I$ (i. e. independent features of same prior standard deviation σ), we do obtain the L2 regularization:

$$\mathbb{P}(h) = \frac{1}{\sqrt{(2\pi)^p|\Sigma|}} e^{-\frac{1}{2}\boldsymbol{\theta}^\top\Sigma^{-1}\boldsymbol{\theta}} \quad (7)$$

$$\begin{aligned}
\Rightarrow -\log \mathbb{P}(h) &= \frac{1}{2\sigma^2}\boldsymbol{\theta}^\top I\boldsymbol{\theta} + \frac{p}{2}\log(2\pi\sigma) \\
&\overset{\text{argmax}}{=} \lambda\|\boldsymbol{\theta}\|_2{}^2, \quad \lambda = \frac{1}{2\sigma^2} \quad (8)
\end{aligned}$$

And similarly, if we assume a multivariate Laplacian prior on the feature weight vector (i. e. $\boldsymbol{\theta}_i \sim Laplace(0, \frac{1}{\lambda})$), we obtain L_1-regularization. In practice, in both cases, the priors basically mean that we expect weights around 0 on average. The main difference between L_1 and L2 regularization is that the Laplacian prior will result in explicitly setting some feature weights to 0 (feature sparsity) while the Gaussian prior will only result in reducing their values (shrinkage).

2.2 Structured regularization

In L_1 and L2 regularizations, features are considered as independent, which makes sense without any additional prior knowledge. However, similar features have similar weights in the case of linear classifiers – equal weights for redundant features in the extreme case – and therefore, if we have some prior knowledge on the relationships between features, we should include that information for better generalization, i. e. include it in the regularization penalty term. Depending on how the similarity between features is encoded, e. g., through sets, trees (Kim and Xing, 2010; Liu and Ye, 2010; Mairal et al., 2010) or graphs (Jenatton et al., 2010), the penalization term varies but in any case, we take into account the structure between features, hence the "structured regularization" terminology. It should not be confused with "structured prediction" where this time the outcome is a structured object as opposed to a scalar (e. g., a class label) classically.

Group lasso Bakin (1999) and later Yuan and Lin (2006) proposed an extension of L_1 regularization to encourage groups of features to either go to zero (as a group) or not (as a group), introducing *group* sparsity in the model. To do so, they proposed to regularize with the $L_{1,2}$ norm of the feature weight vector:

$$\Omega(\boldsymbol{\theta}) = \lambda \sum_g \lambda_g \|\boldsymbol{\theta}_g\|_2 \tag{9}$$

where $\boldsymbol{\theta}_g$ is the subset of feature weights restricted to group g. Note that the groups can be overlapping (Jacob et al., 2009; Schmidt and Murphy, 2010; Jenatton et al., 2011; Yuan et al., 2011) even though it makes the optimization harder.

2.3 Learning

In our case we use a logistic regression loss function in order to integrate our regularization terms easily.

$$\mathcal{L}(x, \boldsymbol{\theta}, y) = \log(1 + \exp(-y\boldsymbol{\theta}^T x)) \tag{10}$$

It is obvious that the framework can be extended to other loss functions (e. g., hinge loss).

For the case of structured regularizers, there exist a plethora of optimization methods such group lasso. Since our tasks involves overlapping groups, we select the method of Yogatama and Smith (2014b).

Algorithm 1 ADMM for overlapping group-lasso

Require: augmented Lagrangian variable ρ, regularization strengths λ_{glas} and λ_{las}

1: **while** update in weights not small **do**
2: $\quad \boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \Omega_{las}(\boldsymbol{\theta}) + \mathcal{L}(\boldsymbol{\theta}) + \frac{\rho}{2} \sum_{i=1}^{V} N_i(\boldsymbol{\theta}_i - \mu_i)^2$
3: $\quad$ **for** $g = 1$ to G **do**
4: $\quad\quad \mathrm{v}_g = \mathrm{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(z_g)$
5: $\quad$ **end for**
6: $\quad u = u + \rho(\mathrm{v} - M\boldsymbol{\theta})$
7: **end while**

Their method uses the alternating directions method of multipliers (Hestenes, 1969; Powell, 1969).

Now given the lasso penalty for each feature and the group lasso regularizer, the problem becomes:

$$\min_{\boldsymbol{\theta}, \mathrm{v}} \Omega_{las}(\boldsymbol{\theta}) + \Omega_{glas}(\mathrm{v}) + \sum_{d=1}^{D} \mathcal{L}(x_d, \boldsymbol{\theta}, y_d) \tag{11}$$

so that $\mathbf{v} = M\boldsymbol{\theta}$, where $\mathbf{v}$ is a copy-vector of $\boldsymbol{\theta}$. The copy-vector $\mathbf{v}$ is needed because the group-lasso regularizer contains overlaps between the used groups. M is an indicator matrix of size $L \times V$, where L is the sum of the total sizes of all groups, and its ones show the link between the actual weights $\boldsymbol{\theta}$ and their copies $\mathbf{v}$. Following Yogatama and Smith (2014b) a constrained optimization problem is formed, that can be transformed to an augmented Lagrangian problem:

$$\Omega_{las}(\boldsymbol{\theta}) + \Omega_{glas}(\mathrm{v}) + \mathcal{L}(\boldsymbol{\theta}) + \mathrm{u}^\top(\mathrm{v} - M\boldsymbol{\theta}) + \frac{\rho}{2}\|\mathrm{v} - M\boldsymbol{\theta}\|_2^2 \tag{12}$$

Essentially, the problem becomes the iterative update of $\boldsymbol{\theta}$, v and u:

$$\min_{\boldsymbol{\theta}} \Omega_{las}(\boldsymbol{\theta}) + \mathcal{L}(\boldsymbol{\theta}) + \mathrm{u}^\top M\boldsymbol{\theta} + \frac{\rho}{2}\|\mathrm{v} - M\boldsymbol{\theta}\|_2^2 \tag{13}$$

$$\min_{\mathrm{v}} \Omega_{glas}(\mathrm{v}) + \mathrm{u}^\top \mathrm{v} + \frac{\rho}{2}\|\mathrm{v} - M\boldsymbol{\theta}\|_2^2 \tag{14}$$

$$\mathrm{u} = \mathrm{u} + \rho(\mathrm{v} - M\boldsymbol{\theta}) \tag{15}$$

Convergence Yogatama and Smith (2014b) proved that ADMM for sparse overlapping group lasso converges. It is also shown that a good approximate solution is reached in a few tens of iterations. Our experiments confirm this as well.

3 Structured Regularization in NLP

In recent efforts there are results to identify useful structures in text that can be used to enhance the effectiveness of the text categorization in a NLP context. Since the main regularization approach we are going to use are variants of the group lasso, we are interested on prior knowledge in terms of groups/clusters that can be found in the training text data. These groups could capture either semantic, or syntactic structures that affiliate words to communities. In our work, we study both semantic and syntactic properties of text data, and incorporate them in structured regularizer. The grouping of terms is produced by either LSI or clustering in the word2vec or graph-of-words space.

3.1 Statistical regularizers

In this section, we present *statistical* regularizers, i. e. with groups of words based on co-occurrences, as opposed to *syntactic* ones (Mitra et al., 1997).

Network of features Sandler et al. (2009) introduced regularized learning with networks of features. They define a graph G whose edges are non-negative with larger weights indicating greater similarity. Conversely, a weight of zero means that two features are not believed a priori to be similar. Previous work (Ando and Zhang, 2005; Raina et al., 2006; Krupka and Tishby, 2007) shows such similarities can be inferred from prior domain knowledge and statistics computed on unlabeled data.

The weights of G are mapped in a matrix P, where $P_{ij} \geq 0$ gives the weight of the directed edge from vertex i to vertex j. The out-degree of each vertex is constrained to sum to one, $\sum_j P_{ij} = 1$, so that no feature "dominates" the graph.

$$\Omega_{network}(\boldsymbol{\theta}) = \lambda_{net} \sum \boldsymbol{\theta}_k^\top M \boldsymbol{\theta}_k \qquad (16)$$

where $M = \alpha(I-P)^\top(I-P)+\beta I$. The matrix M is symmetric positive definite, and therefore it possesses a Bayesian interpretation in which the weight vector $\boldsymbol{\theta}$, is *a priori* normally distributed with mean zero and covariance matrix $2M^{-1}$. However, preliminary results show poorer performance compared to structured regularizers in larger datasets.

Sentence regularizer Yogatama and Smith (2014b) proposed to define groups as the sentences in the training dataset. The main idea is to define a group $d_{d,s}$ for every sentence s in every training document d so that each group holds weights for occurring words in its sentence. Thus a word can be a member of one group for every distinct (training) sentence it occurs in. The regularizer is:

$$\Omega_{sen}(\boldsymbol{\theta}) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \lambda_{d,s} \|\boldsymbol{\theta}_{d,s}\|_2 \qquad (17)$$

where S_d is the number of sentences in document d.

Since modern text datasets typically contain thousands of sentences and many words appear in more than one sentence, the sentence regularizer could potentially lead to thousands heavily overlapping groups. As stated in the work of Yogatama and Smith (2014b), a rather important fact is that the regularizer will force all the weights of a sentence, if it is recognized as irrelevant. Respectively, it will keep all the weights of a relevant sentence, even though the group contains unimportant words. Fortunately, the problem can be resolved by adding a lasso regularization (Friedman et al., 2010).

3.2 Semantic regularizers

In this section, we present *semantic* regularizers that define groups based on how semantically close words are.

LDA regularizer Yogatama and Smith (2014a) considered topics as another type of structure. It is obvious that textual data can contain a huge number of topics and especially topics that overlap each other. Again the main idea is to penalize weights for words that co-occur in the same topic, instead of treating the weight of each word separately.

Having a training corpus, topics can be easily extracted with the help of the latent Dirichlet allocation (LDA) model (Blei et al., 2003). In our experiments, we form a group by extracting the n most probable words in a topic. We note that the extracted topics can vary depending the text preprocessing methods we apply on the data.

LSI regularizer Latent Semantic Indexing (LSI) can also be used in order to identify topics or groups and thus discover correlation between terms (Deerwester et al., 1990). LSI uses singular value decomposition (SVD) on the document-term matrix to

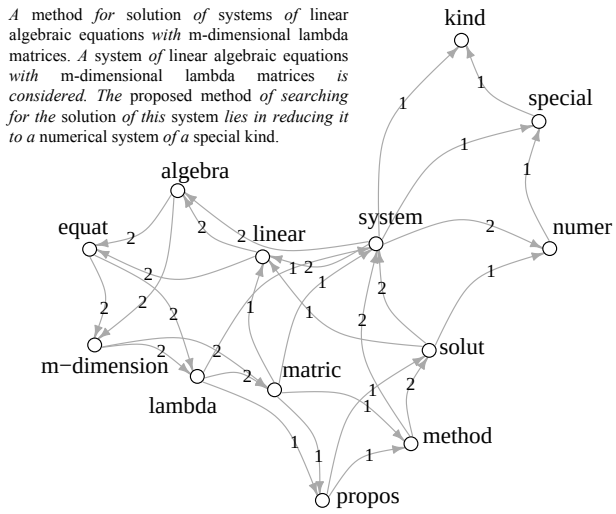


Figure 1: A Graph-of-words example.

identify latent variables that link co-occurring terms with documents. The main basis behind LSI is that words being used in the same contexts (i. e. the documents) tend to have similar meanings. We used LSI as a baseline and compare it with other standard baselines as well as other proposed structured regularizers. In our work we keep the top 10 words which contribute the most in a topic.

The regularizer for both LDA and LSI is:

$$\Omega_{LDA,LSI}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda \|\boldsymbol{\theta}_k\|_2 \qquad (18)$$

where K is the number of topics.

3.3 Graphical regularizers

In this section we present our proposed regularizers based on graph-of-words and word2vec. Essentially the word2vec space can be seen as a large graph where nodes represent terms and edges similarities between them.

Graph-of-words regularizer Following the idea of the network of features, we introduce a simpler and faster technique to identify relationships between features. We create a big collection graph from the training documents, where the nodes correspond to terms and edges correspond to co-occurrence of terms in a sliding window. We present a toy example of a graph-of-words in Figure 1.

A critical advantage of graph-of-words is that it easily encodes term dependency and term order (via edge direction). The strength of the dependence between two words can also be captured by assigning a weight to the edge that links them.

Graph-of-words was originally an idea of Mihalcea and Tarau (2004) and Erkan and Radev (2004) who applied it to the tasks of unsupervised keyword extraction and extractive single document summarization. Rousseau and Vazirgiannis (2013) and Malliaros and Skianis (2015) showed it performs well in the tasks of information retrieval and text categorization. Notably, the former effort ranked nodes based on a modified version of the PageRank algorithm.

Community detection on graph-of-words Our goal is to identify groups or communities of words. Having constructed the collection-level graph-of-words, we can now apply community detection algorithms (Fortunato, 2010).

In our case we use the Louvain method, a community detection algorithm for non-overlapping groups described in the work of Blondel et al. (2008). Essentially it is a fast modularity maximization approach, which iteratively optimizes local communities until we reach optimal global modularity given some perturbations to the current community state. The regularizer becomes:

$$\Omega_{gow}(\boldsymbol{\theta}) = \sum_{c=1}^{C} \lambda \|\boldsymbol{\theta}_c\|_2 \qquad (19)$$

where c ranges over the C communities. Thus $\boldsymbol{\theta}_c$ corresponds to the sub-vector of $\boldsymbol{\theta}$ such that the corresponding features are present in the community c. Note that in this case we do not have overlapping groups, since we use a non-overlapping version of the algorithm.

As we observe that the collection-level graph-of-words does not create well separated communities of terms, overlapping community detection algorithms, like the work of Xie et al. (2013) fail to identify “good" groups and do not offer better results.

Word2vec regularizer Mikolov et al. (2013) proposed the word2vec method for learning continuous vector representations of words from large text datasets. Word2vec manages to capture the actual meaning of words and map them to a multi-dimensional vector space, giving the possibility of

applying vector operations on them. We introduce another novel regularizer method, by applying unsupervised clustering algorithms on the word2vec space.

Clustering on word2vec Word2vec contains millions of words represented as vectors. Since word2vec succeeds in capturing semantic similarity between words, semantically related words tend to group together and create large clusters that can be interpreted as "topics".

In order to extract these groups, we use a fast clustering algorithm such as K-Means (Macqueen, 1967) and especially Minibatch K-means. The regularizer is:

$$\Omega_{word2vec}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda \|\boldsymbol{\theta}_k\|_2 \qquad (20)$$

where K is the number of clusters we extracted from the word2vec space.

Clustering these semantic vectors is a very interesting area to study and could be a research topic by itself. The actual clustering output could vary as we change the number of clusters we are trying to identify. In this paper we do not focus on optimizing the clustering process.

4 Experiments

We evaluated our structured regularizers on several well-known datasets for the text categorization task. Table 1 summarizes statistics about the ten datasets we used in our experiments.

4.1 Datasets

Topic categorization. From the 20 Newsgroups[2] dataset, we examine four binary classification tasks. We end up with binary classification problems, where we classify a document according to two related categories: comp.sys: ibm.pc.hardware vs. mac.hardware; rec.sport: baseball vs. hockey; sci: med vs. space and alt.atheism vs. soc.religion.christian. We use the 20NG dataset from Python.

Sentiment analysis. The sentiment analysis datasets we examined include movie reviews (Pang and Lee, 2004; Zaidan and Eisner, 2008)[3], floor speeches by U.S. Congressmen deciding "yea"/"nay" votes on the bill under discussion (Thomas et al., 2006)[3] and product reviews from Amazon (Blitzer et al., 2007)[4].

	dataset	train	dev	test	# words	# sents
20NG	science	949	238	790	25787	16411
	sports	957	240	796	21938	14997
	religion	863	216	717	18822	18853
	comp.	934	234	777	16282	10772
Sentiment	vote	1175	257	860	19813	43563
	movie	1600	200	200	43800	49433
	books	1440	360	200	21545	13806
	dvd	1440	360	200	21086	13794
	electr.	1440	360	200	10961	10227
	kitch.	1440	360	200	9248	8998

Table 1: Descriptive statistics of the datasets

4.2 Experimental setup

As features we use unigram frequency concatenated with an additional unregularized bias term. We reproduce standard regularizers like lasso, ridge, elastic and state-of-the-art structured regularizers like sentence, LDA as baselines and compare them with our proposed methods.

For LSI, LDA and word2vec we use the gensim package (Řehůřek and Sojka, 2010) in Python. For the learning part we used Matlab and specifically code by Schmidt et al. (2007).

We split the training set in a stratified manner to retain the percentage of classes. We use 80% of the data for training and 20% for validation.

All the hyperparameters are tuned on the development dataset, using accuracy as the evaluation criterion. For lasso and ridge regularization, we choose λ from $\{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$. For elastic net, we perform grid search on the same set of values as ridge and lasso experiments for λ_{rid} and λ_{las}. For the LDA, LSI, sentence, graph-of-words (GoW), word2vec regularizers, we perform grid search on the same set of values as ridge and lasso experiments for the ρ, λ_{glas}, λ_{las} parameters. In the case we get the same accuracy on the development data, the model with the highest sparsity is selected. For

[2] http://qwone.com/~jason/20Newsgroups/

[3] http://www.cs.cornell.edu/~ainur/data.html

[4] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	0.946	0.916	0.954	0.954	**0.968**	**0.968***	0.942	0.967*	**0.968***
	sports	0.908	0.907	0.925	0.920	0.959	0.964*	**0.966**	0.959*	0.946*
	religion	0.894	0.876	0.895	0.890	0.918	0.907*	**0.934**	0.911*	0.916*
	computer	0.846	0.843	0.869	0.856	0.891	0.885*	0.904	0.885*	**0.911***
Sentiment	vote	0.606	0.643	0.616	0.622	**0.658**	0.653	0.656	0.640	0.651
	movie	0.865	0.860	0.870	0.875	**0.900**	0.895	0.895	0.895	0.890
	books	0.750	0.770	0.760	0.780	0.790	0.795	0.785	0.790	**0.800**
	dvd	0.765	0.735	0.770	0.760	0.800	**0.805***	0.785	0.795*	0.795*
	electr.	0.790	0.800	0.800	**0.825**	0.800	0.815	0.805	0.820	0.815
	kitch.	0.760	0.800	0.775	0.800	0.845	**0.860***	0.855	0.840	0.855*

Table 2: Accuracy results on the test sets. Bold font marks the best performance for a dataset. * indicates statistical significance of improvement over lasso at $p < 0.05$ using micro sign test for one of our models LSI, GoW and word2vec (underlined).

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	100	1	100	63	19	20	86	19	21
	sports	100	1	100	5	60	11	6.4	55	44
	religion	100	1	100	3	94	31	99	10	85
	computer	100	2	100	7	40	35	77	38	18
Sentiment	vote	100	1	100	8	15	16	13	97	13
	movie	100	1	100	59	72	81	55	90	62
	books	100	3	100	14	41	74	72	90	99
	dvd	100	2	100	28	64	8	8	58	64
	electr.	100	4	100	6	10	8	43	8	9
	kitch.	100	5	100	79	73	44	27	75	46

Table 3: Fraction (in %) of non-zero feature weights in each model for each dataset: the smaller, the more compact the model.

LDA we set the number of topics to 1000 and we keep the 10 most probable words of each topic as a group. For LSI we keep 1000 latent dimensions and we select the 10 most significant words per topic. For the clustering process on word2vec we ran Minibatch-Kmeans for max 2000 clusters. For each word belonging to a cluster, we also keep the top 5 or 10 nearest words so that we introduce overlapping groups. The intuition behind this is that words can be part of multiple “concepts" or topics, thus they can belong to many clusters.

4.3 Results

In Table 2 we report the results of our experiments on the aforementioned datasets, and we distinguish our proposed regularizers LSI, GoW, word2vec with underlining. Our results are inline and confirm that of (Yogatama and Smith, 2014a) showing the advantages of using structured regularizers in the text categorization task. The group based regularizers perform systematically better than the baseline ones.

We observe that the word2vec clustering based regularizers performs very well - achieving best performance for three out of the ten data sets while it is quite fast with regards to execution time as it appears in Table 3 (i. e. it is four to ten times faster than the sentence based one).

The LSI based regularization, proposed for the first time in this paper, performs surprisingly well as it achieves the best performance for three of the ten datasets. This is somehow interpreted by the fact that this method extracts the inherent dimensions that best represent the different semantics of the documents - as we see as well in the anecdotal

	dataset	GoW	word2vec
20NG	science	79	691
	sports	137	630
	religion	35	639
	computer	95	594

Table 4: Number of groups.

	dataset	lasso	ridge	elastic	group lasso				
					LDA	LSI	sentence	GoW	word2vec
20NG	science	10	1.6	1.6	15	11	76	12	19
	sports	12	3	3	7	20	67	5	9
	religion	12	3	7	10	4	248	6	20
	computer	7	1.4	0.8	8	6	43	5	10

Table 5: Time (in seconds) for learning with best hyperparameters.

$= 0$	piscataway combination jil@donuts0.uucp jamie reading/seeing chambliss left-handedness abilities lubin acad sci obesity page erythromycin bottom
$\neq 0$	and space the launch health for use that medical you space cancer and nasa hiv health shuttle for tobacco that cancer that research center space hiv aids are use theory keyboard data telescope available are from system information space ftp

Table 6: Examples with LSI regularizer.

$= 0$	village town edc fashionable trendy trendy fashionable points guard guarding crown title champion champions
$\neq 0$	numbness tingling dizziness fevers laryngitis bronchitis undergo undergoing undergoes undergone healed mankind humanity civilization planet nasa kunin lang tao kay kong

Table 7: Examples with word2vec regularizer.

examples in Table 6, 7, 8. This method proves as well very fast as it appears in Table 5 (i.e. it is three to sixty times faster than the sentence based one).

The GoW based regularization although very fast, did not outperform the other methods (while it has a very good performance in general). It remains to be seen whether a more thorough parameter tuning and community detection algorithm selection would improve further the accuracy of the method.

In Table 3 we present the feature space sizes retained by each of the regularizers for each dataset. As expected the lasso regularizer sets the vast majority of the features' weights to zero, and thus a very sparse feature space is generated. This fact has as a consequence the significant decrease in accuracy performance. Our proposed structured regularizers

$= 0$	islands inta spain galapagos canary originated anodise advertises jewelry mercedes benzes diamond trendy octave chanute lillienthal
$\neq 0$	vibrational broiled relieving succumb spacewalks dna nf-psychiatry itself commented usenet golded insects alternate self-consistent retrospect

Table 8: Examples with graph-of-words regularizer.

managed to perform better in most of the cases, introducing more sparse models compared to the state-of-the-art regularizers.

4.4 Time complexity

Although certain types of structured regularizers improve significantly the accuracy and address the problem of overfitting, they require a notable amount of time in the learning process.

As seen in Yogatama and Smith (2014b), a considerable disadvantage is the need of search for the optimal hyperparameters: λ_{glas}, λ_{lasso} , and ρ, whereas standard baselines like lasso and ridge only have one hyperparameter and elastic net has two.

Parallel grid search can be critical for finding the optimal set of hyperparameters, since there is no dependency on each other, but again the process can be very expensive. Especially for the case of the sentence regularizer, the process can be extremely slow due to two factors. First, the high number of sentences in text data. Second, sentences consist of heavily overlapping groups, that include words reappearing in one or more sentences. On the contrary, as it appears on Table 4, the number of clusters in the clustering based regularizers is significantly smaller than that of the sentences - and definitely controlled by the designer - thus resulting in much faster computation. The update of $\mathbf{v}$ still remains time consuming for small datasets, even with parallelization.

Our proposed structured regularizers are considerably faster in reaching convergence, since they of-

fer a smaller number of groups with less overlapping between words. For example, on the computer subset of the 20NG dataset, learning models with the best hyperparameter value(s) for lasso, ridge, and elastic net took 7, 1.4, and 0.8 seconds, respectively, on an Intel Xeon CPU E5-1607 3.00 GHz machine with 4 cores and 128GB RAM. Given the best hyperparameter values the LSI regularizer takes 6 seconds to converge, the word2vec regularizer takes 10 seconds to reach convergence, the graph-of-words takes 4 seconds while the sentence regularizer requires 43 seconds. Table 5 summarizes required learning time on 20NG datasets.

We also need to consider the time needed to extract the groups. For word2vec, Minibatch K-means requires 15 minutes to cluster the pre-trained vectors by Google. The clustering is executed only once. Getting the clusters of words that belong to the vocabulary of each dataset requires 20 minutes, but can be further optimized. Finding also the communities in the graph-of-words approach with the Louvain algorithm, is very fast and requires a few minutes depending on the size and structure of the graph.

In Tables 6, 7, 8 we show examples of our proposed regularizers-removed and -selected groups (in v) in the science subset of the 20NG dataset. Words with weights (in w) of magnitude greater than 10^{-3} are highlighted in red (sci.med) and blue (sci.space).

5 Conclusion & Future Work

This paper proposes new types of structured regularizers to improve not only the accuracy but also the efficiency of the text categorization task. We mainly focused on how to find and extract semantic and syntactic structures that lead to sparser feature spaces and therefore to faster learning times. Overall, our results demonstrate that linguistic prior knowledge in the data can be used to improve categorization performance for baseline bag-of-words models, by mining inherent structures. We only considered logistic regression because of its interpretation for L2 regularizers as Gaussian prior on the feature weights and following Sandler et al. (2009), we considered a non-diagonal covariance matrix for L2 based on word similarity before moving to group lasso as presented in the paper. We are not expecting a significant change in results with different loss functions as the proposed regularizers are not log loss specific.

Future work could involve a more thorough investigation on how to create and cluster graphs, i. e. covering weighted and/or signed cases. Finding better clusters in the word2vec space is also a critical part. This is not only restricted in finding the best number of clusters but what type of clusters we are trying to extract. Gaussian Mixture Models (McLachlan and Basford, 1988) could be applied in order to capture overlapping groups at the cost of high complexity. Furthermore, topical word embeddings (Liu et al., 2015) can be considered for regularization. This approach could enhance the regularization on topic specific datasets. Additionally, we plan on exploring alternative regularization algorithms diverging from the group-lasso method.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Sergey Bakin. 1999. *Adaptive regression and model selection in data mining problems*. Ph.D., The Australian National University, Canberra, Australia, May.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 440–447. ACL.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Santo Fortunato. 2010. Community detection in graphs. *Physics reports*, 486(3):75–174.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. 2010. A note on the group lasso and a sparse group lasso. Technical report, Department of Statistics, Stanford University.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The elements of statistical learning*, volume 2. Springer.

Magnus R. Hestenes. 1969. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303—-320.

Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. 2009. Group Lasso with Overlap and Graph Lasso. In *Proceedings of the 26th International Conference on Machine Learning*, ICML '09, pages 433–440.

Rodolphe Jenatton, Julien Mairal, Francis Bach, and Guillaume Obozinski. 2010. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning*, ICML '10, pages 487–494.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. 2011. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12:2777–2824.

Seyoung Kim and Eric P. Xing. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th International Conference on Machine Learning*, ICML '10, pages 543–550.

Eyal Krupka and Naftali Tishby. 2007. Incorporating Prior Knowledge on Features into Learning. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2 of *AISTATS '07*, pages 227–234.

Jun Liu and Jieping Ye. 2010. Moreau-Yosida Regularization for Grouped Tree Structure Learning. In *Advances in Neural Information Processing Systems 23*, NIPS '10, pages 1459–1467.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the 29th national conference on Artificial intelligence*, pages 2418–2424.

J. Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Julien Mairal, Rodolphe Jenatton, Francis Bach, and Guillaume Obozinski. 2010. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems 23*, NIPS '10, pages 1558–1566.

Fragkiskos D. Malliaros and Konstantinos Skianis. 2015. Graph-based term weighting for text categorization. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1473–1479.

G.J. McLachlan and K.E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP '04, pages 404–411.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, ICLR '13.

Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. 1997. An Analysis of Statistical and Syntactic Phrases. In *Proceedings of the 5th International Conference on Computer-Assisted Information Retrieval*, volume 97 of *RIAO '97*, pages 200–214.

Bo Pang and Lilian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, pages 271–278.

M. J. D. Powell. 1969. A method for nonlinear constraints in minimization problems. *R. Fletcher editor, Optimization*, pages 283—-298.

Rajat Raina, Andrew Y. Ng, and Daphne Koller. 2006. Constructing Informative Priors Using Transfer Learning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 713–720.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Information and knowledge management*, CIKM '13, pages 59–68.

Ted Sandler, John Blitzer, Partha P. Talukdar, and Lyle H. Ungar. 2009. Regularized learning with networks of features. In *Advances in Neural Information Processing Systems 22*, NIPS '09, pages 1401–1408.

Mark W. Schmidt and Kevin Murphy. 2010. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, AISTATS '10, pages 709–716. JMLR Workshop and Conference Proceedings.

Mark W. Schmidt, Glenn Fung, and Rómer Rosales. 2007. Fast optimization methods for L1 regularization: A comparative study and two new approaches. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pages 286–297.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 327–335.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Vladimir Naumovich Vapnik. 1991. Principles of Risk Minimization for Learning Theory. In *Advances in Neural Information Processing Systems 4*, NIPS '91, pages 831–838.

Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. 2013. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):43:1–43:35.

Dani. Yogatama and Noah A. Smith. 2014a. Linguistic structured sparsity in text categorization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL '14, pages 786–796.

Dani Yogatama and Noah A. Smith. 2014b. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *ICML '14*, pages 656–664.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68(1):49–67.

Lei Yuan, Jun Liu, and Jieping Ye. 2011. Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems 24*, NIPS '11, pages 352–360.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 31–40.

Deep Reinforcement Learning with a Combinatorial Action Space for Predicting Popular Reddit Threads

Ji He[*], **Mari Ostendorf**[*], **Xiaodong He**[†], **Jianshu Chen**[†], **Jianfeng Gao**[†], **Lihong Li**[†], **Li Deng**[†]
[*]Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA
{jvking, ostendor}@uw.edu
[†]Microsoft Research, Redmond, WA 98052, USA
{xiaohe, jianshuc, jfgao, lihongli, deng}@microsoft.com

Abstract

We introduce an online popularity prediction and tracking task as a benchmark task for reinforcement learning with a combinatorial, natural language action space. A specified number of discussion threads predicted to be popular are recommended, chosen from a fixed window of recent comments to track. Novel deep reinforcement learning architectures are studied for effective modeling of the value function associated with actions comprised of *interdependent* sub-actions. The proposed model, which represents dependence between sub-actions through a bi-directional LSTM, gives the best performance across different experimental configurations and domains, and it also generalizes well with varying numbers of recommendation requests.

1 Introduction

This paper is concerned with learning policies for sequential decision-making tasks, where a system takes actions given options characterized by natural language with the goal of maximizing a long-term reward. More specifically, we consider tasks with a combinatorial action space, where each action is a set of multiple interdependent sub-actions. The problem of a combinatorial natural language action space arises in many applications. For example, in real-time news feed recommendation, a user may want to read diverse topics of interest, and an action (i.e. recommendation) from the computer agent would consist of a set of news articles that are not all similar in topics (Yue and Guestrin, 2011). In advertisement placement, an action is a selection of several ads to display, and bundling with complementary products might receive higher click-through-rate than displaying all similar popular products.

In this work, we consider Reddit popularity prediction, which is similar to newsfeed recommendation but different in two respects. First, our goal is not to make recommendations based on an individual's preferences, but instead based on the anticipated long-term interest level of a broad group of readers from a target community. Second, we try to predict rather than detect popularity. Unlike individual interests, community interest level is not often immediately clear; there is a time lag before the level of interest starts to take off. Here, the goal is for the recommendation system to identify and track written documents (e.g. news articles, comments in discussion forum threads, or scientific articles) in real time – attempting to identify hot updates before they become hot to keep the reader at the leading edge. The premise is that the user's bandwidth is limited, and only a limited number of things can be recommended out of several possibilities. In our experimental work, we use discussion forum text, where the recommendations correspond to recent posts or comments, assessing interest based on community response as observed in "likes" or other positive reactions to those comments. For training purposes, we can use community response measured at a time much later than the original post or publication. This problem is well-suited to the reinforcement learning paradigm, since the reward (the level of community uptake or positive response) is not immediately known, so the system needs to learn a mechanism for estimating future reactions. Different from

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1838–1848,
Austin, Texas, November 1-5, 2016.

typical reinforcement learning, the action space is combinatorial since an action corresponds to a set of comments (sub-actions) chosen from a larger set of candidates. A sub-action is a written comment (or document, for another variant of this task).

Two challenges associated with this problem include the potentially high computational complexity of the combinatorial action space and the development of a framework for estimating the long-term reward (the Q-value in reinforcement learning) from a combination of sub-actions characterized by natural language. Here, we focus on the second problem, exploring different deep neural network architectures in an effort to efficiently account for the potential redundancy and/or temporal dependency of different sub-actions in relation to the state space. We sidestep the computational complexity issue (for now) by working with a task where the number of combinations is not too large and by further reducing costs by random sampling.

There are two main contributions in this paper. First, we propose a novel reinforcement learning task with both states and combinatorial actions defined by natural language,[1] which is introduced in section 2. This task, which is based on comment popularity prediction using data from the Reddit discussion forum, can serve as a benchmark in social media recommendation and trend spotting. The second contribution is the development of a novel deep reinforcement learning architecture for handling a combinatorial action space associated with natural language. Prior work related to both the task and deep reinforcement learning is reviewed in section 3, Details for the new models and baseline architectures are described in section 4. Experimental results in section 5 show the proposed methods outperform baseline models and that a bidirectional LSTM is effective for characterizing the combined utility of sub-actions. A brief summary of findings and open questions are in section 6.

2 Popularity Prediction and Tracking

Our experiments are based on Reddit[2], one of the world's largest public discussion forums. On Reddit, registered users initiate a post and people respond with comments, either to the original post or one of its associated comments. Together, the comments and the original post form a discussion tree, which grows as new comments are contributed. It has been show that discussions tend to have a hierarchical topic structure (Weninger et al., 2013), i.e. different branches of the discussion reflect narrowing of higher level topics. Reddit discussions are grouped into different domains, called subreddits, according to different topics or themes. Depending on the popularity of the subreddit, a post can receive hundreds of comments.

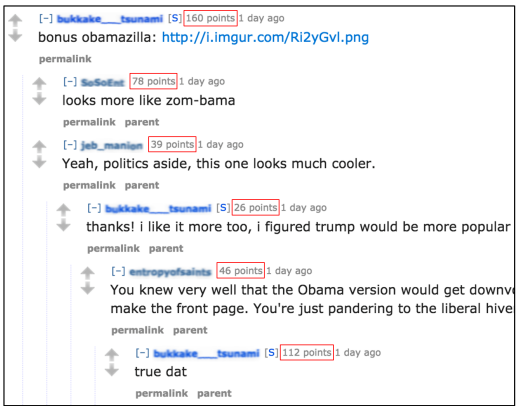


Figure 1: A snapshot of the top of a Reddit discussion tree, where karma scores are shown in red boxes.

Comments (and posts) are associated with positive and negative votes (i.e., likes and dislikes) from registered users that are combined to get a *karma score*, which can be used as a measure for popularity. An example of the top of a Reddit discussion tree is given in Figure 1. The scores in red boxes mark the current karma (popularity) of each comment, and it is quite common that a lower karma comment (e.g. "Yeah, politics aside, this one looks much cooler", compared to "looks more like zom-bama") will lead to more children and popular comments in the future (e.g. "true dat"). Note that the karma scores are dynamic, changing as readers react to the evolving discussion and eventually settling down as the discussion trails off. In a real-time comment recommendation system, the eventual karma of a comment is not immediately available, so prediction of popularity is based on the text in the comment in the context of prior comments in the subtree and other comments in the current time window.

[1] Simulator code and Reddit discussion identifiers are released at `https://github.com/jvking/reddit-RL-simulator`

[2] `http://www.reddit.com`

Popularity prediction and tracking in the Reddit setting is used in this paper for studying reinforcement learning to model long-term rewards in a combinatorial action space. At each time step, the state corresponds to the collection of comments previously recommended. The system aims at automatically picking a few lines of the discussion to follow from the new set of comments in a given window, which is a combinatorial action. Thread popularity tracking can be thought of as a proxy task for news or scientific article recommendation. It has the advantages that "documents" (comments) are relatively short and that the long-term reward can be characterized by Reddit voting scores, which makes this task easier to work with for algorithm development than these larger related tasks.

In this work, we only consider new comments associated with the threads of the discussion that we are currently following to limit the number of possible sub-actions at each time step and with the assumption that prior context is needed to interpret the comments. In other words, the new recommendation should focus on comments that are in the subtrees of previously recommended comments. (A variant relaxing this restriction is suggested in the conclusion section.) Typically, one would expect some interdependencies between comments made in the same window if they fall under the same subtree, because they correspond to a reply to the same parent. In addition, there may be some temporal dependency, since one sub-action may be a comment on the other. These dependencies will affect the combined utility of the sub-actions.

According to our experiments, the performance is significantly worse when we learn a myopic policy compared to reinforcement learning with the same feature set. This shows that long-term dependency indeed matters, as illustrated in Figure 1. This serves as a justification that reinforcement learning is an appropriate approach for modeling popularity of a discussion thread.

3 Related Work

There is a large body of work on reinforcement learning. Among those of most interest here are deep reinforcement learning methods that leverage neural networks because of their success in handling large discrete state/action spaces. Early work such as TD-gammon used a neural network to approximate the state value function (Tesauro, 1995). Recent advances in deep learning (LeCun et al., 2015; Deng and Yu, 2014; Hinton et al., 2012; Krizhevsky et al., 2012; Sordoni et al., 2015) inspired significant progress by combining deep learning with reinforcement learning (Mnih et al., 2015; Silver et al., 2016; Lillicrap et al., 2016; Duan et al., 2016). In natural language processing, reinforcement learning has been applied successfully to dialogue systems that generate natural language and converse with a human user (Scheffler and Young, 2002; Singh et al., 1999; Wen et al., 2016). There has also been interest in mapping text instructions to sequences of executable actions and extracting textual knowledge to improve game control performance (Branavan et al., 2009; Branavan et al., 2011).

Recently, Narasimhan et al. (2015) studied the task of text-based games with a deep Q-learning framework. He et al. (2016) proposed to use a separate deep network for handling natural language actions and to model Q-values via state-action interaction. Nogueira and Cho (2016) have also proposed a goal-driven web navigation task for language-based sequential decision making. Narasimhan et al. (2016) applied reinforcement learning for acquiring and incorporating external evidence to improve information extraction accuracy. The study that we present with Reddit popularity tracking differs from these other text-based reinforcement learning tasks in that the language in both state and action spaces is unconstrained and quite rich.

Dulac-Arnold et al. (2016) also investigated a problem of large discrete action spaces. A Wolpertinger architecture is proposed to reduce computational complexity of evaluating all actions. While a combinatorial action space can be large and discrete, their method does not directly apply in our case, because the possible actions are changing over different states. In addition, our work differs in that its focus is on modeling the combined action-value function rather than on reducing computational complexity. Other work that targets a structured action space includes: an actor-critic algorithm, where actions can have real-valued parameters (Hausknecht and Stone, 2016); and the factored Markov Decision Process (MDP) (Guestrin et al., 2001; Sallans and

Hinton, 2004), with certain independence assumptions between a next-state component and a sub-action. As for a bandits setting, Yue and Guestrin (2011) considered diversification of multi-item recommendation, but their methodology is limited to using linear approximation with hand-crafted features.

The task explored in our paper – detecting and tracking popular threads in a discussion – is somewhat related to topic detection and tracking (Allan, 2012; Mathioudakis and Koudas, 2010), but it differs in that the goal is not to track topics based on frequency, but rather based on reader response. Thus, our work is more closely related to popularity prediction for social media and online news. These studies have explored a variety of definitions (or measurements) of popularity, including: the volume of comments in response to blog posts (Yano and Smith, 2010) and news articles (Tasgkias et al., 2009; Tatar et al., 2011), the number of Twitter shares of news articles (Bandari et al., 2012), the number of reshares on Facebook (Cheng et al., 2014) and retweets on Twitter (Suh et al., 2010; Hong et al., 2011; Tan et al., 2014; Zhao et al., 2015), the rate of posts related to a source rumor (Lukasik et al., 2015), and the difference in the number of reader up and down votes on posts and comments in Reddit discussion forums (Lakkaraju et al., 2013; Jaech et al., 2015). An advantage of working with the Reddit data is that both positive and negative reactions are accounted for in the karma score. Of the prior work on Reddit, the task explored here is most similar to (Jaech et al., 2015) in that it involves choosing relatively high karma comments (or threads) from a time-limited set rather than directly predicting comment (or post) karma. Prior work on popularity prediction used supervised learning; this is the first work that frames tracking hot topics in social media with deep reinforcement learning.

4 Characterizing a combinatorial action space

4.1 Notation

In this sequential decision making problem, at each time step t, the agent receives a text string that describes the state $s_t \in \mathcal{S}$ (i.e., "state-text") and picks a text string that describes the action $a_t \in \mathcal{A}$ (i.e., "action-text"), where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. Here, we assume a_t is chosen from a set of given candidates. In our case both $\mathcal{S}$ and $\mathcal{A}$ are described by natural language. Given the state-text and action-texts, the agent aims to select the best action in order to maximize its long-term reward. Then the environment state is updated $s_{t+1} = s'$ according to a probability $p(s'|s, a)$, and the agent receives a reward r_{t+1} for that particular transition. We define the action-value function (i.e. the Q-function) $Q(s, a)$ as the expected return starting from s and taking the action a:

$$Q(s,a) = \mathbb{E}\left\{\sum_{l=0}^{+\infty} \gamma^l r_{t+1+l} | s_t = s, a_t = a\right\}$$

where $\gamma \in (0, 1)$ denotes a discount factor. The Q-function associated with an optimal policy can be found by the Q-learning algorithm (Watkins and Dayan, 1992):

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta_t \cdot \big(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t)\big)$$

where η_t is a learning rate parameter.

The set of comments that are being tracked at time step t is denoted as M_t. All previously tracked comments, as well as the post (root node of the tree), is considered as state s_t ($s_t = \{M_0, M_1, \cdots, M_t\}$), and we initialize $s_0 = M_0$ to be the post. An action is taken when a total of N new comments $\{c_{t,1}, c_{t,2}, \cdots, c_{t,N}\}$ appear as nodes in the subtree of M_t, and the agent picks a set of K comments to be tracked in the next time step $t+1$. Thus we have:

$$a_t = \{c_t^1, c_t^2, \cdots, c_t^K\}, \quad c_t^i \in \{c_{t,1}, c_{t,2}, \cdots, c_{t,N}\} \text{ and } c_t^i \neq c_t^j \text{ if } i \neq j \quad (1)$$

and $M_{t+1} = a_t$. At the same time, by taking action a_t at state s_t, the reward r_{t+1} is the accumulated karma scores, i.e. sum over all comments in M_{t+1}. Note that the reward signal is used in online training, while at model deployment (testing stage), the scores are only used as an evaluation metric.

Following the reinforcement learning tradition, we call tracking of a single discussion tree from start (root node post) to end (no more new comments appear) an *episode*. We also randomly partition all

discussion trees into separate training and testing sets, so that texts seen by the agent in training and testing are from the same domain but different discussions. For each episode, depending on whether training/testing, the simulator randomly picks a discussion tree, and presents the agent with the current state and N new comments.

4.2 Q-function alternatives

With the real-time setting, it is clear that action a_t will affect the next state s_{t+1} and furthermore the future expected reward. The action a_t consists of K comments (sub-actions), making modeling Q-values $Q(s_t, a_t)$ difficult. To handle a large state space, Mnih et al. (2015) proposed a Deep Q-Network (DQN). In case of a large action space, we may use both state and action representations as input to a deep neural network. It is shown that the Deep Reinforcement Relevance Network (DRRN, Figure 2(b)), i.e. two separate deep neural networks for modeling state embedding and action embedding, performs better than per-action DQN (PA-DQN in Figure 2(a)), as well as other DQN variants for dealing with natural language action spaces (He et al., 2016).

Our baseline models include Linear, PA-DQN and DRRN. We concatenate the K sub-actions/comments to form the action representation. The Linear and PA-DQN (Figure 2(a)) take as input a concatenation of state and action representations, and model a single Q-value $Q(s_t, a_t)$ using linear or DNN function approximations. The DRRN consists of a pair of DNNs, one for the state-text embedding and the other for action-text embeddings, which are then used to compute $Q(s_t, a_t)$ via a pairwise interaction function (Figure 2(b)).

One simple alternative approach by utilizing this combinatorial structure is to compute an embedding for each sub-action c_t^i. We can then model the value in picking a particular sub-action, $Q(s_t, c_t^i)$, through a pairwise interaction between the state and this sub-action. $Q(s_t, c_t^i)$ represents the expected accumulated future rewards by including this sub-action. The agent then greedily picks the top-K sub-actions with highest values to achieve the highest $Q(s_t, a_t)$. In this approach, we are assuming the long-term rewards associated with sub-actions are independent of each other. More specifically, greedily picking the top-K sub-actions is equivalent to maximizing the following action-value function:

$$Q(s_t, a_t) = \sum_{i=1}^{K} Q(s_t, c_t^i) \tag{2}$$

while satisfying (1). We call this proposed method DRRN-Sum, and its architecture is shown in Figure 2(c). Similarly as in DRRN, we use two networks to embed state and actions separately. However, for different sub-actions, we keep the network parameters tied. We also use the same top layer dimension and the same pairwise interaction function for all sub-actions.

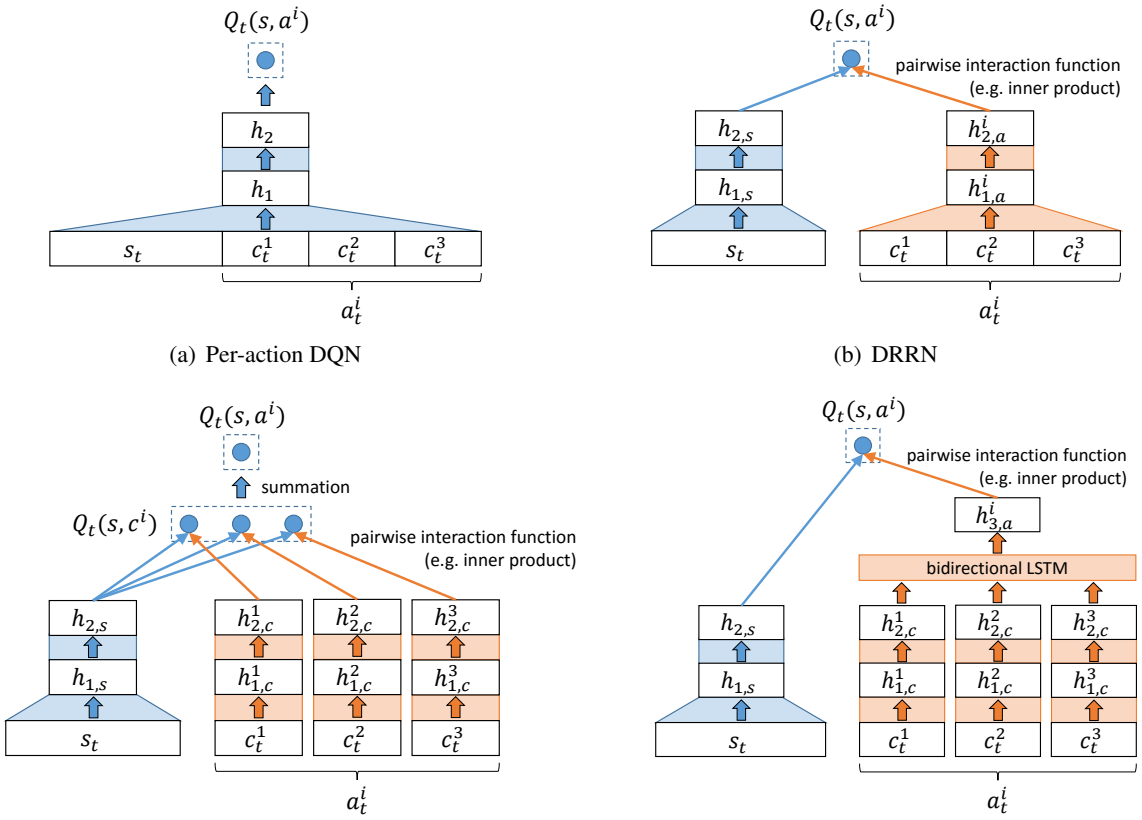
In the case of a linear additive interaction, such as an inner product or bilinear operation, Equation (2) is equivalent to computing the interaction between the state embedding and an action embedding, where the action embedding is obtained linearly by summing over K sub-action embeddings. When sub-actions have strong correlation, this independence assumption is invalid and can result in a poor estimation of $Q(s_t, a_t)$. For example, most people are interested in the total information stored in the combined action a_t. Due to content redundancy in the sub-actions $c_t^1, c_t^2, \cdots, c_t^K$, we expect $Q(s_t, a_t)$ to be smaller than $\sum_i Q(s_t, c_t^i)$.

To come up with a general model for handling a combinatorial action-value function, we further propose the DRRN-BiLSTM (Figure 2(d)). In this architecture, we use a DNN to generate an embedding for each comment. Then a Bidirectional Long Short-Term Memory (Graves and Schmidhuber, 2005) is used to combine a sequence of K comment embeddings. As the Bidirectional LSTM has a larger capacity due to its nonlinear structure, we expect it will capture more details on how the embeddings for the sub-actions combine into an action embedding. Note that both of our proposed methods (DRRN-Sum and DRRN-BiLSTM) can handle a varying value of K, while for the DQN and DRRN baselines, we need to use a fixed K in training and testing.

5 Experiments

5.1 Datasets and Experimental Configurations

Our data consists of 5 subreddits (askscience, askmen, todayilearned, worldnews, nfl) with diverse



(a) Per-action DQN (b) DRRN

(c) DRRN-Sum (d) DRRN-BiLSTM

Figure 2: Different deep Q-learning architectures

topics and genres. In our experiments, in order to have long enough discussion threads, we filter out discussion trees with fewer than 100 comments. For each subreddit, we randomly partition 90% of the data for online training, and 10% of the data for testing (deployment). The basic subreddit statistics are shown in Table 1. We report the random policy performances and heuristic upper bound performances (averaged over 10,000 episodes) in Table 2 and Table 3.[3] The upper bound performances are obtained using stabilized karma scores and offline constructed tree structure. The mean and standard deviation are obtained by 5 independent runs.

In all our experiments we set $N = 10$. Explicitly representing all N-choose-K actions requires a lot of memory and does not scale up. We therefore use a variant of Q-learning: when taking the max over

Subreddit	# Posts (in k)	# Comments (in M)
askscience	0.94	0.32
askmen	4.45	1.06
todayilearned	9.44	5.11
worldnews	9.88	5.99
nfl	11.73	6.12

Table 1: Basic statistics of filtered subreddit data sets

Subreddit	Random	Upper bound
askscience	321.3 (7.0)	2109.0 (16.5)
askmen	132.4 (0.7)	651.4 (2.8)
todayilearned	390.3 (5.7)	2679.6 (30.1)
worldnews	205.8 (4.5)	1853.4 (44.4)
nfl	237.1 (1.4)	1338.2 (13.2)

Table 2: Mean and standard deviation of random and upper-bound performance (with $N = 10, K = 3$) across different subreddits.

[3]Upper bounds are estimated by greedily searching through each discussion tree to find K max karma discussion threads (overlapped comments are counted only once). This upper bound may not be attainable in a real-time setting.

K	Random	Upper bound
2	201.0 (2.1)	1991.3 (2.9)
3	321.3 (7.0)	2109.0 (16.5)
4	447.1 (10.8)	2206.6 (8.2)
5	561.3 (18.8)	2298.0 (29.1)

Table 3: Mean and standard deviation of random and upper-bound performance on askscience, with $N = 10$ and $K = 2, 3, 4, 5$.

possible next-actions, we instead randomly subsample m' actions and take the max over them. We set $m' = 10$ throughout our experiments. This heuristic technique works well in our experiments.

For text preprocessing we remove punctuation and lowercase capital letters. For each state s_t and comment c_t^i, we use a bag-of-words representation with the same vocabulary in all networks. The vocabulary contains the most frequent 5,000 words; the out-of-vocabulary rate is 7.1%.

In terms of the Q-learning agent, fully-connected neural networks are used for text embeddings. The network has $L = 2$ hidden layers, each with 20 nodes, and model parameters are initialized with small random numbers. ϵ-greedy is used for exploration-exploitation, and we keep $\epsilon = 0.1$ throughout online training and testing. We pick the discount factor $\gamma = 0.9$. During online training, we use experience replay (Lin, 1992) and the memory size is set to 10,000 tuples of $(s_t, a_t, r_{t+1}, s_{t+1})$. For each experience replay, 500 episodes are generated and stored in a first-in-first-out fashion, and multiple epochs are trained for each model. Minibatch stochastic gradient descent is implemented with a batch size of 100. The learning rate is kept constant: $\eta_t = 0.000001$.

The proposed methods are compared with three baseline models: Linear, per-action DQN (PA-DQN), and DRRN. For both Linear and PA-DQN, the state and comments are concatenated as an input. For the DRRN, the state and comments are sent through two separate deep neural networks. However, in our baselines, we do not explicitly model how values associated with each comment are combined to form the action value. For the DRRN baseline and proposed methods (DRRN-Sum and DRRN-BiLSTM), we use an inner product as the pairwise interaction function.

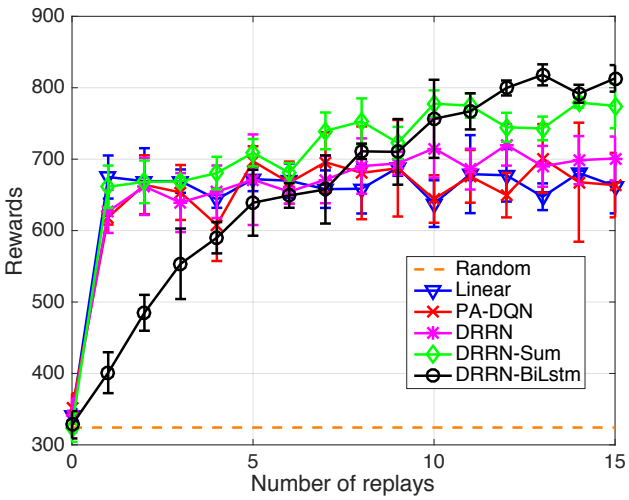


Figure 3: Learning curves of baselines and proposed methods on "askscience"

5.2 Experimental Results

In Figure 3 we provide learning curves of different models on the askscience subreddit during online learning. In this experiment, we set $N = 10, K = 3$. Each curve is obtained by averaging over 3 independent runs, and the error bars are also shown. All models start with random performance, and converge after approximately 15 experience replays. The DRRN-Sum converges as fast as baseline models, with better converged performance. DRRN-BiLSTM converges slower than other methods, but with the best converged performance.

After we train all the models on the training set, we fix the model parameters and apply (deploy) on the test set, where the models predict which action to take but no reward is shown until evaluation. The test performance is averaged over 1000 episodes, and we report mean and standard deviation over 5 independent runs.

On askscience, we try multiple settings with $N = 10$, $K = 2, 3, 4, 5$ and the results are shown in Table 4. Both DRRN-Sum and DRRN-BiLSTM consistently outperform baseline methods. The DRRN-BiLSTM performs better with larger K, probably due to the greater chance of redundancy in combining more sub-actions.

We also perform online training and test across different subreddits. With $N = 10, K = 3$, the test performance gains over the linear baseline are shown in Figure 4. Again, the test performance is

K	Linear	PA-DQN	DRRN	DRRN-Sum	DRRN-BiLSTM
2	553.3 (2.8)	556.8 (14.5)	553.0 (17.5)	569.6 (18.4)	**573.2 (12.9)**
3	656.2 (22.5)	668.3 (19.9)	694.9 (15.5)	704.3 (20.1)	**711.1 (8.7)**
4	812.5 (23.4)	818.0 (29.9)	828.2 (27.5)	829.9 (13.2)	**854.7 (16.0)**
5	861.6 (28.3)	884.3 (11.4)	921.8 (10.7)	942.3 (19.1)	**980.9 (21.1)**

Table 4: On askscience, average karma scores and standard deviation of baselines and proposed methods (with $N = 10$)

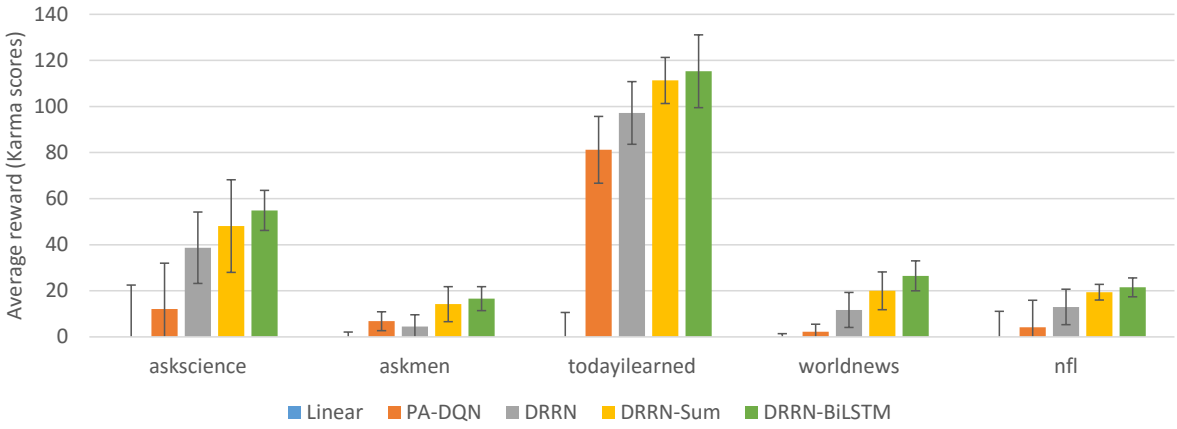


Figure 4: Average karma score gains over the linear baseline and standard deviation across different subreddits (with $N = 10$, $K = 3$).

K	DRRN-Sum	DRRN-BiLSTM
2	538.5 (18.9)	**551.2 (10.5)**
4	819.1 (14.7)	**829.9 (11.1)**
5	921.6 (15.6)	**951.3 (15.7)**

Table 5: On askscience, average karma scores and standard deviation of proposed methods trained with $K = 3$ and test with different K's

averaged over 1000 episodes, and we report mean and standard deviation over 5 independent runs. The findings are consistent with those for askscience. Since different subreddits may have very different karma scores distributions and language style, this suggests the algorithms apply to different text genres.

In actual model deployment, a possible scenario is that users may have different requests. For example, a user may ask the agent to provide $K = 2$ discussion threads on one day, due to limited reading time, and ask the agent to provide $K = 5$ discussion threads on the other day. For the baseline models (Linear, PA-DQN, DRRN), we will need to train separate models for different K's. The proposed methods (DRRN-Sum and DRRN-BiLSTM), on the other hand, can easily handle a varying K. To test whether the performance indeed generalizes well, we train proposed models on askscience with $N = 10, K = 3$ and test them with $N = 10, K \in 2, 4, 5$, as shown in Table 5. Compared to the proposed models that are specifically trained for these K's (Table 4), the generalized test performance indeed degrades, as expected. However, in many cases, our proposed methods still outperform all three baselines (Linear, PA-DQN and DRRN) that are trained specifically for these K's. This shows that the proposed methods can generalize to varying K's even if it is trained on a particular value of K.

In Table 6, we show an anecdotal example with state and sub-actions. The two sub-actions are strongly correlated and have redundant information. By combining the second sub-action compared to choosing just the first sub-action alone, DRRN-Sum and DRRN-BiLSTM predict 86% and 26% relative increase in action-value, respectively. Since these two sub-actions are highly redundant, we hypothesize DRRN-BiLSTM is better than DRRN-Sum at capturing interdependency between sub-actions.

6 Conclusion

In this paper we introduce a new reinforcement learning task associated with predicting and tracking popular threads on Reddit. The states and actions

State text (partially shown)
Are there any cosmological phenomena that we strongly suspect will occur, but the universe just isn't old enough for them to have happened yet?
Comments (sub-actions) (partially shown)
[1] White dwarf stars will eventually stop emitting light and become black dwarfs. [2] Yes, there are quite a few, such as: White dwarfs will cool down to black dwarfs.

Table 6: An example state and its sub-actions

are all described by natural language so the task is useful for language studies. We then develop novel deep Q-learning architectures to better model the state-action value function with a combinatorial action space. The proposed DRRN-BiLSTM method not only performs better across different experimental configurations and domains, but it also generalizes well for scenarios where the user can request changes in the number tracked.

This work represents a first step towards addressing the popularity prediction and tracking problem. While performance of the system beats several baselines, it still falls far short of the oracle result. Prior work has shown that timing is an important factor in predicting popularity (Lampe and Resnick, 2004; Jaech et al., 2015), and all the proposed models would benefit from incorporating this information. Another variant might consider short-term reactions to a comment, if any, in the update window. It would also be of interest to explore implementations of backtracking in the sub-action space (incurring a cost), in order to recommend comments that were not selected earlier but have become highly popular. Lastly, it will be important to study principled solutions for handling the computational complexity of the combinatorial action space.

References

J. Allan. 2012. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media.

Roja Bandari, Sitaram Asur, and Bernardo Huberman. 2012. The pulse of news in social media: forecasting popularity. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP*, pages 82–90, August.

S.R.K. Branavan, D. Silver, and R. Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Proc. of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277. Association for Computational Linguistics.

Justin Cheng, Lada Adamic, P. Alex Dow, Jon Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted? In *Proc. Int. Conf. World Wide Web (WWW)*.

L. Deng and D. Yu. 2014. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, and J. Hunt. 2016. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

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

C. Guestrin, D. Koller, and R. Parr. 2001. Multiagent planning with factored MDPs. In *NIPS*, volume 1, pages 1523–1530.

M. Hausknecht and P. Stone. 2016. Deep reinforcement learning in parameterized action space. In *International Conference on Learning Representations*.

J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97.

Liangjie Hong, Ovidiu Dan, and Brian Davison. 2011. Predicting popular messages in Twitter. In *Proc. Int. Conf. World Wide Web (WWW)*, pages 57–58.

A. Jaech, V. Zayats, H. Fang, M. Ostendorf, and H. Hajishirzi. 2015. Talking to the crowd: What do people react to in online discussions? In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 2026–2031, September.

A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105.

Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. 2013. What's in a name? Understanding the interplay between titles, content, and communities in social media. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

C. Lampe and P. Resnick. 2004. Slash(dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 543–550.

Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

T. P Lillicrap, J. J Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

L-J Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

M. Mathioudakis and N. Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G Bellemare, A. Graves, M. Riedmiller, A. K Fidjeland, G. Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

K. Narasimhan, T. Kulkarni, and R. Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, September.

K. Narasimhan, A. Yala, and R. Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv:1603.07954*.

R. Nogueira and K. Cho. 2016. Webnav: A new large-scale task for natural language based sequential decision making. *arXiv preprint arXiv:1602.02261*.

B. Sallans and G. E Hinton. 2004. Reinforcement learning with factored states and actions. *The Journal of Machine Learning Research*, 5:1063–1088.

K. Scheffler and S. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of the second International Conference on Human Language Technology Research*, pages 12–19.

D. Silver, A. Huang, C. J Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

S. P Singh, M. J Kearns, D. J Litman, and M. A Walker. 1999. Reinforcement learning for spoken dialogue systems. In *NIPS*, pages 956–962.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT 2015*.

B. Suh, L. Hong, P. Pirolli, and E. H. Chi. 2010. Want to be retweeted? Large scale analytics on factors impacting retweet in twitter network. In *Proc. IEEE Inter. Conf. on Social Computing (SocialCom)*, pages 177–184.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 175–186.

Manos Tasgkias, Wouter Weerkamp, and Maarten de Rijke. 2009. Predicting the volume of comments on online news stories. In *Proc. CIKM*, pages 1765–1768.

Alexandru Tatar, Jeremie Leguay, Panayotis Antoniadis, Arnaud Limbourg, Marcelo Dias de Amorim, and Serge Fdida. 2011. Predicting the polularity of online articles based on user comments. In *Proc. Inter. Conf. on Web Intelligence, Mining and Semantics (WIMS)*, pages 67:1–67:8.

G. Tesauro. 1995. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68.

C. JCH Watkins and P. Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

T.-H. Wen, M. Gasic, N. Mrksic, L. M Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, and S. Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

T. Weninger, X. A. Zhu, and J. Han. 2013. An exploration of discussion threads in social news sites: A case study of the reddit community. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 579–583. IEEE.

Tae Yano and Noah A. Smith. 2010. What's worthy of comment? Content and comment volume in political blogs. In *Proc. Int. AAAI Conf. Weblogs and Social Media (ICWSM)*.

Y. Yue and C. Guestrin. 2011. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, pages 2483–2491.

Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. 2015. SEISMIC: A self-exciting point process model for predicting Tweet popularity. In *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining*.

Non-Literal Text Reuse in Historical Texts: An Approach to Identify Reuse Transformations and its Application to Bible Reuse

Maria Moritz[1], Andreas Wiederhold[2], Barbara Pavlek[3], Yuri Bizzoni[4], and **Marco Büchler[1]**
[1]Institute of Computer Science, University of Göttingen
[2]Institute for Educational Science, University of Göttingen
[3]Minds and Traditions Research Group, Max Planck Institute for the Science of Human History, Jena
[4]Department of Philosophy, Linguistics, Theory of Science, University of Gothenburg

Abstract

Text reuse refers to citing, copying or alluding text excerpts from a text resource to a new context. While detecting reuse in contemporary languages is well supported—given extensive research, techniques, and corpora—automatically detecting historical text reuse is much more difficult. Corpora of historical languages are less documented and often encompass various genres, linguistic varieties, and topics. In fact, historical text reuse detection is much less understood and empirical studies are necessary to enable and improve its automation. We present a linguistic analysis of text reuse in two ancient data sets. We contribute an automated approach to analyze how an original text was transformed into its reuse, taking linguistic resources into account to understand how they help characterizing the transformation. It is complemented by a manual analysis of a subset of the reuse. Our results show the limitations of approaches focusing on literal reuse detection. Yet, linguistic resources can effectively support understanding the non-literal text reuse transformation process. Our results support practitioners and researchers working on understanding and detecting historical reuse.

1 Introduction

The computational detection of historical text reuse—including citations, quotations or allusions —can be applied in many respects. It can help tracing down historical content (a.k.a., *lines of transmission*), which is essential to the field of textual criticism (Büchler et al., 2012). In the context of massive digitization projects, it can identify relationships between text excerpts referring to the same source. Specifically, detecting copies of the same historical text that have diverged over time (manuscript studies, a.k.a., *Stemma Codicum*) is an important task.

Although much work exists in the field of natural language processing (NLP), many new challenges arise when processing historical text. The most important challenges are the absence of supporting tools and methods, including an agreement on a common orthography, standardization of variants, and a wide range of clean, digitized text (Piotrowski, 2012; Geyken and Gloning, 2014; Zitouni, 2014). Typical statistical approaches from the field of NLP are difficult to apply to historically transferred texts, since these often cover a large timespan and, thus, comprise many different writing styles, text variants or even reuse styles (Büchler, 2013). Our long-term goal is to conceive robust text reuse detection techniques for historical texts. To this end, we need to improve the quantitative empirical understanding of such reuse accompanied by qualitative empirical studies. However, only few such works exist.

We study less- and non-literal text reuse of Bible verses in Ancient Greek and Latin texts. Our focus is on understanding how the reuse instances are transformed from the original verses. We identify operations that characterize how words are changed—e.g., synonymized, capitalized or part-of-speech (PoS) information changed. Since our approach uses external linguistic resources, including Ancient Greek WordNet (AGWN) (Bizzoni et al., 2014; Minozzi, 2009) and various lemma lists, we also show how such resources can help detecting reuse and where the limitations are. We complement the automated approach

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1849–1859,
Austin, Texas, November 1-5, 2016.

with a qualitative manual analysis. We contribute:

- an automated approach to characterize how text is transformed between reuse and original,
- an application of the approach to two text datasets where reuse was manually identified,
- empirical data based on the automated approach, complemented by a manual identification.

Our resulting datasets[1] with rich information about the reuse transformation (e.g., PoS and morphology changes, and words becoming synonyms or hyperonyms, among others) can be used as a benchmark for future reuse detection and classification approaches.

2 Related Work

We first discuss why existing reuse detection approaches are not applicable to historical texts, and then present works trying to address this problem.

Historical Text Reuse and Plagiarism Detection. Büchler (2013) combines state-of-the-art NLP techniques to address reuse detection scenarios for historical texts, ranging from near copies to text excerpts with a minimum overlap. He uses the commonly used method fingerprinting, which selects n-grams from an upfront pre-segmentized corpus. While his approach can discover historical and modern text reuse language-independently, it requires a minimum text similarity—typically at least two common features.

Recognizing modified reuse is difficult in general. Alzahrani et al. (2012) study plagiarism detection techniques: n-gram-, syntax-, and semantics-based approaches. As soon as reused text is slightly modified (e.g., words changed) most systems fail. Barrón-Cedeño et al. (2013) conduct experiments on paraphrasing, observing that complex paraphrasing along with a high paraphrasing density challenges plagiarism detection, and that lexical substitution is the most frequent technique for plagiarizing. The AraPlagDet (Bensalem et al., 2015) initiative focuses on the evaluation of plagiarism detection methods for Arabic texts. Eight methods were submitted and turned out to work with a high accuracy on external plagiarism detection but did not achieve usable results for intrinsic plagiarism detection.

Corpora. Huge parallel corpora of modern languages are used in fields such as paraphrase generation and detection, typically used to train statistical models (Zhao et al., 2009; Madnani and Dorr, 2010). However, such corpora hardly exist for historical languages or are copyrighted, such as the TLG digital library (Pantelia, 2014). Especially in the field of modern reuse investigation, aligned corpora are often used, providing a rich source of paraphrasal sentence pairs in one, sometimes multiple languages. One of such is the Microsoft Research Paraphrase Corpus (MSRP), which contains 5801 manually evaluated, paraphrasal sentence pairs in English (Dolan and Brockett, 2005). Ganitkevitch et al. (2013) present a paraphrase database with over 200 million English paraphrase pairs and 196 million Spanish paraphrases. Each paraphrase pair comes with measures, such as a paraphrase probability score. In ancient literature, efforts are made to collect Biblical reuse. One of such is the collection of Ancinet Greek and Latin quotations based on the the Vetus Latina series and the Novum Testamentum Graecum Editio Critica Maior (Houghton, 2013a; Houghton, 2013b). It contains more than 150,000 Latin citations and about 87,000 Ancient Greek Bible references.

Historical Text Processing in General. Efforts to automatically process ancient texts are made around the Perseus Digital Library project (Crane, 1985), among others. For example, Bamman (2008) presents the discovery of textual allusions in a collection of Classical poetry, using measures such as token similarity, n-grams or syntactic similarity. This allows finding at least the most similar candidates within a closed library. Some works have focused on text reuse in Biblical Greek text. Lee (2007) investigate reuse among the Gospels of the New Testament, aimed at aligning similar sentences. Using source alternation patterns, among others, the approach uses cosine similarity, source verse proximity, and source verse order. Focusing on high recall, the detection of Homeric quotations in Athenaeus' Deipnosophistai' was investigated by Büchler et al. (2012), searching for distinctive words within reuse.

While the approaches above rely on string or feature similarity, Bamman (2011b) attempts to process the semantic space using word-sense disambiguation (Patwardhan et al., 2003; Agirre and Edmonds, 2007). Using a bilingual sense inventory and training set, they classify up to 72 % of word senses correctly.

Utilizing Linguistic Resources. Word nets support

[1] https://bitbucket.org/mariamoritz/emnlp

identifying word relationships. Jing (1998) investigates issues that come with using WordNet (Miller et al., 1990) for language generation. Among others, these comprise issues arising from the adaption of a general lexicon to a specific domain. These were encountered by using a domain corpus and an ontology to prune WordNet to a certain domain.

In our work, we are interested in using linguistic resources (word nets and lemma lists) together with PoS information to model the transformation process of reuse, specifically on an ancient language text to find limitations when applied to non-literal text reuse.

3 Methodology

Our study addresses two main research questions:
RQ1. *What is the extent of non-literal reuse in our datasets?* This analysis provides a baseline for the following characterizations of the non-literal reuse.
RQ2. *How is the non-literally reused text modified in our datasets?* We study kinds and frequencies of semantic, lexical, and morphological changes. We develop an automated approach to identify the reuse transformation, and complement it with a manual, qualitative analysis. We formulate two sub-questions:
RQ2.1. *How can linguistic resources support the discovery of non-literal reuse?* We conjecture that non-literal reuse is difficult to capture automatically (especially due to domain- or author-specific words), but that taking linguistic resources into account helps. We analyze the coverage of words in lemma lists and a synset database, and investigate how useful they are for understanding the reuse transformations.
RQ2.2. *What are the limitations of an automated classification approach relying on linguistic resources?* Our manual analysis investigates the reuse in its full richness, to understand the limitations of the automated approach and identify further characteristics of the reuse in our datasets.

3.1 Study Design

Our study comprises the following main steps. First (**RQ1**), we identify and characterize the literal and non-literal overlap in reuse instances. Second (towards **RQ2**), we define operations reflecting literal reuse, replacements (inspired by semantic relationships, such as synonyms and hyperonyms, supported by AGWN), and morphological changes (e.g., when mapping words contain the same cognate). Our operations are based on a one-word-replacement to better quantify the results. Third (**RQ2.1**), we develop an algorithm that identifies operations by first looking for morphological changes between a word from the reuse and its corresponding candidate from the Bible verse and, in case of no success, by seeking for a semantic relation. We apply it to our two datasets and investigate the relationships of affected words and the literal share. We quantify occurrences of operations and calculate two measures sup_{lem} (lemma support) and sup_{AGWN} (AGWN support) to assess the resources' coverage for our approach. Fourth (**RQ2.2**), we manually analyze a smaller sample of our reuse datasets, using further operations, to understand the full richness of the reuse.

3.2 Datasets

We use the following two text sources, both reusing content from Bible verses. As a ground truth of the reuse, we use manually annotated versions of both, provided to us by Mellerin (2014) and the Biblindex project (Mellerin, 2016; Vinzent et al., 2013).

Our first dataset comes from the primary source text of "Salvation for the Rich" from the Ancient Greek writer Clement of Alexandria (Clément d'Alexandrie, 2011), a well-known author in Biblical literature (Cosaert, 2008). The Biblindex team annotated 128 text passages as Bible reuse instances, adding a footnote with Bible verse pointers to each. We select a total of 95 out of these 128, following four criteria: (i) reuse should not consist of an exact literal copy of a Bible verse (skipping six instances), (ii) reuse should be recognizable by our expert (skipping ten instances), (iii) the reference frame should be within five Bible verses (comparable with sentences) to avoid too much noise in our data to ensure a comparable length to the original Bible verse (skipping nine instances), and (iv) reuse instances should not exceed a length of 40 tokens (1–2 sentences), again to cut the long tail and avoid too much noise (skipping eight instances). Sometimes one reuse instance pointed to different Bible verses or one text passage contained more than one reuse instance, thus, we come up with 199 verse-reuse-pairs. The excerpts point to a total of 15 Bible books.

Our second dataset are extracts from a total of 14 volumes of twelve works and two work collec-

Jer 23 24	si occultabitur vir in absconditis et ego non videbo eum dicit Dominus numquid non caelum **et terram ego impleo** ait Dominus (*Can anyone hide himself in secret places that I will not see him? Said the lord. Do not I fill heaven and earth? Said the Lord*)
literal	**et terram ego impleo** (*and I fill the earth*)
Mk 10 30	Ἤρξατο λέγειν ὁ Πέτρος αὐτῷ, Ἰδοὺ ἡμεῖς ἀφήκαμεν πάντα καὶ ἠκολουθήκαμέν σοι. (*Peter began to say to him: See, we left everything and followed you.*)
literal	ἡμεῖς ἀφήκαμεν πάντα καὶ ἠκολουθήσαμέν σοι (*we left everything and followed you*)
Prv 18 3	**impius cum in profundum venerit** peccatorum **contemnit** sed sequitur eum ignominia et obprobrium (*When the wicked man is come into the depth of sins, also contempt comes but ignominy and reproach follow him)*
more literal	**Impius** , **cum venerit in profundum** malorum , **contemnit** (*When the wicked man is come into the depth of evil*)
1Cor 13 13	νυνὶ δὲ μένει **πίστις** , **ἐλπίς** , **ἀγάπη** , τὰ τρία ταῦτα μείζων δὲ τούτων ἡ **ἀγάπη** (*And now remain faith, hope, love, these three; but the greatest of those is love.*)
less literal	**πίστει** καὶ **ἐλπίδι** καὶ **ἀγάπῃ** (*faith, and hope, and love* - in dative case)
less literal	**ἀγάπην** , **πίστιν** , **ἐλπίδα** (*love, faith, hope* - in accusative case)
less literal	μένει δὲ τὰ τρία ταῦτα , **πίστις** , **ἐλπίς** , **ἀγάπη** · μείζων δὲ ἐν τούτοις ἡ **ἀγάπη** *(and remain these three, faith, hope, love; but the greatest among them is love)*
Mt 12 35	ὁ ἀγαθὸς ἄνθρωπος ἐκ τοῦ ἀγαθοῦ θησαυροῦ ἐκβάλλει ἀγαθά , καὶ ὁ πονηρὸς ἄνθρωπος ἐκ τοῦ πονηροῦ θησαυροῦ ἐκβάλλει πονηρά . (*A good man out of good storage brings out good things , and an evil man out of the evil storage brings evil things .*)
non-literal	Ψυχῆς , τὰ δὲ ἐκτός , κἂν μὲν ἡ ψυχὴ χρῆται καλῶς , καλὰ καὶ ταῦτα δοκεῖ , ἐὰν δὲ πονηρῶς , πονηρά , ὁ κελεύων ἀπαλλοτριοῦν τὰ ὑπάρχοντα (*[are whitin the] soul, and some are out, and if the soul uses them good, those things are also thought of as good, but if [they are used as] bad, [they are thought of as] bad; he who commands the renouncement of possessions*)

Figure 1: Examples of reuse

tions from the Latin writer Bernard of Clairvaux. We again use Biblindex' extracted Bible reuse, which offers over 1,100 reuse instances in alphabetical order. We follow the same selection criteria as for Greek and—starting top-down and dropping only two—we obtain 162 Bible-verse reuse-pairs, which is similar to the number of Greek reuse instances. Specifically, since those reuse instances come from several different primary source works, they point to a total of 31 Bible books. We use the Bible editions from Biblindex, specifically, the data based on *Septuagint* (Rahlfs, 1935b), *Greek New testament* (Aland and Aland, 1966), and *Biblia sacra juxta vulgatam versionem* (Weber R., 1969 1994 2007).

Fig. 1 shows reuse examples, illustrating the wide range of literalness in our data, comprising literal (all tokens overlap), less literal (important tokens overlap), and non-literal (no content word tokens overlap) reuse. For example, Clement's reuse ranges from introducing the overall topic by citing multiple verses, to supporting his argumentation. Specifically, Mk 10 30 is a fully literal reuse from a passage that discusses the problem of rich men in heaven. Clement uses this episode as a main point in his essay. Later he refers to 1Cor 13 13, he again refers to how hard it would be for rich men to enter heaven, explaining that salvation is independent of "external things," but depends on the "virtue of the soul," mentioning faith, hope, and love, the key words in the original verse.

Algorithm 1: Reuse classification algorithm

```
/* Executed for each reuse instance and its corresponding
   Bible verse.  morph(x) returns the part-of-speech
   and/or case of x.  repl_case and repl_pos are masked to
   repl_morph for clarity reasons.  checkm(x,y) returns
   NOPmorph(morph(x),morph(y)) if morph(x) equals morph(y)
   and repl_morph(morph(x),morph(y)) otherwise.          */
input  : L ← set of word-lemma pairs obtained from the lemma resources
input  : S ← set of synsets from AGWN; each synset contains an id and a parent id
input  : T ← list of words of reuse instance (containing part-of-speech information)
input  : B ← list of words of Bible verse (containing part-of-speech information)
output : OP ← list of sets containing up to 3 parameterized operations
s1, s2 ← any two synsets ∈ S.
tmp_op ← temporary variable which presents the absence of a relation but not of a
lemma.
for t in T do
  for b in B do
    if t=b then
      OP ← OP ∪ (NOP(t, b), checkm(morph(t), morph(b)))
      break
    else if lowerCase(t) = b then
      OP ← OP ∪ (lower(t, b), checkm(morph(t), morph(b)))
      break
    else if lowerCase(b) = t then
      OP ← OP ∪ (upper(t, b), checkm(morph(t), morph(b)))
      break
    else if t ∈ L and b ∈ L then
      /* lemma found for original (b) and reuse word (t)  */
      if lemma(t) = lemma(b) then
        OP ← OP ∪ (lem(t, b), checkm(morph(t), morph(b)))
        break
      else if t ∈ s1 and b ∈ s2 and s1 ∈ S and s2 ∈ S then
        if s1 = s2 then
          /* t is synonym of b                          */
          OP ← OP ∪ (repl_syn(t, b)) break
        else if id(s1) = parent_id(s2) then
          /* t is hyperonym of b                        */
          OP ← OP ∪ (repl_hypo(t, b)) break
        else if parent_id(s1) = id(s2) then
          /* t is hyperonym of b                        */
          OP ← OP ∪ (hyper(t, b)) break
        else if parent_id(s1) = parent_id(s2) then
          /* synset of t and synset of b both have the same
             synset as parent                           */
          OP ← OP ∪ (repl_cohypo(t, b)) break

        else
          tmp_op ← (no_rel_found(t, b))

  end
  if tmp_op then
    OP ← OP ∪ tmp_op
  else
    OP ← OP ∪ (lemma_missing(t))
end
return OP
```

3.3 PoS Tagging

The automated and the manual approach also take PoS information into account to understand the reuse transformation. Following the Greek morphology tagging system of Perseus (Bamman and Crane, 2011a), which maps PoS and case information to single characters[2], we manually PoS-tag the 199 reuse instances of Ancient Greek and the 162 of Latin, as well as the original Bible verses. Since Latin and Ancinet Greek PoS-taggers lack available implementations, appropriate trained models or simply accu-

[2] http://nlp.perseus.tufts.edu/syntax/treebank/agdt/1.7/docs/README.txt

		lemma coverage[1]	AGWN coverage[2]				total[3]
	corpus	lem.	syn.	hyper.	hypo.	co-hypo.	
CLTK	Greek Bible[4]	3238	1906	1422	1185	1422	4776
	Clement[5]	739	326	231	175	231	2189
	Latin Bible[4]	2473	1241	905	863	905	2618
	Bernard[5]	1219	643	471	455	471	1335
Biblindex	Greek Bible[4]	752	103	58	67	58	4776
	Clement[5]	455	54	24	33	24	2189
	Latin Bible[4]	2473	1365	1057	1023	1057	2618
	Bernard[5]	1219	701	531	520	531	1335
SBLGNT & LXX	Greek Bible[4]	4718	3385	2616	2092	2616	4776
	Clement[5]	1297	824	582	421	582	2189
	Latin Bible[4,6]	n/a	n/a	n/a	n/a	n/a	2618
	Bernard[5,6]	n/a	n/a	n/a	n/a	n/a	1335
combined	Greek Bible[4]	4723	3449	2684	2156	2684	4776
	Clement[5]	1548	899	653	495	653	2189
	Latin Bible[4]	2473	1378	1057	1023	1057	2618
	Bernard[5]	1219	706	531	520	531	1335

[1] number of tokens found by lemma resource
[2] number of lemmatized tokens covered by AGWN
[3] number of tokens in original and reuse
[4] original [5] reuse [6] no support for Latin

Table 1: Coverage of tokens by language resources

racy (Crane, 1991; vor der Brück et al., 2015), we perform this step manually to assure high accuracy. We also assign cases for the classes noun, article, adjective, and pronoun. We introduce *b* to represent the Latin ablative case, which does not exist in Greek.

3.4 Automated Approach

Our approach is to model the transformation process in terms of parameterized operations applied to the words in the reuse instance in order to obtain the original words. These operations use linguistic resources, such as lemma lists of classical Greek and Biblical Koine, and a synset database. For each transformation, we create the set of operations necessary to transform the reuse instance to its original.

Linguistic Resources. We investigate the following lemma lists to look up lemmatized forms of words—a prerequisite for looking up synsets: *Classical Language Tool Kit (CLTK)* (Johnson et al., 2014 2016) provides Ancient Greek and Latin lemma lists for 953,907 Greek and 270,228 Latin words. *Biblindex' Lemma Lists* contain entries for 65,537 Biblical Greek and 315,021 Latin words. *SBLGNT&LXX* refers to the Greek New Testament of the Society of Biblical Literature (SBLGNT)[3] and the Septuaginta (LXX), a translation of the Old Testament (Rahlfs, 1935a)[4] from the Center for Computer Analysis of Texts at UPenn. We acknowledge code-page corrections by M. Munson. SBLGNT&LXX provide 59,510 word-lemma-pairs.

We use *AGWN* (Bizzoni et al., 2014), which also contains Latin WordNet (Minozzi, 2009), to identify synsets (sets of synonyms) as well as hyperonyms, hyponyms, and co-hyponyms. From the wordnets' 98,950 synsets 33,910 synsets contain Ancient Greek and 27,126 synsets contain Latin words.

Coverage. Table 1 shows the coverage of each resource for our datasets. In the lower part of it we merge all lemma resources into one set of word-lemma pairs. The table shows that CLTK covers the Bible data better than the Hellenistic Greek as used in Clement of Alexandria, an author from 2nd century AD, writing in an archaic style with Biblical vocabulary, while also being influenced by Classical Greek. We also check the coverage of lemmata stemming from the same source (Biblindex) as our reuse. To increase the coverage for Greek, we consult SBLGNT&LXX, which in fact increases it. To not miss important information, we integrate all of the resources' data into our approach. For every lemma of a word we check the semantic relations in AGWN. We experimented with different ways of looking up lemmas and found that lower-casing all Latin tokens improved the success. For Greek, it had the opposite effect, which indicates that the Greek text contains more entities that are not available in lowercase in the lemma lists, so we did not change in that case.[5]

Operations and Classification. We define replacement operations using words and PoS as parameters, to transform a reuse instance *back into the Bible verse* it originates from. Table 2 lists the operations for the computational approach. We introduce the operations *NOPmorph*, *repl_pos*, and *repl_case* for words having the same cognate, and *lemma_missing(reuse_word)* when a word is not

[3] Logos Bible Software, Sbl new testament, 2014 `http://sblgnt.com/about/`

[4] CATSS LXX is prepared by the Thesaurus Linguae Graecae project directed by T. Brunner at UC Irvine, with further verification and adaptation by CATSS towards conformity with the individual Göttingen editions which appeared since 1935. LXXM is morphologically analyzed text of CATSS LXX prepared by CATSS led by R. Kraft (Philadelphia team)

[5] Often, the decision on whether to represent a word in upper or lower case letters is made by the editor, thus, our decision is affected by the edition we use for our research.

operation	description	example
NOP(reuse_word, orig_word)	Original and reuse word are equal.	*NOP(maledictus,maledictus)*
upper(reuse_word, orig_word)	Word is lowercase in reuse and uppercase in original.	*upper(kai,Kai)* - in Greek
lower(reuse_word, orig_word)	Word is uppercase in reuse and lowercase in original.	*lower(Gloriam,gloriam)*
lem(reuse_word, orig_word)	Lemmatization leads to equality of reuse and original.	*lem(penetrat,penetrabit)*
repl_syn(reuse_word, orig_word)	Reuse word replaced with a synonym to match original word.	*repl_syn(magnificavit,glorificavit)*
repl_hyper(reuse_word, orig_word)	Word in bible verse is a hyperonym of the reused word.	*hyper(cupit,habens)*
repl_hypo(reuse_word, orig_word)	Word in bible verse is a hyponym of the reused word.	*hypo(dederit,tollet)*
repl_co-hypo(reuse_word, orig_word)	Reused word and original have the same hyperonym.	*repl_co-hypo(magnificavit,fecit)*
NOPmorph(reuse_tags, orig_tags)	Case or PoS did not change between reused and original word.	*NOPmorph(na,na)*
repl_pos(reuse_tag, orig_tag)	Reuse and original contain the same cognate, but PoS changed.	*repl_pos(n,a)*
repl_case(reuse_tag, orig_tag)	Reuse and original have the same cognate, but the case changed	*repl_case(g,d)* - cases genitive, dative
lemma_missing(reuse_word, orig_word)	Lemma unknown for reuse or original word	*lemma_missing(tentari, inlectus)*
no_rel_found(reuse_wword, orig_word)	Relation for reuse or original word not found in AGWN	*no_rel_found(gloria,arguitur)*

Table 2: Operation list for the automated approach

known to any of our lemma resources as well as *no_rel_found(reuse_word, orig_word)* when the relationship between a reuse word and each potential word from the original is not covered by AGWN.

Algorithm 1 shows our approach to classify the reuse transformation by identifying the operations. For each reuse token, we identify the first applicable operation matching the foremost Bible verse word (iterating the verse) in the following order: exact word match (*NOP: no operation*), case changed to *upper* or *lower*. Thereafter, we look up the lemma and return *lem* if the lemma of the reused word matches the lemma of the original. For these four, we also check the morphology, in addition returning whether the original has the same PoS and case (*NOPmorph*) or whether PoS changed (*repl_pos*), case changed (*repl_case*), or both. So up to three operations can be returned per word. Finally, we check for synonyms (*repl_syn*), hyperonyms (*hyper*), hyponyms (*hypo*), and co-hyponyms (*repl_co-hypo*), but do not check morphology. If a Bible verse word is used as a match, it is not used again for any other word from the reuse.

3.5 Qualitative Approach

To obtain a deeper understanding of the limitations of linguistic resources for our purpose, two graduate students (one Latinist, one Classical Archeologist) manually analyze 100 Greek and 60 Latin reuse instances with their expert knowledge, using an extended set of operations. It comprises *ins(word)* (insert a word) and *del(word)* (delete a word)—two operations we ignore in the automated approach where we focuse on the coverage of the resources. It also has a richer set of replacement operations: those from the upper part of Table 2 (without *upper* and *lower*), and instead of only using *repl_case* when a cognate stays the same, we refine it and assign *all* changing morphological categories from Perseus' tag set for *any* "relativeness" between two words (e.g., repl_case_a_g).

4 Results

We now present the results for our research questions in Sec. 4.1–4.3, which are summarized and further interpreted in Sec. 4.4.

4.1 Literal Share of the Reuse (RQ1)

We obtain a first understanding of the reuse by looking at the percentage of overlapping words between reuse instance and original Bible verse. We measure the longest common substring based on word tokens. Fig. 2 shows the distributions, distinguishing between a *lemmatized* and *non-lemmatized* word comparison.

While lemmatizing words before comparison has only a small impact, we observe differences between the datasets. In our Latin dataset, the overlap is significantly higher than in the Greek dataset Sec. 3.2. 25 % (upper quartile) of Bernard's reuse instances have 50 % or more tokens overlap with their original, which is only the case for less than 25 % in Clement's Greek data. Still, large overlaps of up to 75 % (top

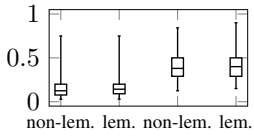


Figure 2: Ratios of literal overlaps between reuse instance and original (left: Greek, right: Latin)

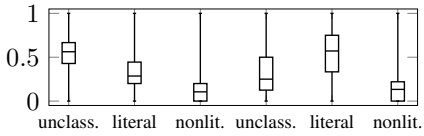


Figure 3: Ratios of unclassified, literal, and non-literal words in reuse instances (left: Greek, right: Latin)

whisker) in our Greek and up to around 90 % in our Latin dataset exist—so a small fraction of the reuse contains literal parts Sec. 3.2.

For a more precise understanding of the literalness, we group operations into literal (NOP, upper, lower, lem), non-literal (repl_syn, repl_hyper, repl_hypo, repl_co-hypo), and unclassified (no_rel_found and lemma_missing). Within each reuse instance, we calculate their relative occurrence using the results of the automated approach (explained shortly). Fig. 3 shows the distribution of these relative occurrences for all reuse instances. It confirms Fig. 2 by showing a higher rate of literalness for Latin compared to Greek. In summary, it also shows that the Latin reuse can be better classified by our approach, which takes the lemma lists and AGWN into account.

4.2 Automated Approach (RQ2.1)

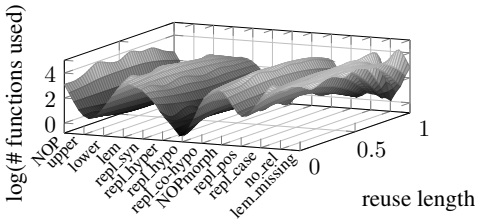
Table 3 shows the total number of operations identified for the transformation from reuse instances to the Greek and Latin originals. For 987 (45 %) out of 2189 words in the Greek instances and for 893 (67 %) out of 1335 words in the Latin instances, we were able to identify at least one operation, which already indicates to what extent the resources are helpful.

Fig. 4 visualizes the distribution of the frequencies (y-axis) of each operation (x-axis) together with the distribution of the operations' positions in the reuse instances (z-axis). The latter is calculated as the relative position $p \in [0..1]$ of an operation with respect to the length of the reuse instance. It indicates that most operation types are distributed over the whole reuse

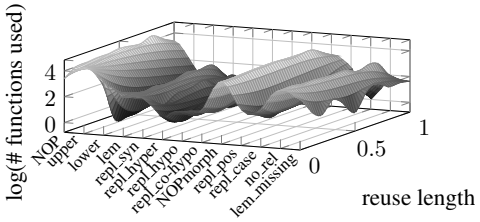
	NOP	upper	lower	lem	syn	hyper	hypo	co-hypo
Occ. Greek	337	6	0	356	153	20	14	101
Occ. Latin	587	0	44	102	60	14	28	68

	NOPmorph	repl_pos	repl_case	no_rel_found	lem_missing
Occ. Greek	420	49	258	563	639
Occ. Latin	617	46	75	347	85

Table 3: Absolute numbers of operations identified automatically



(a) Greek



(b) Latin

Figure 4: Occurrence of operations in reuse instances. X-axis: operations; Y-axis: relative position within reuse instances. Z-axis: natural logarithm of number of operations. Values are smoothed by spline interpolation. The order of operations is arbitrary.

length without a particular trend in both datasets. We only encounter a frequent use of *upper* at the first position in Latin, which means that Bernard often starts his Biblical references with literal Bible words.

After having checked the overall coverage of the linguistic resources for all tokens (cf. Sec. 3.4), we now specifically investigate to what extent the resources support identifying the reuse transformation for the non-literal reuse using our approach. We introduce the measures sup_{lem} and sup_{AGWN} to calculate how often looking up a lemma or subsequently a synset element was successful. This is easy based on our operations. Let $\text{Occ}(o)$ be the number of occurrences of an operation o, obtained from Table 3. The operations that successfully looked up a lemma (before consulting AGWN) are lem_success={lem, syn, repl_hyper, repl_hypo, repl_co-hypo, no_rel_found}. Now recall that lem_missing represents the case when a reuse token was not found in the lemma resources. Then $\text{sup}_{\text{lem}} = \frac{\sum_{\text{Occ}(o)\ o \in \text{lem\_success}}}{\sum_{\text{Occ}(o)\ o \in \text{lem\_success} \cup \{\text{lem\_missing}\}}}$. We obtain a sup_{lem} of 0.65 for the Greek reuse and 0.88 for the Latin reuse. Similarly, the operations that successfully looked up from AGWN are agwn_success={syn,

operation	Greek	Latin	operation	Greek	Latin
repl_syn	78 (40.6%)	91 (40.4%)	repl_gender	6 (3.1%)	1 (0.4%)
repl_ant	1 (0.5%)	0	repl_mood	11 (5.7%)	12 (5.3%)
repl_hyper	3 (1.6%)	0	repl_number	17 (8.9%)	17 (7.6%)
repl_hypo	11 (5.7%)	0	repl_person	5 (2.6%)	14 (6.2%)
lem	1 (0.5%)	2 (0.9%)	repl_pos	18 (9.4%)	33 (14.7%)
repl_co-hypo	0	1 (0.4%)	repl_tense	3 (1.6%)	9 (4.0%)
repl_case	38 (19.8%)	36 (16%)	repl_voice	0	8 (3.6%)

Table 4: Numbers of replacement operations identified for the manual reuse transformation.

operation	Greek	Latin	operation	Greek	Latin
repl_case_a_b	0	6	repl_case_g_a	5	2
repl_case_a_n	9	4	repl_case_g_n	4	2
repl_case_b_a	0	10	repl_case_n_a	7	5
repl_case_d_a	0	2	repl_case_n_d	3	0
repl_case_d_g	3	0	repl_case_v_g	0	2
repl_case_d_n	5	0			

Table 5: Numbers of case replacements

repl_hyper, repl_hypo, repl_co-hypo}, with no_rel_found representing a failed lookup. Then: $\text{sup}_{\text{AGWN}} = \frac{\sum_{\text{Occ}(o)} o \in \text{agwn\_success}}{\sum_{\text{Occ}(o)} o \in \text{agwn\_success} \cup \{\text{no\_rel\_found}\}}$. We obtain sup_{AGWN} of 0.34 for Greek and 0.33 for Latin.

These values can be interpreted as follows. The lemma resources for genre- and time-specific text work well for less-literal reuse, but the resources for semantic relationships (synset databases) show a lack of support and need further development.

4.3 Qualitative Approach (RQ2.2)

We manually identify the transformation operations for 60 reuse instances of the Ancient Greek data and for 100 of the Latin data. Here, NOPs cover 9.3 %, insertions 49.8 %, and deletions cover 30.5 % in the Greek data. NOPs cover 26.1 %, insertions 49.7 %, and deletions 11.9 % in the Latin data.

Table 4 shows the ratios of the various *repl* operations based on the remaining 10.4 % and 12.2 %. Similar to the automated approach, we observe a strong use of synonyms and other semantic-level operations, and also a certain portion of switching morphological categories, which indicates para-phrasal reuse. In the Greek data, PoS changes cover about 9%, out of which a participle became a verb (7 times) and vice-versa (5 times). In our Latin data, PoS changes represent 15% of replacements: often a pronoun changed to a noun (6 times) and a participle became a verb (12 times). Case changes are shown in Table 5. Significantly often, an ablative became an accusative, because often changing prepositions expect different cases, or an accusative was replaced by an ablative or nominative, because para-phrasal expression changed.

We encounter **exceptions** that prevent applying the operations. In the Greek data, one word is replaced with its antonym[6]; once, a synonym also changes its PoS. Four times, more than one morphological category changes, twice an auxiliary is deleted, and five times inserted. We find one writing variance (lem), and three times a synonym is replaced by a multi-word expression. In the Latin data, in 16 cases a synonym is replaced and morphological information changed. Seven times, more than one morphological parameter changes for the same cognate. Eight times, an auxiliary is inserted or deleted, and twice, a writing variance is encountered. A synonym is replaced by more than one word five times. In one case, a reuse is too paraphrasal for any word to match semantic relationships (e.g., *judged calmly*—Bernard vs. *fake friend* - Sal 12 18).

4.4 Summary and Discussion

RQ1. The reuse is significantly non-literal and only lemmatizing words does not help discovering it. Our results show that reuse in two substantial historical texts requires techniques beyond simple preprocessing (e.g., stemming or lemmatizing), which explains why plagiarism-detection systems fail when paraphrases are used (Alzahrani et al., 2012). Bible verses are often used to justify an author's claim, so only relevant parts of the Bible verse are reused. In the reuse the Bible verse is modified to better fit the syntactical and semantic context of an author's new text, as shown in Tables 4 and 5.

RQ2.1. The results from our automated approach are encouraging, showing the feasibility of extending reuse-detection techniques with linguistic resources. Yet, it is not clear which precision and recall could be achieved and how existing techniques need to be adapted and calibrated. This investigation is beyond the scope of this study and subject to our future work.

The linguistic resources support the automated

[6]Translation: "**the** God, the good (**one**)" (Clement) vs. "**none** is good but the God" (Bible).

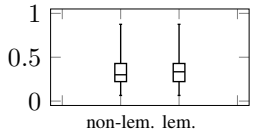


Figure 5: Ratios of literal overlaps in the whole Latin dataset

approach, but only for about one third of the lookups. The manually identified exceptions show that finding a connection between original verse and reuse can be difficult when there is only a vague semantic one.

RQ2.2. Our results show that the automated approach cannot capture the richness of the manual approach. Especially from the exceptions, it is clear that less-literal reuse does not only need information from a word's semantic environment, but also that it needs to be identified by looser relations, such as co-hyponyms, multi-to-multi-word associations or implicit meanings, which can be hidden in structural or more broader expert knowledge.

5 Threats to Validity

External Validity. We enhance the external validity of our work by focusing on Bible verses—one of the oldest, most conveyed, and cited sources of Ancient Greek, offering a vast amount of primary source text and also coming with a long history of scholars studying it. Clement of Alexandria is known for his retelling of biblical excerpts (Clemens, 1905 1909; Freppel, 1865), providing an interesting base for reuse investigation. The french abbot Bernard of Clairvaux (Smith, 2010) is equally known for his influence to the Cistercian order and his work in biblical studies. Furthermore, the chosen lemma resources are the most extensive ones existing for Ancient Greek and Latin. We chose the AGWN, since it is freely available, offering one of the largest synset database for Ancient Greek and Latin.

Internal Validity. A threat is that our ground truth has mistakes, as the PoS tagging was done by one author only and relied on a manual post-correction. The selection criteria in Sec. 3.2 were chosen to ensure quality and comparability. Extreme outliers in the length of the reuse instance or source (multiple Bible verses) are cut-off. For Greek, 33 are cut-off, as opposed to Latin, where our sample is significantly smaller than the whole population that we have. To automatically check whether the sample has similar characteristics with respect to the literal reuse, we create Fig. 5. It shows the overlap of the whole 1128 instances of Bernard's extracted reuse, which when compared to Fig. 2 (right) supports the representativeness of our sample. Last, we can only derive operation replacements when a word token was covered by the lemma sources, contained in AGWN, and when there actually exists a relation between two words. Also, our authors' vocabulary can differ in terms of domain knowledge, personal idiolect, and age of the Biblical vocabulary.

6 Conclusion

We presented a study of historical—and mostly non-literal—text reuse. We automatically and manually characterize the reuse and identify to what extent existing linguistic resources are able to cover non-literal text reuse. Our results show the potential as well as the necessity to develop robust techniques and to extend linguistic resources for analyzing and detecting such reuse. Our results can help to enhance paraphrase generation to model automatic ways on how small text portions can be rephrased. Considering the effects of syntactic rearrangement of reuse can also support such efforts. A smarter automated approach for deriving an original text excerpt would be learning so-called edit scripts (Kehrer, 2014; Chawathe et al., 1996), which more precisely identify operations an author performed on a text to transform it into another version. Whether learning edit scripts on such intricate transformations is possible is an open question and valuable future research. Finally, analyzing further languages and data sets helps to further complete our findings.

Acknowledgments

We thank Laurence Mellerin for providing the datasets we used, and for valuable advice on its content. Our work is funded by the German Federal Ministry of Education and Research (grant 01UG1509).

References

[Agirre and Edmonds2007] Eneko Agirre and Philip Edmonds. 2007. *Word Sense Disambiguation - Algorithms and Applications*. Springer Netherlands.

[Aland and Aland1966] Kurt Aland and Barbara Aland, editors. 1966. *The Greek New Testament*. Deutsche Bibelgesellschaft-United Bible Societies, 27 edition.

[Alzahrani et al.2012] Salha M. Alzahrani, Naomie Salim, and Ajith Abraham. 2012. Understanding plagiarism linguistic patterns, textual features, and detection methods. *Trans. Sys. Man Cyber Part C*, 42(2):133–149.

[Bamman and Crane2008] David Bamman and Gregory Crane. 2008. The logic and discovery of textual allusion. In *LaTeCH (Language Technology for Cultural Heritage Data)*, Marrakech Morocco. LREC.

[Bamman and Crane2011a] David Bamman and Gregory Crane. 2011a. The ancient greek and latin dependency treebanks. In *Caroline Sporleder, Antal van den Bosch, & Kalliopi Zervanou (Eds) Language technology for cultural heritage: Selected papers from the LaTeCH Workshop Series*, pages 79–98, Berlin, Germany. Springer-Verlag.

[Bamman and Crane2011b] David Bamman and Gregory Crane. 2011b. Measuring historical word sense variation. In *Proceedings of the 11th ACM/IEEE-CS Joint Conference on Digital libraries (JCDL 2011)*, pages 1–10. ACM Digital Library.

[Barrón-Cedeño et al.2013] Alberto Barrón-Cedeño, Marta Vila, M.Antònia Martí, and Paolo Rosso. 2013. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistic*, 39(4):917–947.

[Bensalem et al.2015] Imene Bensalem, Imene Boukhalfa, Paolo Rosso, Lahsen Abouenour, Kareem Darwish, and Salim Chikhi. 2015. Overview of the AraPlagDet PAN@FIRE2015 Shared Task on Arabic Plagiarism Detection. In *FIRE 2015 Working Notes Papers, 4-6 December, Gandhinagar, India*, December.

[Bizzoni et al.2014] Yuri Bizzoni, Federico Boschetti, Harry Diakoff, Riccardo Del Gratta, Monica Monachini, and Gregory Crane. 2014. The making of ancient greek wordnet. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

[Büchler et al.2012] Marco Büchler, Gregory Crane, Maria Moritz, and Alison Babeu. 2012. Increasing recall for text re-use in historical documents to support research in the humanities. In *Theory and Practice of Digital Libraries, Lecture Notes in Computer Science*, volume 7489, pages 95–100. Springer, Berlin Heidelberg.

[Büchler2013] Marco Büchler. 2013. *Informationstechnische Aspekte des Historical Text Re-use (English: Computational Aspects of Historical Text Re-use*. Ph.D. thesis, Leipzig University, Germany.

[Chawathe et al.1996] Sudarshan S Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. In *ACM SIGMOD Record*, volume 25, pages 493–504. ACM.

[Clemens1905 1909] Titus Flavius Clemens. 1905-1909. Werke (in greek). In Otto Stählin, editor, *Die Griechischen Christlichen Schrifteller, Berlin, v. 12, 15, 27*. Leipzig.

[Clément d'Alexandrie2011] Clément d'Alexandrie, 2011. *Quel riche peut-être sauvé*. éditions du Cerf, Paris, sources chrtiennes 537 edition.

[Cosaert2008] Carl P. Cosaert. 2008. *The Text of the Gospels in Clement of Alexandria*. New Testament in the Greek Fathers. Society of Biblical Literature.

[Crane1985] Gregory Crane. 1985. Perseus digital library. `http://www.perseus.tufts.edu/hopper/`.

[Crane1991] Gregory Crane. 1991. Generating and parsing classical greek. *Literary and Linguistic Computing*, 6(4):243–245.

[Dolan and Brockett2005] Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

[Freppel1865] Charles-Emile Freppel. 1865. *Clement d'Alexandrie*.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

[Geyken and Gloning2014] Alexander Geyken and Thomas Gloning. 2014. A living text archive of 15th-19th century german: Corpus strategies, technology, organization. In *Corpus Linguistics and Interdisciplinary Perspectives on Language - CLIP*. Narr Tbingen.

[Houghton2013a] H.A.G. Houghton. 2013a. Patristic evidence in the new edition of the vetus latina iohannes. In L. Mellerin and H.A.G. Houghton, editors, *Biblical Quotations in Patristic Texts (Studia Patristica 54)*, pages 69–85. Peeters, Leuven.

[Houghton2013b] H.A.G. Houghton. 2013b. The use of the latin fathers for new testament textual criticism. In B.D. Ehrman and M.W. Holmes, editors, *The Text of the New Testament in Contemporary Research. Essays on the Status Quaestionis second edition. NTTSD.*, pages 375–405. Brill, Leiden.

[Jing1998] Hongyan Jing. 1998. Usage of wordnet in natural language generation. In *Proceedings of the Workshop on Usage of WordNet in Natural Language Processing Systems (COLING-ACL'98)*. Columbia University Academic Commons.

[Johnson et al.2014 2016] Kyle P. Johnson, Patrick J. Burns, Luke Hollis, Martín Pozzi, Amit Shilo, Stephen Margheim, Gitter Badger, and Eamonn Bell. 2014–2016. Cltk: The classical language toolkit. `https:`

//github.com/cltk/cltk. DOI 10.5281/zenodo.44555 v0.1.32.

[Kehrer2014] Timo Kehrer. 2014. Generierung konsistenzerhaltender editierskripte im kontext der modellversionierung. In Wilhelm Hasselbring and Nils Christian Ehmke, editors, *Software Engineering 2014, Fachtagung des GI-Fachbereichs Softwaretechnik, 25. Februar - 28. Februar 2014, Kiel, Deutschland*, volume 227 of *LNI*, pages 57–58. GI.

[Lee2007] John Lee. 2007. A computational model of text reuse in ancient literary texts. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic*, pages 472–479. Association for Computational Linguistics.

[Madnani and Dorr2010] Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Comput. Linguist.*, 36(3):341–387, September.

[Mellerin2014] Laurence Mellerin. 2014. New ways of searching with biblindex, the online index of biblical quotations in early christian literature. In Claire Clivaz, Andrew Gregory, and David Hamidovic, editors, *Digital Humanities in Biblical, Early Jewish and Early Christian Studies*, chapter 11, pages 175–192. Brill, Leiden.

[Mellerin2016] Laurence Mellerin. 2016. Biblindex. http://www.biblindex.mom.fr/.

[Miller et al.1990] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312.

[Minozzi2009] Stefano Minozzi, 2009. *Innsbrucker Beitrge zur Sprachwissenschaft*, volume 137, chapter The Latin WordNet Project, pages 707–716. Institut fr Sprachen und Literaturen der Universitt Innsbruck, Innsbruck.

[Pantelia2014] Maria Pantelia. 2014. Thesaurus linguae graecae. http://stephanus.tlg.uci.edu/index.php.

[Patwardhan et al.2003] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen, 2003. *Computational Linguistics and Intelligent Text Processing: 4th International Conference (CICLing)*, chapter Using Measures of Semantic Relatedness for Word Sense Disambiguation, pages 241–257. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Piotrowski2012] Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts (Synthesis Lectures on Human Language Technologies)*. Morgan & Claypool Publishers.

[Rahlfs1935a] Alfred Rahlfs, editor. 1935a. *Septuaginta*. Württembergische Bibelanstalt, 9 edition. 1971.

[Rahlfs1935b] Alfred Rahlfs, editor. 1935b. *Septuaginta, id est Vetus Testamentum Graece juxta LXX interpretes*. Rahlfs. 2 vol., 1950.

[Smith2010] William Smith. 2010. *Catholic Church Milestones: People and Events That Shaped the Institutional Church*. Indianapolis: Left Coast.

[Vinzent et al.2013] M. Vinzent, L. Mellerin, and H.A.G. Houghton, editors. 2013. *Biblical Quotations in Patristic Texts (Studia Patristica 54)*. Theory and Applications of Natural Language Processing. Peeters, Leuven.

[vor der Brück et al.2015] Tim vor der Brück, Steffen Eger, and Alexander Mehler. 2015. Lexicon-assisted tagging and lemmatization in latin: A comparison of six taggers and two lemmatization models. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 105–113, Beijing, China, July. Association for Computational Linguistics.

[Weber R.1969 1994 2007] Gribomont J. Weber R., Fischer B., editor. 1969, 1994, 2007. *Biblia sacra juxta vulgatam versionem*. Deutsche Bibelgesellschaft.

[Zhao et al.2009] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 834–842, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zitouni2014] Imed Zitouni. 2014. Natural language processing of semitic languages.

A Graph Degeneracy-based Approach to Keyword Extraction *

Antoine J.-P. Tixier[1], **Fragkiskos D. Malliaros**[1,2], **Michalis Vazirgiannis**[1]
[1]Computer Science Laboratory, École Polytechnique, Palaiseau, France
[2]Department of Computer Science and Engineering, UC San Diego, La Jolla, CA, USA
{anti5662,fmalliaros,mvazirg}@lix.polytechnique.fr

Abstract

We operate a change of paradigm and hypothesize that keywords are more likely to be found among *influential* nodes of a graph-of-words rather than among its nodes high on *eigenvector*-related centrality measures. To test this hypothesis, we introduce unsupervised techniques that capitalize on *graph degeneracy*. Our methods strongly and significantly outperform all baselines on two datasets (short and medium size documents), and reach best performance on the third one (long documents).

1 Introduction

Keyword extraction is a central task in NLP. It finds applications from information retrieval (notably web search) to text classification, summarization, and visualization. In this study, we focus on the task of *unsupervised single-document keyword extraction*. Following (Mihalcea and Tarau, 2004), we concentrate on *keywords* only, letting the task of *keyphrase* reconstruction as a post-processing step.

More precisely, while we capitalize on a graph representation of text like several previous approaches, we deviate from them by making the assumption that keywords are not found among *prestigious* nodes (or more generally, nodes high on eigenvector-related centrality metrics), but rather among *influential* nodes. Those nodes may not have many important connections (like their prestigious counterparts), but they are ideally placed at the core of the network. In other words, this switches the objective from capturing the *quality* and *quantity* of single node connections, to taking into account the *density* and *cohesiveness* of groups of nodes. To operate this change of paradigm, we propose several algorithms that leverage the concept of *graph degeneracy* (Malliaros et al., 2016a).

Our contributions are threefold: (1) we propose new unsupervised keyword extraction techniques that reach state-of-the art performance, (2) we apply the K-truss algorithm to the task of keyword extraction for the first time, and (3) we report new insights on the interplay between window size, graph-of-words structure, and performance.

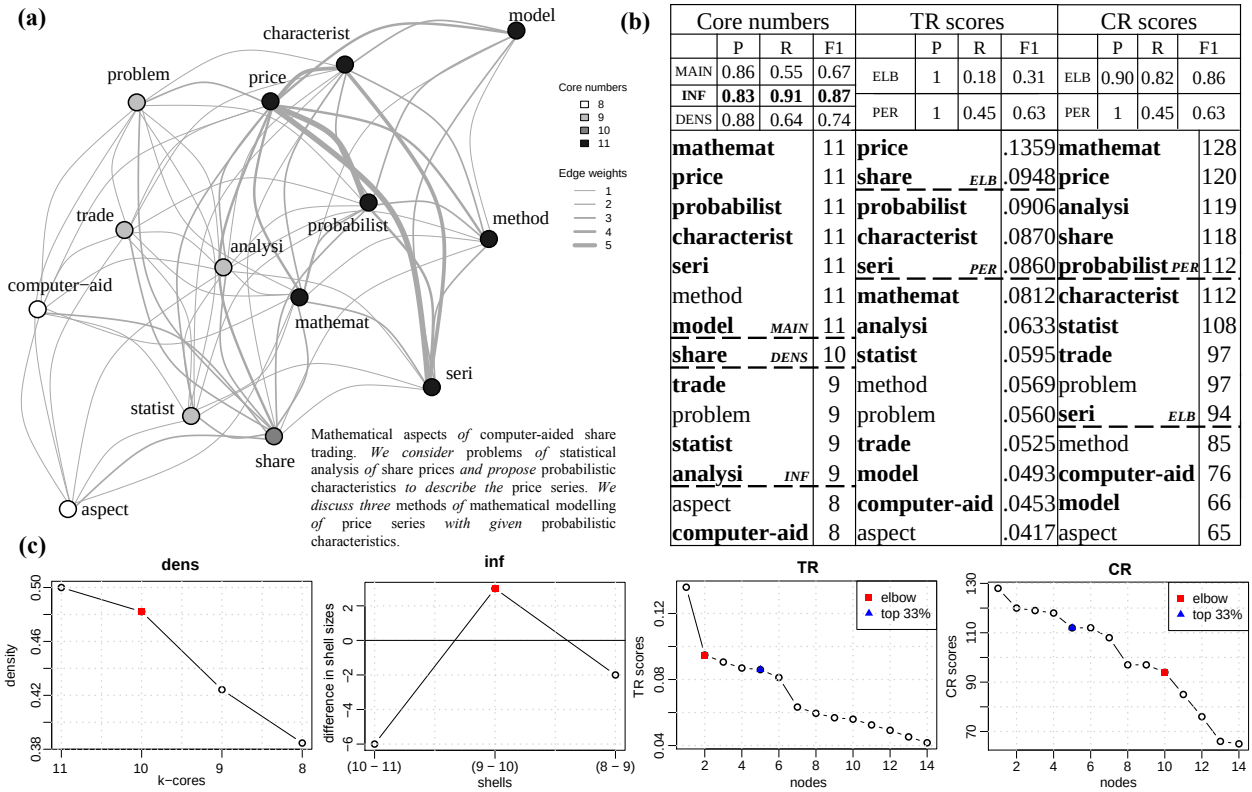
2 Graph-of-Words Representation

Many ways of encoding text as a graph have been explored in order to escape the very limiting *term-independence* assumption made by the traditional vector space model. In this study, we adopt the seminal Graph-of-Words representation (GoW) of (Mihalcea and Tarau, 2004), for its simplicity, high historical success, and above all because it was recently used in several approaches that reached very good performance on various tasks such as information retrieval (Rousseau and Vazirgiannis, 2013), document classification (Malliaros and Skianis, 2015; Rousseau et al., 2015), event detection (Meladianos et al., 2015), and keyword extraction (Rousseau and Vazirgiannis, 2015).

While sophisticated variants of the GoW model would be worth exploring (edge weights based on mutual information or word embeddings, adaptive window size, etc.), we aim here at making a bet-

*This research is supported in part by the OpenPaaS::NG project.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1860–1870,
Austin, Texas, November 1-5, 2016.



Core numbers	P	R	F1
MAIN	0.86	0.55	0.67
INF	**0.83**	**0.91**	**0.87**
DENS	0.88	0.64	0.74

TR scores	P	R	F1
ELB	1	0.18	0.31
PER	1	0.45	0.63

CR scores	P	R	F1
ELB	0.90	0.82	0.86
PER	1	0.45	0.63

		TR scores		CR scores	
mathemat	11	**price**	.1359	**mathemat**	128
price	11	**share** *ELB*	.0948	**price**	120
probabilist	11	**probabilist**	.0906	**analysi**	119
characterist	11	**characterist**	.0870	**share**	118
seri	11	**seri** *PER*	.0860	**probabilist** *PER*	112
method	11	**mathemat**	.0812	**characterist**	112
model *MAIN*	11	**analysi**	.0633	**statist**	108
share *DENS*	10	**statist**	.0595	**trade**	97
trade	9	method	.0569	problem	97
problem	9	problem	.0560	**seri** *ELB*	94
statist	9	**trade**	.0525	method	85
analysi *INF*	9	**model**	.0493	**computer-aid**	76
aspect	8	**computer-aid**	.0453	**model**	66
computer-aid	8	aspect	.0417	aspect	65

Figure 1: (a) Graph-of-words ($W = 8$) for document #1512 of Huth2003 decomposed with k-core (non-(nouns and adjectives) in *italic*). (b) Keywords extracted by each proposed and baseline method (human assigned keywords in **bold**). (c) Selection criterion of each method except *main* (does not apply).

ter use of an existing representation of text, not at proposing a new one. This is why, to demonstrate the additional skill brought by our proposed methods, we stick to the basic GoW framework.

As shown in Figure 1 (a), the GoW representation of (Mihalcea and Tarau, 2004) encodes a piece of text as an undirected graph where nodes are unique nouns and adjectives in the document, and where there is an edge between two nodes if the terms they represent co-occur within a window of predetermined size W that is slided over the entire document from start to finish, overspanning sentences. Furthermore, edges are assigned integer weights matching co-occurrence counts. This fully statistical approach is based on the *Distributional Hypothesis* (Harris, 1954), that is, on the premise that the relationship between words can be determined by the frequency with which they share local contexts of occurrence.

3 Graph Degeneracy

The concept of graph degeneracy was introduced by (Seidman, 1983) with the k-core decomposition technique and was first applied to the study of cohesion in social networks. Here, we consider it as an umbrella term also encompassing the K-truss algorithm (Cohen, 2008). In what follows, $G(V, E)$ is a graph with $|V|$ nodes and $|E|$ edges. Note that for graphs-of-words, the nodes V are labeled according to the unique terms they represent.

3.1 k-core subgraph

A core of order k (or k-core) of G is a maximal connected subgraph of G in which every vertex v has at least degree k (Seidman, 1983). If the edges are unweighted, the degree of v is simply equal to the count of its neighbors, while in the weighted case, the degree of v is the sum of the weights of its incident edges. Note that with the definition of GoW previously given, node degrees (and thus, k) are integers in both cases since edge weights are integers.

As shown in Figure 2 (a), the **k-core decomposition** of G is the set of all its cores from 0 (G itself) to k_{max} (its main core). It forms a hierarchy of nested subgraphs whose cohesiveness and size respectively increase and decrease with k (Seidman, 1983). The main core of G is a coarse approximation of its densest subgraph (Wang and Cheng, 2012), and should be seen as a *seedbed* within which it is possible to find more cohesive subgraphs (Seidman, 1983). Finally, the **core number** of a node is the highest order of a k-core subgraph that contains this node.

3.2 K-truss subgraph

K-truss is a triangle-based extension of k-core introduced by (Cohen, 2008). More precisely, the K-truss subgraph of G is its largest subgraph where every edge belongs to at least $K-2$ triangles. In other words, every edge in the K-truss joins two vertices that have at least $K-2$ common neighbors. Compared to k-core, K-truss thus does not iteratively prune nodes out based on the number of their *direct links*, but also based on the number of their *shared connections*. This more accurately captures cohesiveness.

As a result, the K-trusses are smaller and denser subgraphs than the k-cores, and the maximal K-truss of G better approximates its densest subgraph. In essence, and as illustrated in Figure 2 (b), the K-trusses can be viewed as the essential parts of the k-cores, i.e., what is left after the less cohesive elements have been filtered out (Wang and Cheng, 2012). By analogy with k-core, the **K-truss decomposition** of G is the set of all its K-trusses from $K=2$ to K_{max}, and the **truss number** of an *edge* is the highest order of a truss the edge belongs to. By extension, we define the truss number of a *node* as the maximum truss number of its incident edges.

3.3 k-shell

Depending on context, we will refer to the k-shell as the part of the k-core (or truss) that is not included in the $(k+1)$-core (or truss).

3.4 Graph degeneracy and spreading influence

In social networks, it has been shown that the best spreaders (i.e., the nodes able to propagate information to a large portion of the network at minimal time and cost) are not necessarily the highly connected individuals (i.e., the hubs), but rather those located at the core of the network (i.e., forming dense and cohesive subgraphs with other central nodes), as indicated by graph degeneracy algorithms (Kitsak et al., 2010). Put differently, the spreading influence of a node is related to its structural position within the graph (*density* and *cohesiveness*) rather than to its *prestige* (random walk-based degree). More recently, (Malliaros et al., 2016b) found that the truss number is an even better indicator of spreading influence than the core number. Motivated by these findings, we posit that taking *cohesiveness* into account with the core and truss decomposition of a graph-of-words could improve keyword extraction performance. That way, by analogy with the notion of influential spreaders in social networks, we hypothesize that *influential words* in graphs-of-words will act as representative keywords.

3.5 Complexity

(Batagelj and Zaveršnik, 2002) proposed $\mathcal{O}(|V| + |E|)$ and $\mathcal{O}(|E| \log |V|)$ time algorithms for k-core decomposition in the unweighted (resp. weighted) case. These algorithms are both $\mathcal{O}(|V|)$ in space. Computing the K-truss decomposition is more expensive, and requires $\mathcal{O}(|E|^{1.5})$ time and $\mathcal{O}(|V| + |E|)$ space (Wang and Cheng, 2012). Finally, building a graph-of-words is linear in time and space: $\mathcal{O}(|V|W)$ and $\mathcal{O}(|V| + |E|)$, respectively.

4 Related Work and Point of Departure

TextRank. One of the most popular approaches to the task of *unsupervised single-document keyword extraction* is TextRank (Mihalcea and Tarau, 2004), or TR in what follows. In TR, the nodes of graphs-of-words are ranked based on a modified version of the PageRank algorithm taking edge weights into account, and the top $p\%$ vertices are kept as keywords.
Limitations. TR has proven successful and has been widely used and adapted. However, PageRank, which is based on the concept of random walks and is also related to *eigenvector centrality*, tends to favor nodes with many important connections regardless of any cohesiveness consideration. For undirected graphs, it was even shown that PageRank values are proportional to node degrees (Grolmusz, 2015). While well suited to the task of

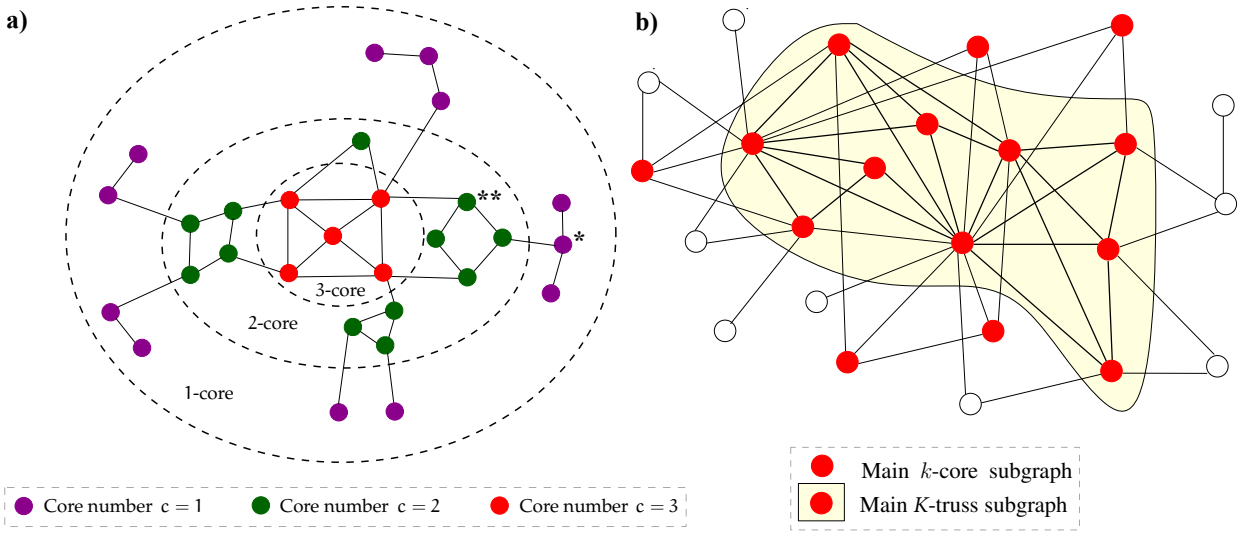


Figure 2: (a) k-core decomposition illustrative example. Note that while nodes * and ** have same degree (3), node ** makes a more influential spreader as it lies in a higher core than node *. (b) k-core versus K-truss. The main K-truss subgraph can be considered as the *core* of the main core.

prestige-based ranking in the Web and social networks (among other things), PageRank may thus not be ideal for keyword extraction. Indeed, a fundamental difference when dealing with text is the paramount importance of cohesiveness: keywords need not only to have many important connections but also to form *dense substructures* with these connections. Actually, most of the time, keywords are n-grams (Rousseau and Vazirgiannis, 2015). Therefore, we hypothesize that keywords are more likely to be found among the *influential spreaders* of a graph-of-words – as extracted by degeneracy-based methods – rather than among the nodes high on *eigenvector*-related centrality measures.

Topical vs. network coherence. Note that, unlike a body of work that tackled the task of keyword extraction and document summarization from a *topical* coherence perspective (Celikyilmaz and Hakkani-Tür, 2011; Chen et al., 2012; Christensen et al., 2013), we deal here with *network* coherence, a purely graph theoretic notion orthogonal to topical coherence.

Graph degeneracy. The aforementioned limitation of TR motivated the use of graph degeneracy to not only extract central nodes, but also nodes forming dense subgraphs. More precisely, (Rousseau and Vazirgiannis, 2015) applied both unweighted and weighted k-core decomposition on graphs-of-words and retained the members of the main cores as keywords. Best results were obtained in the weighted case, with small main cores yielding good precision but low recall, and significantly outperforming TR. As expected, (Rousseau and Vazirgiannis, 2015) observed that cores exhibited the desirable property of containing "long-distance n-grams".

In addition to superior quantitative performance, another advantage of degeneracy-based techniques (compared to TR, which extracts a constant percentage of nodes) is *adaptability*. Indeed, the size of the main core (and more generally, of every level in the hierarchy) depends on the structure of the graph-of-words, which by nature is uniquely tailored to the document at hand. Consequently, the distribution of the number of extracted keywords matches more closely that of the human assigned keywords (Rousseau and Vazirgiannis, 2015).

Limitations. Nevertheless, while (Rousseau and Vazirgiannis, 2015) made great strides, it suffers the following limitations: (1) k-core is *good* but not *best* in capturing cohesiveness; (2) retaining only the main core (or truss) is suboptimal, as one cannot expect all the gold standard keywords to be found within a unique subgraph – actually, many valuable keywords live in lower levels of the hierarchy (see Figure 1); and (3) the coarseness of the k-core

decomposition implies to work at a high granularity level (selecting or discarding a large group of words at a time), which diminishes the flexibility of the extraction process and negatively impacts performance.

Research objectives. To address the aforementioned limitations, we investigate in this study (1) the use of K-truss to get a finer-grained hierarchy of more cohesive subgraphs, in order to filter unwanted words out of the upper levels and improve flexibility; (2) the automated selection of the best level in the core (or truss) hierarchy to increase recall while preserving most of the precision; and (3) the conversion of node core (or truss) numbers into ranks, to decrease granularity from the *subgraph* to the *node* level, while still leveraging the valuable cohesiveness information captured by degeneracy.

5 Proposed Methods

In what follows, we introduce the strategies we devised to implement our research objectives.

5.1 Density

With the underlying assumption that keywords are found in cohesive subgraphs-of-words and are not all contained in the main core (or truss), an intuitive, straightforward stopping criterion when going down the core (or truss) hierarchy is a density-based one. More precisely, it may be advantageous to go down the hierarchy as long as the desirable cohesiveness properties of the main core or truss are maintained, and to stop when these properties are lost. This strategy is more formally detailed in Algorithm 1, where $G(V, E)$ is a graph-of-words, $levels$ corresponds to the vector of the n_{levels} unique k-core (or truss) numbers of V sorted in decreasing order, and the density of G is defined as:

$$density(G) = \frac{|E|}{|V|\,(|V|-1)} \qquad (1)$$

As can be seen in Figure 1 (c) and as detailed in Algorithm 2, the elbow is determined as the most distant point from the line joining the first and last point of the curve. When all points are aligned, the top level is retained (i.e., main core or truss). When there are only two levels, the one giving the highest density is returned.

Algorithm 1: *dens* method

Input : core (or truss) decomposition of G
Output: set of keywords

D $\leftarrow$ empty vector of length n_{levels}
for $n \leftarrow 1$ **to** n_{levels} **do**
  $D[n] \leftarrow density(levels[n]$-core (or truss)$)$
end
$k_{best} \leftarrow levels[elbow(n, D[n])]$
return k_{best}-core (or truss) of G as keywords

Algorithm 2: *elbow*

Input : set of $|S| \geq 2$ points $S = \{(x_0, y_0), ..., (x_{|S|}, y_{|S|})\}$
Output: x_{elbow}

$line \leftarrow \{(x_0, y_0); (x_{|S|}, y_{|S|})\}$
if $|S| > 2$ **then**
  $distance \leftarrow$ empty vector of length $|S|$
  $s \leftarrow 1$
  for (x, y) *in* S **do**
    $distance[s] \leftarrow$ distance from (x, y) to $line$
    $s \leftarrow s + 1$
  end
  if $\exists! s \mid distance[s] = max(distance)$ **then**
    $x_{elbow} \leftarrow x \mid (x, y)$ is most distant from $line$
  else
    $x_{elbow} \leftarrow x_0$
  end
else
  $x_{elbow} \leftarrow x \mid y$ is maximum
end
return x_{elbow}

5.2 Inflexion

The Inflexion method (*inf* in what follows) is an empirically-grounded heuristic that relies on detecting changes in the variation of shell sizes (where size denotes the number of nodes). Recall from subsection 3.3 than the k-shell is the part of the k-core (or truss) that does not survive in the $(k+1)$-core (or truss), that is, the subgraph of G induced by the nodes with core (or truss) number exactly equal to k. In simple terms, the *inf* rule-of-thumb consists in going down the hierarchy as long as the shells in-

crease in size, and stopping otherwise. More precisely, *inf* is implemented as shown in Algorithm 3, by computing the consecutive differences in size across the shells and selecting the first positive point before the drop into the negative half (see Figure 1c). If no point satisfies this requirement, the main core (or truss) is extracted.

Algorithm 3: *inf* method

Input : core (or truss) decomposition of G
Output: set of keywords

CD $\leftarrow$ empty vector of length $n-1$
for $n \leftarrow 1$ **to** $(n_{levels}-1)$ **do**
  $k_l \leftarrow levels[n+1]$; $k_u \leftarrow levels[n]$
  $CD[n] \leftarrow$ size$\big(k_l$-shell$\big)-$size$\big(k_u$-shell$\big)$
end
if $\exists n \mid (CD[n+1] < 0 \wedge CD[n] > 0)$ **then**
  $n_{best} \leftarrow n$
else
  $n_{best} \leftarrow 1$
end
$k_{best} \leftarrow levels[n_{best}]$
return k_{best}-core (or truss) as keywords

Note that both *dens* and *inf* enjoy the same adaptability as the main core retention method of (Rousseau and Vazirgiannis, 2015) explained in Section 4, since the sizes of *all* the subgraphs in the hierarchy suit the structure of the graph-of-words.

5.3 CoreRank

Techniques based on retaining the k_{best}-core (or truss), such as *dens* and *inf* previously described, are better than retaining only the top level but lack flexibility, in that they can only select an entire batch of nodes at a time. This is suboptimal, because of course not all the nodes in a given group are equally good. To address this issue, our proposed CoreRank method (CR in what follows) converts nodes core (or truss) numbers into scores, ranks nodes in decreasing order of their scores, and selects the top $p\%$ nodes (CRP) or the nodes before the elbow in the scores curve (CRE). Note that for simplicity, we still refer to the technique as CR even when dealing with truss numbers.

Flexibility is obviously improved by decreasing granularity from the *subgraph* level to the *node* level. However, to avoid going back to the limitations of TR (absence of cohesiveness considerations), it is crucial to decrease granularity while retaining as much of the desirable information encoded by degeneracy as possible. To accomplish this task, we followed (Bae and Kim, 2014) and assigned to each node the sum of the core (or truss) numbers of its neighbors.

Our CRE method is outlined in Algorithm 4, where $N(v)$ denotes the set of neighbors of vertex v, and $number(v)$ is the core (or truss) number of v. CRP implements the exact same strategy, the only difference being $n_{best} \leftarrow round(|V| * percentage)$ at step 8 (where $percentage$ is a real number between 0 and 1).

Algorithm 4: CRE method

Input : core (or truss) decomposition of G
Output: set of keywords

CR $\leftarrow$ empty vector of length $|V|$
for $n \leftarrow 1$ **to** $|V|$ **do**
  $v \leftarrow V[n]$
  $CR[n] \leftarrow \sum_{u \in N(v)} number(u)$
  $name(CR[n]) \leftarrow label(v)$
end
sort CR in decreasing order
$n_{best} \leftarrow elbow(n, CR[n])$
return $names(CR[1 : n_{best}])$ as keywords

6 Experimental Setup

6.1 Baseline methods

TextRank (TR). We used as our first benchmark the system of (Mihalcea and Tarau, 2004) discussed in Section 4. For the sake of fair comparison with our CRE and CRP methods, we considered two variants of TR that respectively retain nodes based on both the elbow (TRE) and percentage criteria (TRP).

Main. Our second baseline is the main core retention technique of (Rousseau and Vazirgiannis, 2015), also described in Section 4. This method is referred to as *main* in the remainder of this paper.

6.2 Datasets

To evaluate performance, we used three standard, publicly available datasets featuring documents of

various types and sizes. Figure 3 shows the distributions of document size and manually assigned keywords for each dataset.

The **Hulth2003**[1] (Hulth, 2003) dataset contains abstracts drawn from the Inspec database of physics and engineering papers. Following our baselines, we used the 500 documents in the validation set and the "uncontrolled" keywords assigned by human annotators. The mean document size is 120 words and on average, 21 keywords (in terms of unigrams) are available for each document.

We also used the training set of **Marujo2012**[1], containing 450 web news stories of about 440 words on average, covering 10 different topics from art and culture to business, sport, and technology (Marujo et al., 2012). For each story, the keyphrases assigned by at least 9 out of 10 Amazon Mechanical Turkers are provided as gold standard. After splitting the keyphrases into unigrams, this makes for an average of 68 keywords per document, which is much higher than for the two other datasets, even the one comprising long documents (Semeval, see next).

The **Semeval**[2] dataset (Kim et al., 2010) offers parsed scientific papers collected from the ACM Digital Library. More precisely, we used the 100 articles in the test set and the corresponding author-and-reader-assigned keyphrases. Each document is approximately 1,860 words in length and is associated with about 24 keywords.

Notes. In Marujo2012, the keywords were assigned in an extractive manner, but many of them are verbs. In the two other datasets, keywords were freely chosen by human coders in an abstractive way and as such, some of them are not present in the original text. On these datasets, reaching perfect recall is therefore impossible for our methods (and the baselines), which by definition all are extractive.

6.3 Implementation

Before constructing the graphs-of-words and passing them to the keyword extraction methods, we performed the following pre-processing steps:

Stopwords removal. Stopwords from the SMART information retrieval system[3] were discarded.

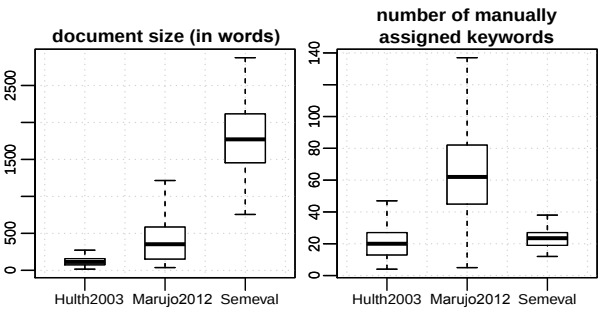


Figure 3: Basic dataset statistics.

Part-of-Speech tagging and screening using the `openNLP` (Hornik, 2015) R (R Core Team, 2015) implementation of the Apache OpenNLP Maxent POS tagger. Then, following (Mihalcea and Tarau, 2004), only nouns and adjectives were kept. For Marujo2012, as many gold standard keywords are verbs, this step was skipped (note that we did experiment with and without POS-based screening but got better results in the second case).

Stemming with the R `SnowballC` package (Bouchet-Valat, 2014) (Porter's stemmer). Gold standard keywords were also stemmed.

After pre-processing, graphs-of-words (as described in Section 2) were constructed for each document and various window sizes (from 3, increasing by 1, until a plateau in scores was reached). We used the R `igraph` package (Csardi and Nepusz, 2006) to write graph building and weighted k-core implementation code. For K-truss, we used the C++ implementation offered by (Wang and Cheng, 2012).

Finally, for TRP and CRP, we retained the top 33% keywords on Hulth2003 and Marujo2012 (short and medium size documents), whereas on Semeval (long documents), we retained the top 15 keywords. This is consistent with our baselines. Indeed, the number of manually assigned keywords increases with document size up to a certain point, and stabilizes afterwards.

The code of the implementation and the datasets can be found here[4].

[1] `https://github.com/snkim/AutomaticKeyphraseExtraction`

[2] `https://github.com/boudinfl/centrality_measures_ijcnlp13/tree/master/data`

[3] `http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop`

[4] `https://github.com/Tixierae/EMNLP_2016`

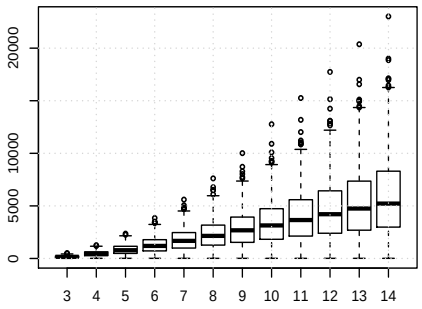


Figure 4: Triangle count versus window size (Hulth2003).

	precision	recall	F1-score
dens	**48.79**	**72.78**	**56.09***
inf	48.96	72.19	55.98*
CRP	61.53	38.73	45.75
CRE	65.33	37.90	44.11
main†	51.95	54.99	50.49
TRP†	65.43	41.37	48.79
TRE†	71.34	36.44	45.77

Table 1: Hulth2003, K-truss, $W = 11$.
*statistical significance at $p < 0.001$ with respect to all baselines.
†baseline systems.

	precision	recall	F1-score
dens	**47.62**	**71.46**	**52.94***
inf	53.88	57.54	49.10*
CRP	54.88	36.01	40.75
CRE	63.17	25.77	34.41
main†	64.05	34.02	36.44
TRP†	55.96	36.48	41.44
TRE†	65.50	21.32	30.68

Table 2: Marujo2012, k-core, $W = 13$.
*statistical significance at $p < 0.001$ with respect to all baselines.
†baseline systems.

7 Experimental Results

7.1 Evaluation

We computed the standard precision, recall, and F-1 measures for each document and averaged them at the dataset level (macro-averaging).

7.2 Precision/Recall trade-off

As shown in Figure 5, our methods *dens* and *inf* outperform the baselines by a wide margin on the datasets containing small and medium size documents (Hulth2003 and Marujo2012). As expected, this superiority is gained from a drastic improvement in recall, for a comparatively lower precision loss. TR and *main* exhibit higher precision than recall, which is in accordance with (Rousseau and Vazirgiannis, 2015). The same observation can be made for our CR method. For TR, the unbalance is more severe on the Hulth2003 and Marujo2012 datasets (short/medium documents) when the elbow method is used (TRE), probably because it tends to retain only a few nodes. However, on Semeval (long documents), using the elbow method (TRE) gives the best trade-off between precision and recall. For CR, still on Semeval, using the elbow method (CRE) even gives better recall than precision.

Overall, compared to the baselines, the unbalance between precision and recall for our methods is less extreme or equivalent. On the Marujo2012 dataset for instance, our proposed *inf* method is almost perfectly balanced and ranks second (significantly better than all baselines).

7.3 Impact of window size

The performance of k-core does not dramatically increase with window size, while K-truss exhibits a surprising "cold start" behavior and only begins to kick-in for sizes greater than 4-5. A possible explanation is that the ability of K-truss (which is triangle-based) to extract meaningful information from a graph depends, up to a certain point, on the amount of triangles in the graph. In the case of graph-of-words, the number of triangles is positively correlated with window size (see Figure 4). It also appears that document size (i.e., graph-of-words structure) is responsible for a lot of performance variability. Specifically, on longer documents, performance plateaus at higher window sizes.

7.4 Best models comparison

For each dataset, we retained the degeneracy technique and window size giving the absolute best performance in F1-score, and compared all methods under these settings (see Tables 1–3). We tested for statistical significance in macro-averaged F1 scores using the non-parametric version of the t-test, the Mann-Whitney U test[5].

On Hulth2003 and Marujo2012 (short and medium size documents), our methods *dens* and *inf* strongly and significantly outperform all baselines, with respective absolute improvements of more than 5.5% with respect to the best performing baseline

[5] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/wilcox.test.html

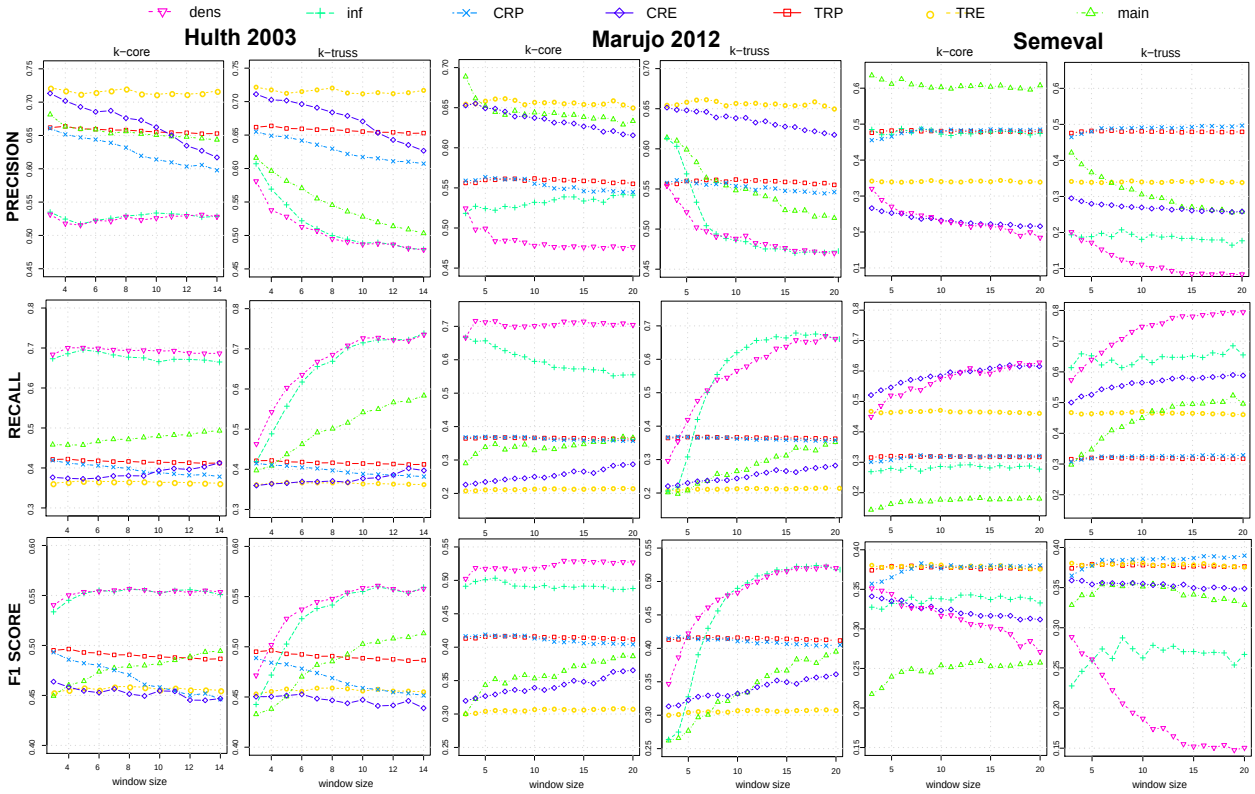


Figure 5: Impact of window size on performance.

	precision	recall	F1-score
dens	8.44	79.45	15.06
inf	17.70	65.53	26.68
CRP	**49.67**	**32.88**	**38.98***
CRE	25.82	58.80	34.86
main†	25.73	49.61	32.83
TRP†	47.93	31.74	37.64
TRE†	33.87	46.08	37.55

Table 3: Semeval, K-truss, $W = 20$.
*statistical significance at $p < 0.001$ w.r.t. *main*. †baseline systems.

(*main*). On Semeval, which features larger pieces of text, our CRP technique improves on TRP, the best performing baseline, by more than 1 %, altough the difference is not statistically significant. However, CRP is head and shoulders above *main*, with an absolute gain of 6%. This suggests that converting the cohesiveness information captured by degeneracy into ranks may be valuable for large documents.

Finally, the poor performance of the *dens* and *inf* methods on Semeval (Table 3) might be explained by the fact that these methods are only capable of selecting an entire batch of nodes (i.e., subgraph-of-words) at a time. This lack of flexibility seems to become a handicap on long documents for which the graphs-of-words, and thus the subgraphs corresponding to the k-core (or truss) hierarchy levels, are very large. This analysis is consistent with the observation that conversely, approaches that work at a finer granularity level (node level) prove superior on long documents, such as our proposed CRP method which reaches best performance on Semeval.

8 Conclusion and Future Work

Our results provide empirical evidence that *spreading influence* may be a better "keywordness" metric than *eigenvector* (or *random walk*)-based criteria. Our CRP method is currently very basic and leveraging edge weights/direction or combining it with other scores could yield better results. Also, more meaningful edge weights could be obtained by merging local co-occurrence statistics with external semantic knowledge offered by pre-trained word embeddings (Wang et al., 2014). The direct use of density-based objective functions could also prove valuable.

References

Joonhyun Bae and Sangwook Kim. 2014. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications*, 395:549–559.

Vladimir Batagelj and Matjaž Zaveršnik. 2002. Generalized cores. *arXiv preprint cs/0202039*.

Milan Bouchet-Valat, 2014. *SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library*. R package version 0.5.1.

Asli Celikyilmaz and Dilek Hakkani-Tür. 2011. Discovery of topically coherent sentences for extractive summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 491–499. Association for Computational Linguistics.

Yun-Nung Chen, Yu Huang, Hung-Yi Lee, and Lin-Shan Lee. 2012. Unsupervised two-stage keyword extraction from spoken documents by topic coherence and support vector machine. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5041–5044. IEEE.

Janara Christensen, Stephen Soderland Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *HLT-NAACL*, pages 1163–1173. Citeseer.

Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, page 16.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Vince Grolmusz. 2015. A note on the pagerank of undirected graphs. *Information Processing Letters*, 115(6):633–634.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kurt Hornik, 2015. *openNLP: Apache OpenNLP Tools Interface*. R package version 0.2-5.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 216–223. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. 2010. Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888–893.

Fragkiskos D Malliaros and Konstantinos Skianis. 2015. Graph-based term weighting for text categorization. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1473–1479. ACM.

Fragkiskos D. Malliaros, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. 2016a. Core decomposition in graphs: concepts, algorithms and applications. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT*, pages 720–721.

Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. 2016b. Locating influential nodes in complex networks. *Scientific Reports*, 6:19307.

Luis Marujo, Anatole Gershman, Jaime Carbonell, Robert Frederking, and Jo ao P. Neto. 2012. Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization. In *Proceedings of LREC 2012*. ELRA.

Polykarpos Meladianos, Giannis Nikolentzos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2015. Degeneracy-based real-time sub-event detection in twitter stream. In *Ninth International AAAI Conference on Web and Social Media (ICWSM)*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

R Core Team, 2015. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Conference on Information & Knowledge Management (CIKM)*, pages 59–68. ACM.

François Rousseau and Michalis Vazirgiannis. 2015. Main core retention on graph-of-words for single-document keyword extraction. In *Advances in Information Retrieval*, pages 382–393. Springer.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *ACL*, volume 15, page 107.

Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks*, 5(3):269–287.

Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment*, 5(9):812–823.

Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, page 39.

Predicting the Relative Difficulty of Single Sentences With and Without Surrounding Context

Elliot Schumacher
Carnegie Mellon University
eschumac@cs.cmu.edu

Maxine Eskenazi
Carnegie Mellon University
max@cs.cmu.edu

Gwen Frishkoff
University of Oregon
gfrishkoff@gmail.com

Kevyn Collins-Thompson
University of Michigan
kevynct@umich.edu

Abstract

The problem of accurately predicting relative reading difficulty across a set of sentences arises in a number of important natural language applications, such as finding and curating effective usage examples for intelligent language tutoring systems. Yet while significant research has explored document- and passage-level reading difficulty, the special challenges involved in assessing aspects of readability for single sentences have received much less attention, particularly when considering the role of surrounding passages. We introduce and evaluate a novel approach for estimating the relative reading difficulty of a set of sentences, with and without surrounding context. Using different sets of lexical and grammatical features, we explore models for predicting pairwise relative difficulty using logistic regression, and examine rankings generated by aggregating pairwise difficulty labels using a Bayesian rating system to form a final ranking. We also compare rankings derived for sentences assessed with and without context, and find that contextual features can help predict differences in relative difficulty judgments across these two conditions.

1 Introduction

The reading difficulty, or *readability*, of a text is an estimate of linguistic complexity and is typically based on lexical and syntactic features, such as text length, word frequency, and grammatical complexity (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005; Kidwell et al., 2011; Kanungo and Orr, 2009). Such estimates are often expressed as age- or grade-level measures and are useful for a range of educational and research applications. For example, instructors often wish to select stories or books that are appropriately matched to student grade level.

Many measures have been designed to calculate readability at the document level (e.g., for web pages, articles, or books) (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005), as well as the paragraph or passage level (Kidwell et al., 2011; Kanungo and Orr, 2009). However, much less work has attempted to characterize the readability of single sentences (Pilán et al., 2014). This problem is challenging because single sentences provide less data than is typically required for reliable estimates, particularly for measures that rely on aggregate statistics.

The absence of reliable single-sentence estimates points to a gap in natural language processing (NLP) research. Single sentences are used in a variety of experimental and NLP applications: for example, in studies of reading comprehension. Because readability estimates have been shown to predict a substantial portion of variance in comprehension of different texts, it would be useful to have measures of single-sentence readability. Thus, one aim of the current study was to estimate the *relative readability* of single sentences with a high degree of accuracy. To our knowledge, general-purpose methods for computing such estimates for native language (L1) readers have not been developed, and thus our goal was to develop a method that would characterize sentence-level difficulty for that group.

The second aim is to compare the readability of single sentences in isolation with the readability of

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1871–1881,
Austin, Texas, November 1-5, 2016.

these same sentences embedded in a larger context (e.g., paragraph, passage, or document). When a single sentence is extracted from a text, it is likely to contain linguistic elements, such as anaphora (e.g., "he" or "the man"), that are semantically or syntactically dependent on surrounding context. Not surprisingly, sentences that contain these contextual dependencies take more effort to comprehend: an anaphoric noun phrase, or NP (e.g., "he"), automatically triggers the need to resolve reference, typically by understanding the link between the anaphor and a full NP from a previous sentence (e.g., "John" or "The man that I introduced you to at the party last night" (Perfetti and Frishkoff, 2008). In general, studies have shown a link between reading comprehension and the presence of such cross-sentence relationships in the text (McNamara, 2001; Liederholm et al., 2000; Voss and Silfies, 1996). This implies that the very notion of readability at the sentence level may depend on context as well as word- and sentence-level features. Therefore, it is important to compare readability estimates for single sentences that occur in isolation with those that occur within a larger passage, particularly if the target sentence contains coreferences, implied meanings, or other dependencies with its context.

To address these aims, the present study first conducted two crowdsourcing experiments. In the first, 'sentence-only' experiment, workers were asked to judge which of two "target" sentences they thought was more difficult. In the second, 'sentence-in-passage' experiment, another group of workers was presented with the same target sentences that were used in the first experiment. However, in the second experiment, target sentences were embedded in their original contexts.

Next, we analyzed these judgments of relative readability for each condition (sentence-only versus sentence-in-passage) by developing models for predicting pairwise relative difficulty of sentences. These models used a rich representation of target sentences based on a combination of lexical, syntactic, and discourse features. Significant differences were found in readability judgments for sentences with and without their surrounding context. This demonstrates that discourse-level features (i.e., features related to coherence and cohesion) can affect the readability of single sentences.

2 Related Work

Recent approaches to estimating readability have used a variety of linguistic features and prediction models (Collins-Thompson, 2014). The Lexile Framework (Stenner, 1996) uses word frequency estimates in a large corpus as a proxy for lexical difficulty, and sentence length as a grammatical feature. Methods based on statistical machine learning, such as the reading difficulty measures developed by Collins-Thompson and Callan (Collins-Thompson and Callan, 2004) and (Schwarm and Ostendorf, 2005) used a feature set based on language models. Later work (Heilman et al., 2008) incorporated grammatical features by parsing the sentences in a text, and creating subtrees of one- to three-level depth as separate features. Such features allow more detailed, direct analysis of the sentence structure itself instead of traditional proxies for syntactic complexity likes sentence length. The linguistic features proposed in these works capture specific aspects of language difficulty applied at the document level, whereas our work investigates the effectiveness of these feature types for characterizing aspects of difficulty at the sentence level.

Methods have been proposed to measure the readability of shorter portions of text (e.g. typically less than 100 words), including sentences. The approach most similar to ours is the prediction of relative sentence difficulty (with associated readability ranking) for the deaf introduced by Inui et al. (2001). That work focused on effective morphosyntactic features for that target population with an SVM binary classifier, whereas our approach (1) is intended for a broader population of L1 learners and thus explores the effectiveness of adding a rich, lexically-derived feature set, (2) uses a logistic regression model to estimate class probabilities and interprets the results of that model, compared to applying an SVM without interpretation to obtain a binary label, (3) examines differences in predicting sentence difficulty both in and out of passage context, and (4) creates and uses a new dataset based on a crowdsourced approach, using hundreds of non-experts to gather thousands of pairwise preferences, compared to a questionnaire deployed to a small number of experts. In other domains, a model was proposed to predict the readability of short web summaries in Kanungo and Orr 2009. In Kidwell et al. (2011), , a set of Age of

Acquisition estimates for individual words, representing the lexical component of difficulty, was used to predict the grade levels of passages. Some approaches have explored the classification of specific aspects of sentences, as opposed to reading difficulty classification. For example, (Pilán et al., 2014) classified individual sentences that would be understood by second-language learners. Another work (Kilgarriff et al., 2008) identified sentences that would be good dictionary examples by looking for specific desirable features. Davenport et al. 2014 used a traditional method of readability (Flesch-Kincaid), within the larger context of exploring relationships between the difficulty of tweets in a geographic area and demographics. Research in text simplification has applied sentence-level models of difficulty as part of simplification-based optimization objectives. For example, Woodsend and Lapata (2011) use word and syllable count as proxy features for sentence difficulty when implicitly comparing different simplified variants of a sentence.

Other approaches have considered the relationship of reading difficulty to structures within in the whole text. These relationships can include the number of coreferences present in a text. Coh-Metrix (Graesser et al., 2011) measures text cohesiveness, accounting for both the reading difficulty of the text and other lexical and syntactic measures as well as a measure of prior knowledge needed for comprehension, and the genre of the text. Coh-Metrix uses co-reference detection as a factor in the cohesiveness of a text, typically at the document or passage level. Such cohesiveness factors account for the difficulty of constructing the mental representation of texts with more complex internal structure. TextEvaluator (Sheehan et al., 2013; Sheehan et al., 2014) is designed to help educators select materials for instruction. The tool includes several components in its evaluation of text, including narrativity, style, and cohesion, beyond traditional difficulty and is again at the whole document level. This approach illustrates that the difficulty of a text relies on the relationships within it. This motivates the need to consider context when measuring difficulty.

Generating reading difficulty rankings of longer texts from pairwise preferences has been performed in other contexts. Tanaka-Ishii et al. (2010) explored an approach for sorting texts by readability based on pairwise preferences. Later, Chen et al. (2013) also proposed a model to obtain passage readability ranking by aggregating pairwise comparisons made by crowdworkers. In De Clercq et al.(2014), pairwise judgments of whole passages were obtained from crowdworkers and were found to give comparable results in aggregate to those obtained from experts. A pairwise ranking of text readability was created in Pitler and Nenkova (2008) in which readability was defined by subjective questions asked to the reader after finishing the article, such as "How well-written is this article?". All of the above previous work was focused on ordering longer text passages, not single sentences as we do here.

Finally, research in the Machine Translation field has explored pairwise prediction of the best translation between two sentence options. For example, in Song and Cohn (2011), a pairwise prediction model was built using n-gram precision and recall, as well as function, content, and word counts. However, unlike pairwise prediction of difficulty, the prediction is done with respect to a reference sentence, or set of reference sentences.

3 Data Collection and Processing

We now describe methods used to create our dataset of sentences, to collect pairwise assessments of difficulty, and to aggregate these pairwise preferences into a complete ranking.

3.1 Data Set

The study sentences were drawn from a corpus combining the American National Corpus (Reppen et al., 2005), the New York Times Corpus (Sandhaus, 2008), and the North American News Text Corpus (McClosky et al., 2008). The domain of these corpora is largely news text, but also includes other topics, such as travel guides and other non-fiction. In total, this database contains 60,663,803 sentences that served as initial candidates. Sentences were filtered out that didn't include one of the 70 target words that the third author selected for a study on teaching vocabulary to 8-14 year-old students. Other sentences were removed based on length, keeping only sentences of between 6 and 20 words. Some sentences were removed due to the presence of one or more rare words. Finally, sentences were annotated with the surrounding document reading level, us-

ing a lexical readability model (Collins-Thompson and Callan, 2004).The data set gathered by (Collins-Thompson and Callan, 2004) was used in order to add to the amount of lower-level reading material in the collected corpora.

With these sentences, two crowdsourced tasks were prepared to gather pairwise assessments of sentence reading difficulty. In one task, the sentences were presented alone, outside of their original passage context. In the other task, the same sentences were presented within their original passage context. The objective was to generate two sets of pairwise comparisons of the readability of a sentence. In total, 120 sentence pairs were used for the first task and 120 passage pairs were used for the second. Each sentence was compared to five others, which created 300 comparisons in each task. The five sentences matched to each sentence were selected to ensure that pairs with a range of document level differences would be created. Within each type of pair, a random pair was selected.

There were several constraints when generating pairs for comparison. To allow for sentences to be taken from documents with a range of reading levels, sentences were selected evenly from documents at each reading level. From the twelve standard U.S. grade levels used in readability, each document was considered to be part of a bin consisting of two adjacent grade levels, such as grades 1 and 2, for example. Sentences were selected evenly from those bins.

Each sentence needed sufficient context to ensure there would be equivalent context for each item that would be compared, so only passages of sufficient size were included. To ensure passages were of similar length, only passages that had between 136 and 160 words were included. Contexts having at least two sentences before and after the sentence in question were strongly preferred. Each selected sentence was paired with one sentence from each of the other grade level bins. For example, a sentence from grade 1 would be paired with one sentence each from grade 3-4, 5-6, 7-8, 9-10, and 11-12. Finally, each pair of sentences was presented in AB and BA order. For each pair, there were seven worker decisions. There were 296 unique workers for the sentence-only task, and 355 for the sentence-in-passage task.

3.2 Crowdsourcing

Both of these tasks were carried out on the Crowdflower platform. The workers were first given instructions for each task, which included a description of the general purpose of the task. In the sentence-only task, workers were asked to select which of the two sentences was more difficult. In the sentence-within-passage task, workers were similarly asked to decide which underlined sentence was more difficult. The instructions for the latter requested that the workers make their judgment only on the sentence, not on the whole context. In both tasks, there was an option for "I don't know or can't decide". The workers were asked to make their decision based on the vocabulary and grammatical structure of the sentences. Finally, examples for each task were provided with explanations for each answer.

For each task, at least 40 gold standard questions were created from pairs of sentences that were judged to be sufficiently distinct from one another so that they could easily be answered correctly. For the sentence-in-passage task, several gold standard questions were written to verify that the instructions were being followed, since it was possible that a worker might judge the sentences based on the quality of the passage alone. These gold examples consisted of an easier sentence in a difficult passage compared with a difficult sentence within an easy passage. For each task, the worker saw three questions, including one gold standard question. A worker needed to maintain an 85% accuracy rating on gold standard questions to continue, and needed to spend at least 25 seconds per page, which contained 3 questions each.

A weighted disagreement rate was calculated for each worker. If a worker's response to a question differed from the most frequent answer to that question, the percentage of agreement was counted against the worker. If a worker, for the sentence-only task, had a disagreement rate (the weighted disagreement penalty divided by the total questions they answered) of 15% or higher, their contribution was removed from the data set (or 17% or higher for the sentence in passage task). The agreement for the sentence-in-passage task is lower than the sentence-only task (88.93% and 90.33% respectively), so the permitted disagreement level is higher for that task. This resulted in the removal of 5.7% and 4.5% of

pairwise judgments, respectively. For each question, there was an optional text form to allow workers to submit feedback. The sentence-only task paid 11 cents per page, and the sentence-in-passage task paid 22 cents per page.

3.3 Ranking Generation

Each task resulted in 4,200 pairwise preference judgments, excluding gold-standard answers. To aggregate these pairwise preferences into an overall ranking of sentences, we use a simple, publicly available approach evaluated by Chen et al. as being competitive with their own Crowd-BT aggregation method: the Microsoft Trueskill algorithm (Herbrich et al., 2007). Trueskill is a Bayesian skill rating system that generalized the well-known Elo rating system, in that it generates a ranking from pairwise decisions. As Trueskill's ranking algorithm depends on the order in which the samples are processed, we report the ranking as an average of 50 runs.

The judgments were not aggregated for each comparison. Instead, each of the judgments was treated individually. This allows Trueskill to consider the degree of agreement between workers, since a sentence judgment that has high agreement reflects a larger difference in ranking than one that has lower agreement. Each sentence was considered a player, and the winner between two, A or B, was the sentence considered most difficult. If a worker chose "I don't know or can't tell", it was considered a draw. The prediction resulting in "I don't know or can't tell" is rare; 2.2% of decisions in the sentence only task resulted in a draw, and 2.0% for sentences within passages. After processing each of the judgments, a rating can be built of sentences, ranked from least difficult to most difficult. We can compare the resulting rankings for the sentence-only task and the sentence-in-passage task to see the effect of context on relative sentence difficulty.

4 Modeling Pairwise Relative Difficulty

Our first step in exploring relative difficulty ordering for a set of sentences was to develop a model that could accurately predict relative difficulty for a single pair of sentences, corresponding to the pairwise judgements of relative difficulty we gathered from the crowd. We did this for both the sentence-only and the sentence-in-passage tasks. In predicting a pairwise judgment for the sentence-only task, the model uses only the sentence texts. In the model for the sentence-in-passage task, the Stanford Deterministic Coreference Resolution System (Raghunathan et al., 2010) is used to find coreference chains within the passage. From these coreference chains, sentences with references to and from the target sentence can be identified. If any additional sentences are found, these are used in a separate feature set that is included in the model; for all possible features, they are calculated for the target sentence, and separately for the additional sentence set.

Prior to training the final model, feature selection was done on random splits of the training data. Training data was used to fit a Random Forest Classifier, and based on the resulting classifier, the most important variables were selected using sklearn's feature importance method. The top 2% of the features (or 1% for the sentence-in-passage with coreference, since the feature set size is doubled) were selected automatically. This resulted in a feature size of 40-50 features. We implemented our models using scikit-learn (Pedregosa et al., 2011) in Python.

The resulting features were used to train a Logistic Regression model. While other prediction models such as Support Vector Machines have been applied to relative readability prediction (Inui and Yamamoto, 2001), we chose Logistic Regression due to its ability to provide estimates of class probabilities (which may be important for reliability when deploying a system that recommends high-quality items for learners), its connection to the Rasch psychometric model used with reading assessments (Ehara et al., 2012), and the interpretable nature of the resulting parameter weights. Since a given feature has a value for sentence A and B, if a feature was selected for only Sentence A or B, the feature for the other sentence was also added. We used the NLTK library (Bird et al., 2009) to tokenize the sentence for feature processing.

At the sentence level, the familiarity of the words is a significant factor to consider in any judgment of difficulty. The grammatical structure of a sentence is also important to consider: if the sentence uses a more familiar structure, it is likely to be considered less difficult than a sentence with more unusual structure. We thus identified two groups of potential

features: lexical and grammatical, described below.

4.1 Lexical Features

For lexical features, based partly on the work of (Song and Cohn 2011) we included the percentage of non-stop words (using NLTK list), the total number of words and the total number of characters as features. We included the percentage of words in the text found in the Revised Dale-Chall word list (Dale and Chall, 2000) to capture the presence of more difficult words in the sentence.

Because sentences that contain rarer sequences of words are likely to be more difficult, and the likelihood of the sentence based on a large corpus should reflect this, we included the n-gram likelihood of each sentence, over each of 1-5 n-grams, as a feature. The Microsoft WebLM service (Wang et al., 2010) was used to calculate the n-gram likelihood.

In the field of psycholinguistics, Age of Acquisition (AoA) refers to the age at which a word is first learned by a child. A database of 51,715 words collected by (Kuperman et al., 2012) provides a rich resource for use in reading difficulty measures. With this dataset, we computed several additional features: the average, maximum, and standard deviation of the aggregated AoA for all words in a sentence that were present in the database. Since the data set also includes the number of syllables in each word, and as (Kincaid et al., 1975) proposes that words with more syllables are more difficult, we also included the average and maximum syllable count as potential features.

4.2 Syntactic Features

We parsed each sentence in the data set using the BLLIP Parser (Charniak and Johnson, 2005), which includes a pre-trained model built on the Wall Street Journal Corpus. This provided both a syntactic tree and part of speech tags for the sentence. As Part of Speech tagging is often used as a high-level linguistic feature, we computed percentages for each PoS tag present, since the percentages might vary between difficult sentences and easier sentences. The percentage for each Part of Speech tag is defined as the number of times a certain tag occurred, divided by the total tags. The diversity of part of speech tags was used since this might vary between difficult and easier sentences.

Using the syntactic tree provided by the parser, we obtained the likelihood of the parse, and the likelihood produced by the re-ranker, as syntactic features. If a sentence parse has a comparatively high likelihood, it is likely to be a more common structure and thus more likely to be easier to read. The length and height of the parse were also included as features, since each of these could reflect the difficulty of the parse. Including the entire parse of the sentence would create too much sparsity since syntactic parses vary highly from sentence to sentence. Therefore, as was done in (Heilman et al., 2008), subtrees of depth one to three were created from the syntactic parse, and were added as features. This creates a smaller feature set, and one that can potentially model specific grammatical structures that are associated with a specific level of difficulty.

5 Pairwise Difficulty Prediction Results

The performance of the logistic regression models trained with different feature sets, for each task, is shown in Table 1. We reported the mean and standard deviation of the accuracy of each model over 200 randomly selected training and testing splits. Each test set consisted of 20% of the data, and contained 60 aggregate pairs, all of which are sentences (24 in total) that were not present in the training data. The test sets for the sentence-in-passage and sentence-only task contain the same sentence pairs, but the individual judgements are different.

For comparison, an oracle is included that represents the accuracy a model would achieve if it made the optimal prediction for each aggregate pair. Due to disagreement within the crowd, the oracle cannot reach 100% accuracy. For example, for some pair A and B, if 10 workers selected A as the more difficult sentence, and 4 workers selected B, the oracle's prediction for that pair would be that that A is more difficult. The judgments of the four workers that selected B would be counted as inaccurate, since the feature set is the same for the judgments with A and the judgments with B. Therefore, the oracle represents the highest accuracy a model can achieve, consistent with the provided labels, using the features provided.

Examining the results in Table 1, we find the best performing configuration, Model B, used all features as candidates. The exact number of features selected

Model	Sentence Only			In Passage, With Coref			In Passage, No Coref		
	Acc.	S.D.	p-value	Acc.	S.D.	p-value	Acc.	S.D.	p-value
Oracle (A)	90.13%	2.71%	—	87.81%	1.84%	—	87.81%	1.84%	—
All Features (B)	**84.69%**	3.46%	0.01 ↓	**81.66%**	3.17%	0.005 ↓	**81.91%**	3.27%	← 0.04
AoA + Parse L. (C)	**84.33%**	3.13%	0.001 ↓	**81.27%**	3.93%	0.001 ↓	**80.84%**	3.61%	← 0.001
AoA (D)	**79.62%**	2.71%	0.001 ↓	**79.72%**	2.86%	0.001 ↓	**78.99%**	2.58%	← 0.001
Strat. Random	50.28%	1.68%	—	50.31%	2.01%	—	50.31%	2.01%	—

Table 1: Mean and standard deviation of accuracy on 200 randomized samples of 20% held out data. 'With coref' indicates coreference features were used. The arrow indicates which immediately adjacent accuracy result is used for p-value comparison, e.g. Model B sentence-only is compared to model C sentence-only, and model B passage, no coref is compared to model B passage, with coref.

varied depending on the task. However, the simplest model, the Age of Acquisition model (D) consisting of the average, standard deviation, and maximum AoA features (sentence-only: 6 features, sentence-in-passage: 12 features) performed well, achieving over 78% accuracy on all tasks, showing that most of the relative difficulty signal at the sentence level can be captured with a few lexical difficulty features. The Age of Acquisition + Parse Likelihood model (C) consists of all Age of Acquisition features, plus the likelihood of the parse (sentence-only: 10 features, sentence-in-passage: 20 features)[1].

To assess the contribution of different features to the model prediction, feature group importances are reported in Table 2. As features for a given group are often highly correlated with each other, such as in Age of Acquisition, the importance is calculated for feature groups. Based on the method described for Model B, each feature group is removed from consideration in the model, and the resulting error rate from Model B is used to calculate an importance measure. The most important feature is normalized to have a value of 1.0, with the rest being relative to the difference in error rate from the original model, averaged across splits.

These prediction results show that relative reading difficulty can be predicted for sentence pairs with high accuracy, even with fairly simple feature sets. In particular, the results for AoA model D, which uses a small number of targeted features, are competitive with the best model B that relies on a much larger feature set. The addition of coreference features did result in small but significant changes in the accuracy of the sentence-in-passage task, although in one case the accuracy was reduced with coreference features.

[1]The p-value for each accuracy measurement compares its significance, using a paired t-test, to the neighboring model in the direction of the arrow. For example, the sentence-only Model B is compared to sentence-only Model A.

Sentence Only		Sentence in Passage (with Coref)	
Feature	Imp.	Feature	Imp.
Age of Acq.	1.00	Age of Acq.	1.00
Part of Speech	0.28	Syllables	0.27
Syn. Score	0.22	Part of Speech	0.23
Syn. Other	0.21	Syn. Tree	0.18
Syllables	0.19	Dale Chall	0.17
Ngram L.	0.19	Content Word %	0.17
Word Len.	0.17	Word Len.	0.16
Dale Chall	0.16	Syn. Other	0.16
Content Word %	0.15	Syn. Score	0.12
Syn. Tree	0.12	Ngram L.	0.10

Table 2: Relative feature importance for Model B. Feature importance is the increase in absolute error with a specific feature group removed, averaged across cross-validation folds used for Table 1, and normalized relative to the most informative feature. For Sentence in Passage, feature groups include coreference features.

	Value
Avg. Abs. Diff	9.3
Avg. Abs. Std Dev	7.7
Pearson's correlation	**0.94***
Spearman's correlation	**0.94***

Table 3: Comparison of rankings generated with and without passage. Asterisk * indicates $p < 0.0001$.

% Diff	Pearson	p-val.	Spearman	p-val.
Reranker	**-0.33**	0.0002	**-0.29**	0.001
Parser	**-0.33**	0.0002	**-0.28**	0.002

Table 4: Correlation between difference in rank and percentage difference in features.

6 Ranking Results

Using the pairwise aggregation method described in Sec. 3.3, we ranked sentences by relative difficulty for both sentence-only and sentence-in-passage tasks. By observing how the overall rank or-

	Sentence				Sentence-In-Passage			
	All		Gold Only		All		Gold Only	
	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
AoA Avg	**0.6971**	**0.7151**	**0.7366**	**0.7598**	**0.7155**	**0.7356**	**0.6220**	**0.6482**
AoA Std Dev	**0.6366**	**0.6596**	**0.7074**	**0.7385**	**0.6779**	**0.7023**	**0.5742**	**0.5825**
AoA Max	**0.7084**	**0.6814**	**0.8036**	**0.7877**	**0.7408**	**0.7155**	**0.6215**	**0.6127**
Parser L.	**-0.4942**	**-0.5297**	**-0.4605***	**0.4920**	**-0.4172**	**-0.4465**	**-0.5099**	**-0.5157**
Reranker L.	**-0.4923**	**-0.5280**	**-0.4574***	**-0.4751**	**-0.4139**	**-0.4450**	**-0.4969**	**-0.4879***

Table 5: Sentence-Only and Sentence-In-Passage Ranking Correlation with Individual Features. Gold indicates only gold-standard questions were used to build ranking. All correlations have $p < 0.0001$ except those with an asterisk *, which have $p < 0.001$.

dering of sentences changes across these conditions, we can identify differences in how workers judged the relative difficulty of sentences with and without context.

6.1 Rank Differences

We report differences in ranking in terms of mean and standard deviation of the absolute difference in rank index of each sentence across the two rankings, along with Pearson's coefficient and Spearman's rank order coefficients. Comparisons between the rankings for each task are shown in Table 3.

In comparing crowd-generated rankings for the sentence-only and sentence-in-passage task, the results show a statistically significant aggregate difference in how the crowd ranks sentence difficulty with and without the surrounding passage. While the correlation between the two rankings is high, and the average normalized change in rank position is 7.7%, multiple sentences exhibited a large change in ranking. For example, the sentence '*As a result, the police had little incentive to make concessions.*' was ranked significantly easier when presented out of context than when presented in context (rank change: -30 positions). For that example, the surrounding passage explained the complex political environment referred to indirectly in that sentence.

6.2 Feature Correlation with Rank Differences

To examine why sentences may be ranked as more or less difficult, depending on the context, we examined the correlation between a sentence's change in rank (Sentence-Only Ranking minus the Sentence-in-Passage ranking) and the normalized difference in feature values between the sentence representation and the remaining context representation. We found that percentage change in parser and reranker likelihoods had the most significant correlation (-0.33) with ranking change, as shown in Table 4.

To interpret this result, note that the parser and reranker likelihood represent the probability the parser and reranker models assign to the syntactic parse produced by the sentence. In other words, they are a measure of how likely it is that the sentence structure occurs, based on the model's training data. If the difficulty of the sentence-in-passage is ranked higher than the sentence alone, this correlates with the target sentence having a syntactic structure with higher likelihood than the average of the surrounding sentence structures. This means that if a sentence that has a frequently-seen syntactic structure is in a passage with sentences that have less common structures, the sentence within the passage is more likely to be judged as more difficult. The reverse is also true: if a sentence that has a more unusual syntactic structure is in a passage with sentences with more familiar structures, the sentence without the surrounding passage is more likely to be ranked as more difficult.

We also examined the rank correlation of crowd-generated rankings with rankings produced by sorting sentences based on the value of individual features. In addition to the full rankings, we constructed a ranking produced only by the gold standard examples, denoted *Gold Only* and included this in the comparison. The gold standard questions consist of examples constructed by the authors to have a clear relative difficulty result. The rank correlations are shown in Table 5 for both tasks.

The reasons for discrepancies in relative difficulty assessment between the sentence-only and sentence-in-passage conditions require further exploration. While the correlation between the percentage change in probability of the parse and the difference in ranking is significant, it is not large. It does indicate that despite judges being explicitly

	Crowd	
	Pearson	Spearman
Expert label	**0.85**	**0.84**
Document-based label	**0.70**	**0.70**

Table 6: Correlation between sentence readability labels and crowd-generated ranking, for expert (sentence-level) and document-based labels (from document readability prediction). All correlations have $p < 0.0001$.

told to only consider the sentence, the properties of the surrounding passage may indeed influence the perceived relative difficulty of the sentence.

6.3 Review of Data

The pairwise prediction results indicate that a large proportion of the crowdsourced pair orderings can be decided using vocabulary features, due to the strong performance of the Age of Acquisition features. To identify the relative importance of vocabulary and syntax in our data, we reviewed each pair and judged whether the sentence's syntax or vocabulary, or the combination of both, were needed to correctly predict the more difficult sentence. For many pairs, either syntax or vocabulary could be used to correctly predict the more difficult sentence since each factor indicated the same sentence was more difficult. We found that 19% of pairs had only a vocabulary distinction, and 65% of pairs could be judged correctly either by vocabulary or syntax. Therefore, 84% of pairs could be judged using vocabulary, which explains the high performance of the Age of Acquisition features.

The level of a sentence's source document was used as a proxy for the sentence's grade level when building the pairs. To build a sentence-level gold standard for this dataset, we asked a teacher with a Master of Education with a Reading Specialist focus and 30 years of experience in elementary and high school reading instruction, to identify the grade level of each sentence. This expert was asked to assign either a single grade level or a range of levels to each of the 120 sentences. From this, an expert ranking was created, using the midpoint of each expert-assigned range. The correlation between the expert sentence ranking and the crowd ranking can be seen in Table 6, reinforcing the finding that crowdsourced judgments can provide an accurate ranking of difficulty (De Clercq et al., 2014).

7 Conclusion

Using a rich sentence representation based on lexical and syntactic features leveraged from previous work on document-level readability, we introduced and evaluated several models for predicting the relative reading difficulty of single sentences, with and without surrounding context. We found that while the best prediction performance was obtained by using all feature classes, simpler representations based on lexical features such as Age of Acquisition norms were effective. The accuracy achieved by the best prediction model came within 6% of the oracle accuracy for both tasks.

Many of the features identified had a high correlation with the rankings produced by the crowd. This indicates that these features can be used to build a model of sentence difficulty. With the rankings built from crowdsourced judgments on sentence difficulty, small but significant differences were found in how sentences are ranked with and without the surrounding passages. This result suggests that properties of the surrounding passage of a sentence can change the perceived difficulty of a sentence.

In future work, we plan to increase the number of sentences in our data set, so that additional more fine-grained features might be considered. For example, weights for lexical features could be more accurately estimated with more data. Our use of the crowd-based labels was intended to reduce noise in the ranking analysis, but we also intend to use the pairwise predictions produced by the logistic model as the input to the aggregation model, so that rankings can be obtained for previously unseen sentences in operational settings. Another goal is to obtain absolute difficulty labels for sentences by calibrating ordinal ranges based on the relative ranking. Finally, we are interested in the contribution of context in understanding the meaning of an unknown word.

Acknowledgments

We thank the anonymous reviewers for their suggestions, and Ann Schumacher for serving as grade level annotator. This work was supported in part by Dept. of Education grant R305A140647 to the University of Michigan. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsors.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. " O'Reilly Media, Inc.".

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *HLT-NAACL*, pages 193–200.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: a survey of current and future research. *International Journal of Applied Linguistics*, 165(2):97–135.

Edgar Dale and Jeanne S Chall. 2000. Readability revisited: The new dale-chall readability formula. `http://opi.mt.gov/Pub/RTI/Forms/School/Choteau/The\%20Dale-Chall\%20Word\%20List.doc`. Accessed: 2016-5-10.

James R. A. Davenport and Robert DeLine. 2014. The readability of tweets and their geographic correlation with education. *CoRR*, abs/1401.6058.

Orphée De Clercq, Veronique Hoste, Bart Desmet, Philip Van Oosten, Martine De Cock, and Lieve Macken. 2014. Using the crowd for readability prediction. *Natural Language Engineering*, 20(03):293–325.

Yo Ehara, Issei Sato, Hidekazu Oiwa, and Hiroshi Nakagawa. 2012. Mining words in the minds of second language learners: Learner-specific word difficulty. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 799–814.

Arthur C Graesser, Danielle S McNamara, and Jonna M Kulikowich. 2011. Coh-metrix providing multilevel analyses of text characteristics. *Educational Researcher*, 40(5):223–234.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, EANL '08, pages 71–79, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, January.

Kentaro Inui and Satomi Yamamoto. 2001. Corpus-based acquisition of sentence readability ranking models for deaf people. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, November 27-30, 2001, Hitotsubashi Memorial Hall, National Center of Sciences, Tokyo, Japan*, pages 159–166.

Tapas Kanungo and David Orr. 2009. Predicting the readability of short web summaries. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 202–211. ACM.

Paul Kidwell, Guy Lebanon, and Kevyn Collins-Thompson. 2011. Statistical estimation of word acquisition with application to readability prediction. *Journal of the American Statistical Association*, 106(493):21–30.

Adam Kilgarriff, Milos Husák, Katy McAdam, Michael Rundell, and Pavel Rychlỳ. 2008. Gdex: Automatically finding good dictionary examples in a corpus. In *Proceedings of the XIII EURALEX International Congress (Barcelona, 15-19 July 2008)*, pages 425–432.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, 44(4):978–990.

Tracy Liederholm, Michelle Gaddy Everson, Paul van den Broek, Maureen Mischinski, Alex Crittenden, and Jay Samuels. 2000. Effects of causal text revisions on more-and less-skilled readers' comprehension of easy and difficult texts. *Cognition and Instruction*, pages 525–556.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. Bllip north american news text, complete. *Linguistic Data Consortium*.

Danielle S McNamara. 2001. Reading both high-coherence and low-coherence texts: Effects of text sequence and prior knowledge. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 55(1):51.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python.

Journal of Machine Learning Research, 12:2825–2830.

Charles Perfetti and Gwen A. Frishkoff. 2008. The neural bases of text and discourse processing. In B. Stemmer and H. A. Whitaker (Eds.) Handbook of the Neuroscience of Language, pages 165–174. Cambridge:MA, Elsevier.

Ildikó Pilán, Elena Volodina, and Richard Johansson. 2014. Rule-based and machine learning approaches for second language sentence-level readability. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 186–195, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

Randi Reppen, Nancy Ide, and Keith Suderman. 2005. American national corpus (anc) second release. *Linguistic Data Consortium*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Kathleen M Sheehan, Michael Flor, and Diane Napolitano. 2013. A two-stage approach for generating unbiased estimates of text complexity. In *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*, pages 49–58.

Kathleen M Sheehan, Irene Kostin, Diane Napolitano, and Michael Flor. 2014. The textevaluator tool. *The Elementary School Journal*, 115(2):184–209.

Xingyi Song and Trevor Cohn. 2011. Regression and ranking based optimisation for sentence level machine translation evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 123–129. Association for Computational Linguistics.

A Jackson Stenner. 1996. Measuring reading comprehension with the lexile framework.

Kumiko Tanaka-Ishii, Satoshi Tezuka, and Hiroshi Terada. 2010. Sorting texts by readability. *Comput. Linguist.*, 36(2):203–227, June.

James Voss and Laurie Silfies. 1996. Learning from history text: The interaction of knowledge and comprehension skill with text structure. *Cognition and Instruction*, 14(1):45–68.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, and Paul Hsu. 2010. An overview of microsoft web n-gram corpus and applications. June.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 409–420, Stroudsburg, PA, USA. Association for Computational Linguistics.

A Neural Approach to Automated Essay Scoring

Kaveh Taghipour and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
{kaveh, nght}@comp.nus.edu.sg

Abstract

Traditional automated essay scoring systems rely on carefully designed features to evaluate and score essays. The performance of such systems is tightly bound to the quality of the underlying features. However, it is laborious to manually design the most informative features for such a system. In this paper, we develop an approach based on recurrent neural networks to learn the relation between an essay and its assigned score, without any feature engineering. We explore several neural network models for the task of automated essay scoring and perform some analysis to get some insights of the models. The results show that our best system, which is based on long short-term memory networks, outperforms a strong baseline by 5.6% in terms of quadratic weighted Kappa, without requiring any feature engineering.

1 Introduction

There is a recent surge of interest in neural networks, which are based on continuous-space representation of the input and non-linear functions. Hence, neural networks are capable of modeling complex patterns in data. Moreover, since these methods do not depend on manual engineering of features, they can be applied to solve problems in an end-to-end fashion. SENNA (Collobert et al., 2011) and neural machine translation (Bahdanau et al., 2015) are two notable examples in natural language processing that operate without any external task-specific knowledge. In this paper, we report a system based on neural networks to take advantage of their modeling capacity and generalization power for the automated essay scoring (AES) task.

Essay writing is usually a part of the student assessment process. Several organizations, such as Educational Testing Service (ETS)[1], evaluate the writing skills of students in their examinations. Because of the large number of students participating in these exams, grading all essays is very time-consuming. Thus, some organizations have been using AES systems to reduce the time and cost of scoring essays.

Automated essay scoring refers to the process of grading student essays without human interference. An AES system takes as input an essay written for a given prompt, and then assigns a numeric score to the essay reflecting its quality, based on its content, grammar, and organization. Such AES systems are usually based on regression methods applied to a set of carefully designed features. The process of feature engineering is the most difficult part of building AES systems. Moreover, it is challenging for humans to consider all the factors that are involved in assigning a score to an essay.

Our AES system, on the other hand, *learns* the features and relation between an essay and its score automatically. Since the system is based on recurrent neural networks, it can effectively encode the information required for essay evaluation and learn the complex patterns in the data through non-linear neural layers. Our system is among the first AES systems based on neural networks designed without any hand-crafted features. Our results show that our system outperforms a strong baseline and

[1] https://www.ets.org

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1882–1891,
Austin, Texas, November 1-5, 2016.

achieves state-of-the-art performance in automated essay scoring. In order to make it easier for other researchers to replicate our results, we have made the source code of our system publicly available[2].

The rest of this paper is organized as follows. Section 2 gives an overview of related work in the literature. Section 3 describes the automated essay scoring task and the evaluation metric used in this paper. We provide the details of our approach in Section 4, and present and discuss the results of our experimental evaluation in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

There exist many automated essay scoring systems (Shermis and Burstein, 2013) and some of them are being used in high-stakes assessments. e-rater (Attali and Burstein, 2004) and Intelligent Essay Assessor (Foltz et al., 1999) are two notable examples of AES systems. In 2012, a competition on automated essay scoring called 'Automated Student Assessment Prize' (ASAP)[3] was organized by Kaggle and sponsored by the Hewlett Foundation. A comprehensive comparison of AES systems was made in the ASAP competition. Although many AES systems have been developed to date, they have been built with hand-crafted features and supervised machine learning algorithms.

Researchers have devoted a substantial amount of effort to design effective features for automated essay scoring. These features can be as simple as essay length (Chen and He, 2013) or more complicated such as lexical complexity, grammaticality of a text (Attali and Burstein, 2004), or syntactic features (Chen and He, 2013). Readability features (Zesch et al., 2015) have also been proposed in the literature as another source of information. Moreover, text coherence has also been exploited to assess the flow of information and argumentation of an essay (Chen and He, 2013). A detailed overview of the features used in AES systems can be found in (Zesch et al., 2015). Moreover, some attempts have been made to address different aspects of essay writing independently. For example, argument strength and organization of essays have been tackled by some researchers through designing task-specific features for each aspect (Persing et al., 2010; Persing and Ng, 2015).

Our system, however, accepts an essay text as input directly and learns the features automatically from the data. To do so, we have developed a method based on recurrent neural networks to score the essays in an end-to-end manner. We have explored a variety of neural network models in this paper to identify the most suitable model. Our best model is a long short-term memory neural network (Hochreiter and Schmidhuber, 1997) and is trained as a regression method. Similar recurrent neural network approaches have recently been used successfully in a number of other NLP tasks. For example, Bahdanau et al. (2015) have proposed an attentive neural approach to machine translation based on gated recurrent units (Cho et al., 2014). Neural approaches have also been used for syntactic parsing. In (Vinyals et al., 2015), long short-term memory networks have been used to obtain parse trees by using a sequence-to-sequence model and formulating the parsing task as a sequence generation problem. Apart from these examples, recurrent neural networks have also been used for opinion mining (Irsoy and Cardie, 2014), sequence labeling (Ma and Hovy, 2016), language modeling (Kim et al., 2016; Sundermeyer et al., 2015), etc.

3 Automated Essay Scoring

In this section, we define the automated essay scoring task and the evaluation metric used for assessing the quality of AES systems.

3.1 Task Description

Automated essay scoring systems are used in evaluating and scoring student essays written based on a given prompt. The performance of these systems is assessed by comparing their scores assigned to a set of essays to human-assigned gold-standard scores. Since the output of AES systems is usually a real-valued number, the task is often addressed as a supervised machine learning task (mostly by regression or preference ranking). Machine learning algorithms are used to learn the relationship between the essays and reference scores.

[2]https://github.com/nusnlp/nea

[3]https://www.kaggle.com/c/asap-aes

3.2 Evaluation Metric

The output of an AES system can be compared to the ratings assigned by human annotators using various measures of correlation or agreement (Yannakoudakis and Cummins, 2015). These measures include Pearson's correlation, Spearman's correlation, Kendall's Tau, and quadratic weighted Kappa (QWK). The ASAP competition adopted QWK as the official evaluation metric. Since we use the ASAP data set for evaluation in this paper, we also use QWK as the evaluation metric in our experiments.

Quadratic weighted Kappa is calculated as follows. First, a weight matrix $\mathbf{W}$ is constructed according to Equation 1:

$$\mathbf{W}_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{1}$$

where i and j are the reference rating (assigned by a human annotator) and the hypothesis rating (assigned by an AES system), respectively, and N is the number of possible ratings. A matrix $\mathbf{O}$ is calculated such that $\mathbf{O}_{i,j}$ denotes the number of essays that receive a rating i by the human annotator and a rating j by the AES system. An expected count matrix $\mathbf{E}$ is calculated as the outer product of histogram vectors of the two (reference and hypothesis) ratings. The matrix $\mathbf{E}$ is then normalized such that the sum of elements in $\mathbf{E}$ and the sum of elements in $\mathbf{O}$ are the same. Finally, given the matrices $\mathbf{O}$ and $\mathbf{E}$, the QWK score is calculated according to Equation 2:

$$\kappa = 1 - \frac{\sum_{i,j} \mathbf{W}_{i,j}\mathbf{O}_{i,j}}{\sum_{i,j} \mathbf{W}_{i,j}\mathbf{E}_{i,j}} \tag{2}$$

In our experiments, we compare the QWK score of our system to well-established baselines. We also perform a one-tailed paired t-test to determine whether the obtained improvement is statistically significant.

4 A Recurrent Neural Network Approach

Recurrent neural networks are one of the most successful machine learning models and have attracted the attention of researchers from various fields. Compared to feed-forward neural networks, recurrent neural networks are theoretically more powerful and are capable of learning more complex patterns from data. Therefore, we have mainly focused on recurrent networks in this paper. This section gives a description of the recurrent neural network architecture that we have used for the essay scoring task and the training process.

4.1 Model Architecture

The neural network architecture that we have used in this paper is illustrated in Figure 1. We now describe each layer in our neural network in detail.

Lookup Table Layer: The first layer of our neural network projects each word into a d_{LT} dimensional space. Given a sequence of words $\mathcal{W}$ represented by their *one-hot* representations $(\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_M)$, the output of the lookup table layer is calculated by Equation 3:

$$LT(\mathcal{W}) = (\mathbf{E}.\mathbf{w}_1, \mathbf{E}.\mathbf{w}_2, \cdots, \mathbf{E}.\mathbf{w}_M) \tag{3}$$

where $\mathbf{E}$ is the word embeddings matrix and will be learnt during training.

Convolution Layer: Once the dense representation of the input sequence $\mathcal{W}$ is calculated, it is fed into the recurrent layer of the network. However, it might be beneficial for the network to extract *local features* from the sequence before applying the recurrent operation. This *optional* characteristic can be achieved by applying a convolution layer on the output of the lookup table layer. In order to extract local features from the sequence, the convolution layer applies a linear transformation to all M windows in the given sequence of vectors[4]. Given a window of dense word representations $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l$, the convolution layer first concatenates these vectors to form a vector $\bar{\mathbf{x}}$ of length $l.d_{LT}$ and then uses Equation 4 to calculate the output vector of length d_c:

$$Conv(\bar{\mathbf{x}}) = \mathbf{W}.\bar{\mathbf{x}} + \mathbf{b} \tag{4}$$

In Equation 4, $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the network and are *shared* across all windows in the sequence.

[4]The number of input vectors and the number of output vectors of the convolution layer are the same because we pad the sequence to avoid losing border windows.

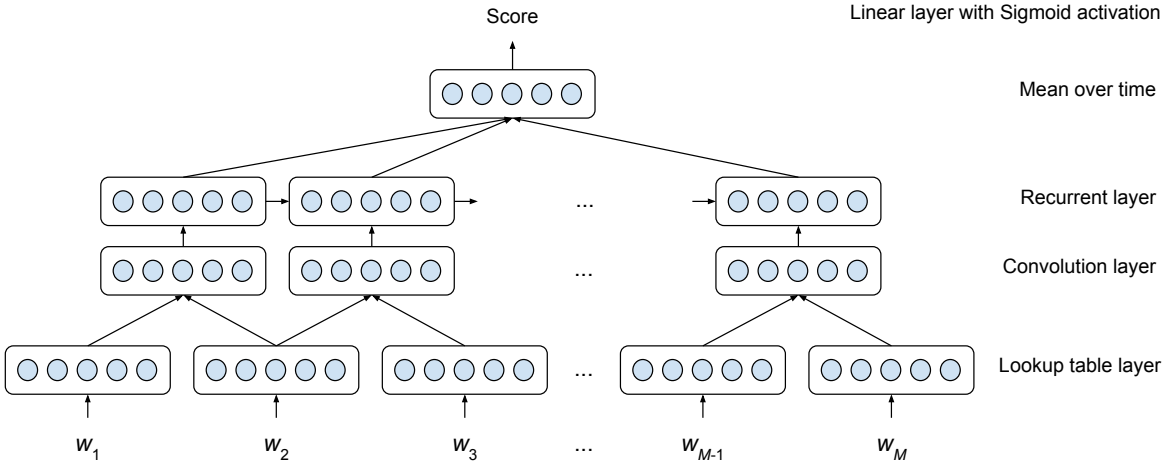


Figure 1: The convolutional recurrent neural network architecture.

The convolution layer can be seen as a function that extracts feature vectors from n-grams. Since this layer provides n-gram level information to the subsequent layers of the neural network, it can potentially capture local contextual dependencies in the essay and consequently improve the performance of the system.

Recurrent Layer: After generating embeddings (whether from the convolution layer or directly from the lookup table layer), the recurrent layer starts processing the input to generate a representation for the given essay. This representation should ideally encode all the information required for grading the essay. However, since the essays are usually long, consisting of hundreds of words, the learnt vector representation might not be sufficient for accurate scoring. For this reason, we preserve all the intermediate states of the recurrent layer to keep track of the important bits of information from processing the essay. We experimented with basic recurrent units (RNN) (Elman, 1990), gated recurrent units (GRU) (Cho et al., 2014), and long short-term memory units (LSTM) (Hochreiter and Schmidhuber, 1997) to identify the best choice for our task. Since LSTM outperforms the other two units, we only describe LSTM in this section.

Long short-term memory units are modified recurrent units that can cope with the problem of vanishing gradients more effectively (Pascanu et al., 2013). LSTMs can learn to preserve or forget the information required for the final representation. In order to control the flow of information during processing of the input sequence, LSTM units make use of three gates to discard (forget) or pass the information through time. The following equations formally describe the LSTM function:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i.\mathbf{x}_t + \mathbf{U}_i.\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f.\mathbf{x}_t + \mathbf{U}_f.\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c.\mathbf{x}_t + \mathbf{U}_c.\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{i}_t \circ \tilde{\mathbf{c}}_t + \mathbf{f}_t \circ \mathbf{c}_{t-1} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o.\mathbf{x}_t + \mathbf{U}_o.\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned} \tag{5}$$

$\mathbf{x}_t$ and $\mathbf{h}_t$ are the input and output vectors at time t, respectively. $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_c$, $\mathbf{W}_o$, $\mathbf{U}_i$, $\mathbf{U}_f$, $\mathbf{U}_c$, and $\mathbf{U}_o$ are weight matrices and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_c$, and $\mathbf{b}_o$ are bias vectors. The symbol $\circ$ denotes element-wise multiplication and σ represents the sigmoid function.

Mean over Time: The outputs of the recurrent layer, $\mathcal{H} = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_M)$, are fed into the mean-over-time layer. This layer receives M vectors of length d_r as input and calculates an average vector of the same length. This layer's function is defined in Equation 6:

$$MoT(\mathcal{H}) = \frac{1}{M}\sum_{t=1}^{M}\mathbf{h}_t \tag{6}$$

The mean-over-time layer is responsible for aggregating the variable number of inputs into a fixed length vector. Once this vector is calculated, it is fed into the linear layer to be mapped into a score.

Instead of taking the mean of the intermediate recurrent layer states $\mathbf{h}_t$, we could use the last state vector $\mathbf{h}_M$ to compute the score and remove the mean-over-time layer. However, as we will show in Section 5.2, it is much more effective to use the mean-over-time layer and take all recurrent states into account.

Linear Layer with Sigmoid Activation: The linear layer maps its input vector generated by the mean-over-time layer to a scalar value. This mapping is simply a linear transformation of the input vector and therefore, the computed value is not bounded. Since we need a bounded value in the range of valid scores for each prompt, we apply a sigmoid function to limit the possible scores to the range of $(0, 1)$. The mapping of the linear layer after applying the sigmoid activation function is given by Equation 7:

$$s(\mathbf{x}) = \text{sigmoid}(\mathbf{w}.\mathbf{x} + b) \tag{7}$$

where $\mathbf{x}$ is the input vector ($MoT(\mathcal{H})$), $\mathbf{w}$ is the weight vector, and b is the bias value.

We normalize all gold-standard scores to $[0, 1]$ and use them to train the network. However, during testing, we rescale the output of the network to the original score range and use the rescaled scores to evaluate the system.

4.2 Training

We use the RMSProp optimization algorithm (Dauphin et al., 2015) to minimize the mean squared error (MSE) loss function over the training data. Given N training essays and their corresponding normalized gold-standard scores s_i^*, the model computes the predicted scores s_i for all training essays and then updates the network parameters such that the mean squared error is minimized. The loss function is shown in Equation 8:

$$MSE(\mathbf{s}, \mathbf{s}^*) = \frac{1}{N}\sum_{i=1}^{N}(s_i - s_i^*)^2 \tag{8}$$

Additionally, we make use of dropout regularization to avoid overfitting. We also clip the gradient if the norm of the gradient is larger than a threshold.

We do not use any early stopping methods. Instead, we train the neural network model for a fixed number of epochs and monitor the performance of the model on the development set after each epoch. Once training is finished, we select the model with the best QWK score on the development set.

5 Experiments

In this section, we describe our experimental setup and present the results. Moreover, an analysis of the results and some discussion are provided in this section.

5.1 Setup

The dataset that we have used in our experiments is the same dataset used in the ASAP competition run by Kaggle (see Table 1 for some statistics). We use quadratic weighted Kappa as the evaluation metric, following the ASAP competition. Since the test set used in the competition is not publicly available, we use 5-fold cross validation to evaluate our systems. In each fold, 60% of the data is used as our training set, 20% as the development set, and 20% as the test set. We train the model for a fixed number of epochs and then choose the best model based on the development set. We tokenize the essays using the NLTK[5] tokenizer, lowercase the text, and normalize the gold-standard scores to the range of $[0, 1]$. During testing, we rescale the system-generated normalized scores to the original range of scores and measure the performance.

Prompt	#Essays	Avg length	Scores
1	1,783	350	2–12
2	1,800	350	1–6
3	1,726	150	0–3
4	1,772	150	0–3
5	1,805	150	0–4
6	1,800	150	0–4
7	1,569	250	0–30
8	723	650	0–60

Table 1: Statistics of the ASAP dataset.

In order to evaluate the performance of our system, we compare it to a publicly available open-source[6] AES system called 'Enhanced AI Scor-

[5]http://www.nltk.org

[6]https://github.com/edx/ease

ing Engine' (EASE). This system is the best open-source system that participated in the ASAP competition, and was ranked third among all 154 participating teams. EASE is based on hand-crafted features and regression methods. The features that are extracted by EASE can be categorized into four classes:

- Length-based features
- Parts-of-Speech (POS)
- Word overlap with the prompt
- Bag of n-grams

After extracting the features, a regression algorithm is used to build a model based on the training data. The details of the features and the results of using support vector regression (SVR) and Bayesian linear ridge regression (BLRR) are reported in (Phandi et al., 2015). We use these two regression methods as our baseline systems.

Our system has several hyper-parameters that need to be set. We use the RMSProp optimizer with decay rate (ρ) set to 0.9 to train the network and we set the base learning rate to 0.001. The mini-batch size is 32 in our experiments[7] and we train the network for 50 epochs. The vocabulary is the 4,000 most frequent words in the training data and all other words are mapped to a special token that represents unknown words. We regularize the network by using dropout (Srivastava et al., 2014) and we set the dropout probability to 0.5. During training, the norm of the gradient is clipped to a maximum value of 10. We set the word embedding dimension (d_{LT}) to 50 and the output dimension of the recurrent layer (d_r) to 300. If a convolution layer is used, the window size (l) is set to 3 and the output dimension of this layer (d_c) is set to 50. Finally, we initialize the lookup table layer using pre-trained word embeddings[8] released by Zou et al. (2013). Moreover, the bias value of the linear layer is initialized such that the network's output before training is almost equal to the average score in the training data.

[7]To create mini-batches for training, we pad all essays in a mini-batch using a dummy token to make them have the same length. To eliminate the effect of padding tokens during training, we mask them to prevent the network from miscalculating the gradients.

[8]http://ai.stanford.edu/∼wzou/mt

We have performed several experiments to identify the best model architecture for our task. These architectural choices are summarized below:

- Convolutional vs. recurrent neural network
- RNN unit type (basic RNN, GRU, or LSTM)
- Using mean-over-time over all recurrent states vs. using only the last recurrent state
- Using mean-over-time vs. an attention mechanism
- Using a recurrent layer vs. a convolutional recurrent layer
- Unidirectional vs. bidirectional LSTM

We have used 8 Tesla K80 GPUs to perform our experiments in parallel.

5.2 Results and Discussion

In this section, we present the results of our evaluation by comparing our system to the above-mentioned baselines (SVR and BLRR). Table 2 (rows 1 to 4) shows the QWK scores of our systems on the eight prompts from the ASAP dataset[9]. This table also contains the results of our statistical significance tests. The baseline score that we have used for hypothesis testing is underlined and the statistically significant improvements ($p < 0.05$) over the baseline are marked with '*'. It should be noted that all neural network models in Table 2 are unidirectional and include the mean-over-time layer. Except for the CNN model, convolution layer is *not* included in the networks.

According to Table 2, all model variations are able to learn the task properly and perform competitively compared to the baselines. However, LSTM performs significantly better than all other systems and outperforms the baseline by a large margin (4.1%). However, basic RNN falls behind other models and does not perform as accurately as GRU or LSTM.

[9]To aggregate the QWK scores of all prompts, Fisher transformation was used in the ASAP competition before averaging QWK scores. However, we found that applying Fisher transformation only slightly changes the scores. (If we apply this method to aggregate QWK scores, our best ensemble system (row 7, Table 2) would obtain a QWK score of 0.768.) Therefore we simply take the average of QWK scores across prompts.

ID	Systems	Prompts								
		1	2	3	4	5	6	7	8	Avg QWK
1	CNN	0.797	0.634	0.646	0.767	0.746	0.757	0.746	**0.687**	0.722
2	RNN	0.687	0.633	0.552	0.744	0.732	0.757	0.743	0.553	0.675
3	GRU	0.616	0.591	0.668	0.787	0.795	0.800	0.752	0.573	0.698
4	LSTM	0.775	0.687	0.683	0.795	**0.818**	0.813	0.805	0.594	0.746*
5	CNN (10 runs)	0.804	0.656	0.637	0.762	0.752	0.765	0.750	0.680	0.726*
6	LSTM (10 runs)	0.808	**0.697**	0.689	**0.805**	**0.818**	**0.827**	**0.811**	0.598	0.756*
7	(5) + (6)	**0.821**	0.688	**0.694**	**0.805**	0.807	0.819	0.808	0.644	**0.761***
8	EASE (SVR)	0.781	0.621	0.630	0.749	0.782	0.771	0.727	0.534	0.699
9	EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705

Table 2: The QWK scores of the various neural network models and the baselines. The baseline for the statistical significance tests is underlined and statistically significant improvements ($p < 0.05$) are marked with '*'.

This behaviour is probably because of the relatively long sequences of words in essays. GRU and LSTM have been shown to 'remember' sequences and long-term dependencies much more effectively and therefore, we believe this is the reason behind RNN's relatively poor performance.

Additionally, we perform some experiments to evaluate ensembles of our systems. We create variants of our network by training with different random initializations of the parameters. To combine these models, we simply take the average of the scores predicted by these networks. This approach is shown to improve performance by reducing the variance of the model and therefore make the predictions more accurate. Table 2 (rows 5 and 6) shows the results of CNN and LSTM ensembles over 10 runs. Moreover, we combine CNN ensembles and LSTM ensembles together to make the predictions (row 7).

As shown in Table 2, ensembles of models always lead to improvements. We obtain 0.4% and 1.0% improvement from CNN and LSTM ensembles, respectively. However, our best model (row 7 in Table 2) is the ensemble of 10 instances of CNN models and 10 instances of LSTM models and outperforms the baseline BLRR system by 5.6%.

It is possible to use the last state of the recurrent layer to predict the score instead of taking the mean over all intermediate states. In order to observe the effects of this architectural choice, we test the network with and without the mean-over-time layer. The results of this experiment are presented in Table 3, clearly showing that the neural network fails to learn the task properly in the absence of the mean-over-time layer. When the mean-over-time layer is not used in the model, the network needs to efficiently encode the whole essay into a single state vector and then use it to predict the score. However, when the mean-over-time layer is included, the model has *direct* access to all intermediate states and can recall the required intermediate information much more effectively and therefore is able to predict the score more accurately.

Systems	Avg QWK
LSTM	**0.746***
LSTM w/o MoT	0.540
LSTM+attention	0.731*
CNN+LSTM	0.708
BLSTM	0.699
EASE (SVR)	0.699
EASE (BLRR)	0.705

Table 3: The QWK scores of LSTM neural network variants. The baseline for the statistical significance tests is underlined and statistically significant improvements ($p < 0.05$) are marked with '*'.

Additionally, we experiment with three other neural network architectures. Instead of using mean-over-time to average intermediate states, we use an attention mechanism (Bahdanau et al., 2015) to compute a weighted sum of the states. In this case, we calculate the dot product of the intermediate states and a vector trained by the neural network, and then apply a softmax operation to obtain the normalized weights. Another alternative is to add a convolution layer before feeding the embeddings to the recurrent LSTM layer (CNN+LSTM) and evaluate the model. We also use a bidirectional LSTM model (BLSTM), in which the sequence of words

is processed in both directions and the intermediate states generated by both LSTM layers are merged and then fed into the mean-over-time layer. The results of testing these architectures are summarized in Table 3.

The attention mechanism significantly improves the results compared to LSTM without mean-over-time, but it does not perform as well as LSTM with mean-over-time. The other two architectural choices do not lead to further improvements over the LSTM neural network. This observation is in line with the findings of some other researchers (Kadlec et al., 2015) and is probably because of the relatively small number of training examples compared to the capacity of the models.

We have also compared the accuracy of our best system (shown as 'AES') with human performance, presented in Table 4. To do so, we calculate the agreement (QWK scores) between our system and each of the two human annotators separately ('AES - H1' and 'AES - H2'), as well as the agreement between the two human annotators ('H1 - H2'). According to Table 4, the performance of our system on average is very close to human annotators. In fact, for some of the prompts, the agreement between our system and the human annotators is even higher than the agreement between human annotators. In general, we can conclude that our method is just below the upper limit and approaching human-level performance.

We also compare our system to a recently published automated essay scoring method based on neural networks (Alikaniotis et al., 2016). Instead of performing cross validation, Alikaniotis et al. (2016) partition the ASAP dataset into two parts by using 80% of the data for training and the remaining 20% for testing. For comparison, we also carry out an experiment on the same training and test data used in (Alikaniotis et al., 2016). Following how QWK scores are computed in Alikaniotis et al. (2016), instead of calculating QWK for each prompt separately and averaging them, we calculate the QWK score for the whole test set, by setting the minimum score to 0 and the maximum score to 60. Using this evaluation setup, our LSTM system achieves a QWK score of 0.987, higher than the QWK score of 0.96 of the best system in (Alikaniotis et al., 2016). In this way of calculating QWK scores, since

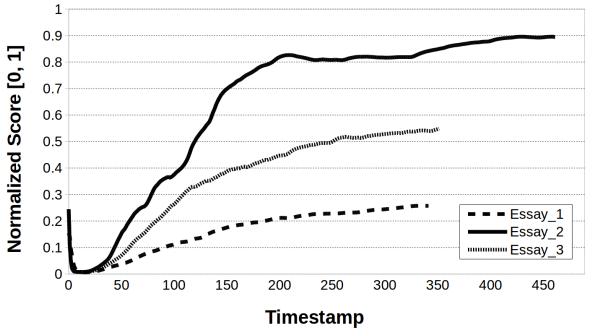


Figure 2: Score variations per timestamp. All scores are normalized to the range of $[0, 1]$.

the majority of the test essays have a much smaller score range (see Table 1) compared to $[0, 60]$, the differences between the system-predicted scores and the gold-standard scores will be small most of the time. For example, more than 55% of the essays in the test set have a score range of $[0, 3]$ or $[0, 4]$ and therefore, for these prompts, the differences between human-assigned gold-standard scores and the scores predicted by an AES system will be small in the range of $[0, 60]$. For this reason, in contrast to prompt-specific QWK calculation, the QWK scores are much higher in this evaluation setting and far exceed the QWK score for human agreement when computed in a prompt-specific way (see Table 4).

Interpreting neural network models and the interactions between nodes is not an easy task. However, it is possible to gain an insight of a network by analyzing the behavior of particular nodes. In order to understand how our neural network assigns the scores, we monitor the score variations while testing the model. Figure 2 displays the score variations for three essays after processing each word (at each timestamp) by the neural network. We have selected a poorly written essay, a well written essay, and an average essay with *normalized* gold-standard scores of 0.2, 0.8, and 0.6, respectively.

According to Figure 2, the network learns to take *essay length* into account and assigns a very low score to all short essays with fewer than 50 words, regardless of the content. This pattern recurs for all essays and is not specific to the three selected essays in Figure 2. However, if an essay is long enough, the content becomes more important and the AES system starts discriminating well written

Description	Prompts								
	1	2	3	4	5	6	7	8	Avg QWK
AES - H1	0.750	0.684	0.662	0.759	0.751	0.791	0.731	0.607	0.717
AES - H2	0.767	0.690	0.632	0.762	0.769	0.775	0.752	0.530	0.710
H1 - H2	0.721	0.812	0.769	0.851	0.753	0.776	0.720	0.627	0.754

Table 4: Comparison with human performance. H1 and H2 denote human rater 1 and human rater 2, respectively, and AES refers to our best system (ensemble of CNN and LSTM models).

essays from poorly written ones. As shown in Figure 2, the model properly assigns a higher score to the well written essay 2, while giving lower scores to the other essays. This observation confirms that the model successfully learns the required features for automated essay scoring. While it is difficult to associate different parts of the neural network model with specific features, it is clear that appropriate indicators of essay quality are being learnt, including essay length and essay content.

6 Conclusion

In this paper, we have proposed an approach based on recurrent neural networks to tackle the task of automated essay scoring. Our method does not rely on any feature engineering and automatically learns the representations required for the task. We have explored a variety of neural network model architectures for automated essay scoring and have achieved significant improvements over a strong open-source baseline. Our best system outperforms the baseline by 5.6% in terms of quadratic weighted Kappa. Furthermore, an analysis of the network has been performed to get an insight of the recurrent neural network model and we show that the method effectively utilizes essay content to extract the required information for scoring essays.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2013-T2-1-150. We are also grateful to the anonymous reviewers for their helpful comments.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. Technical report, Educational Testing Service.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

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

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems 28*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. The Intelligent Essay Assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*.

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

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for Ubuntu corpus dialogs. In *Proceesings of the NIPS 2015*

Workshop on Machine Learning for Spoken Language Understanding and Interaction.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mark D. Shermis and Jill Burstein, editors. 2013. *Handbook of Automated Essay Evaluation: Current Applications and New Directions*. Routledge.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*.

Helen Yannakoudakis and Ronan Cummins. 2015. Evaluating the performance of automated text scoring systems. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Non-uniform Language Detection in Technical Writing

Weibo Wang[1], **Abidalrahman Moh'd**[1], **Aminul Islam**[2]
Axel J. Soto[3], and **Evangelos E. Milios**[1]

[1]Faculty of Computer Science, Dalhousie University, Canada
{weibo,amohd,eem}@cs.dal.ca
[2]School of Computing and Informatics, University of Louisiana at Lafayette, USA
aminul@louisiana.edu
[3]School of Computer Science, University of Manchester, UK
axel.soto@manchester.ac.uk

Abstract

Technical writing in professional environments, such as user manual authoring, requires the use of uniform language. Non-uniform language detection is a novel task, which aims to guarantee the consistency for technical writing by detecting sentences in a document that are intended to have the same meaning within a similar context but use different words or writing style. This paper proposes an approach that utilizes text similarity algorithms at lexical, syntactic, semantic and pragmatic levels. Different features are extracted and integrated by applying a machine learning classification method. We tested our method using smart phone user manuals, and compared its performance against the state-of-the-art methods in a related area. The experiments demonstrate that our approach achieves the upper bound performance for this task.

1 Introduction

Technical writing, such as creating device operation manuals and user guide handbooks, is a special writing task that requires accurate text to describe a certain product or operation. To avoid ambiguity and bring accurate and straightforward understanding to readers, technical writing requires consistency in the use of terminology and uniform language (Farkas, 1985). There are always demands from modern industries to improve the quality of technical documents in cost-efficient ways.

Non-uniform Language Detection (NLD) aims to avoid inner-inconsistency and ambiguity of technical content by identifying non-uniform sentences. Such sentences are intended to have the same meaning or usage within a similar context but use different words or writing style. However, even though non-uniform sentences tend to have similar wording, similar sentence pairs do not necessarily indicate a non-uniform language instance. For example, here are four similar sentence pairs cited from the iPhone user manual (Apple Inc., 2015), where only two pairs are true non-uniform language instances:

(1) tap the screen to **show** the controls.
tap the screen to **display** the controls.

(2) tap the screen to **show** the controls.
tap the screen to **display** the controls.

(3) if the **photo** hasn't been downloaded yet, tap the download notice first.
if the **video** hasn't been downloaded yet, tap the download notice first.

(4) you can also turn blue tooth on or off in control center.
you can also turn **wi-fi and** blue tooth on or off in control center.

As we can see above, the pattern of difference within each sentence pair could be between one word and one word, or one word and multiple words, or one sentence having extra words or phrases that the other sentence does not have. Each pattern could be a true or false non-uniform language instance depending on the content and context. The word '**show**' and '**display**' are synonyms in Example (1). Both sentences convey the same meaning, so they are an instance of non-uniform language. In Example (2), even though '**enter**' and '**write**' are not synonyms, since the two sentences describe the same

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1892–1900,
Austin, Texas, November 1-5, 2016.

operation, they should be considered as non-uniform language as well. In Example (3), even though the only different words between the sentences, '**photo**' and '**video**', are both media contents, because they are different objects, they should not be regarded as non-uniform language. In Example (4), it is a false candidate because each sentence mentions different functions. However, the two sentences are unequal in length, thus it is hard to know what the extra phrase '**wi-fi and**' should be compared against. Therefore, it is challenging to distinguish true and false occurrences of non-uniform cases based on text similarity algorithms only, and finer grained analyses need to be applied. To address the problem of NLD, this paper proposes a methodology for detecting non-uniform language within a technical document at the sentence level. A schematic diagram of our approach is shown in Figure 1.

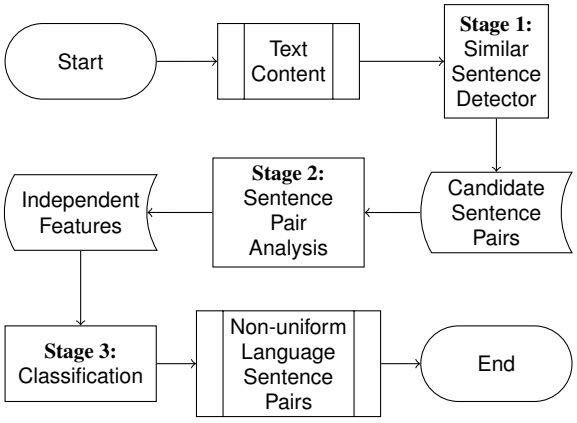


Figure 1: Schematic diagram of our approach

It is worth to mention that NLD is similar to Plagiarism Detection and Paraphrase Detection (PD) as all these tasks aim to capture similar sentences with the same meaning (Das and Smith, 2009). However, the goal of authors in plagiarism and paraphrasing is to change as many words as possible to increase the differences between texts, whereas in technical writing, the authors try to avoid such differences, but they do not always succeed and thus NLD solutions are needed. Cases of plagiarism and paraphrasing with high lexical differences will be typically classified as NLD negative, and cases with low lexical differences will be typically classified as NLD positive. While true positive cases for both NLD and PD can exist, there are not likely to happen in practice since textual differences in PD tend to be much higher than in NLD.

To address the NLD task, Natural Language Processing (NLP) techniques at lexical, syntactic, semantic, and pragmatic levels are utilized. Our approach also integrates resources such as Part-of-Speech (POS) tagger (Bird et al., 2009), WordNet (Miller et al., 1990), Google Tri-gram Method (GTM) (Islam et al., 2012; Mei et al., 2015), and Flickr[1]. Analyses from different perspectives are applied, and the results are regarded as independent features that are finally integrated by applying a classification method based on Support Vector Machine (SVM). A ground truth dataset created from three smart phone user manuals is used for evaluation, and it is made publicly available[2]. The experiments on this dataset demonstrate that the proposed solution to the NLD task is the most efficient method to date, and the final result is close to the upper bound performance.

2 Related Work

NLD is closely related to PD, which aims to detect sentences that have essentially the same meaning. However, paraphrase is a restatement using different words to make it appear different from the original text. PD techniques cannot perform well on the NLD task as they focus on variations at a coarser granularity. We reviewed studies in the PD area, and found the Recursive Auto-Encoder (RAE) (Socher et al., 2011), and the Semantic Text Similarity (STS) (Islam and Inkpen, 2008) to be the state-of-the-art methods using supervised and unsupervised-based PD, respectively. However, all the four examples provided in the introduction section would be recognized as paraphrases by these analyzers, even though only two of the pairs are real non-uniform language cases. Thus, state-of-the-art PD techniques are unable to make accurate judgments on these instances since PD do not address the necessary level of detail for the NLD task.

Another related area to NLD is near-duplicate text detection. It focuses on short text such as mobile phone short messages, or tweets, which are intended to have the same meaning but differ in terms of in-

[1]Flickr: https://www.flickr.com/

[2]The resource is available at: https://goo.gl/6wRchr

formal abbreviations, transliterations, and network languages (Gong et al., 2008). The detection and elimination of near-duplicate text is of great importance for other text language processing such as clustering, opinion mining, and topic detection (Sun et al., 2013). However, the studies in this area focus on reducing the comparison time in large scale text databases and creating informal abbreviation corpus, rather than exploring the text similarity methods. Basic similarity methods, such as Longest Common Substring (LCS) are utilized, but they are not sufficient to address the NLD task as LCS captures the matching words and their order between texts and using LCS alone will give high recall and low precision for the NLD task. For the following NLD negative example, LCS returns a high similarity score:

(5) If the **photo** hasn't been downloaded yet, tap the download notice first.
If the **music** hasn't been downloaded yet, tap the download notice first.

Examples of this type are common in technical writing, so other features are needed besides LCS to recognize NLD positives.

There is a research domain named near-duplicate document detection, which seems literally related to NLD, but also represents a different task. It focuses on documents that are identical in terms of written content but differ in a small portion of the document such as advertisements, counters and timestamps (Manku et al., 2007). Such documents are important to be identified for web crawling and the automatic collection of digital libraries. Since this area focuses on the variations between two documents, especially the variations on metadata, rather than the written content within one document, their proposed solutions are not a good fit for the NLD tasks.

3 Non-uniform Language Detection

As we have shown in Figure 1, a framework consisting of three stages is proposed to address the NLD task. The first stage extracts candidate sentence pairs that have high text similarity within a document. The second stage performs comprehensive analyses on each candidate sentence pair. The analyses are performed at lexical, syntactical, semantic, and pragmatic levels, where multiple NLP resources such as POS tagger, WordNet, GTM, and Flickr are utilized. The final stage integrates all the analysis results by applying a classification method based on SVM to classify the candidate sentence pairs as true or false cases of non-uniform language.

3.1 Stage 1: Similar Sentences Detection

To extract the candidate sentence pairs, three text similarity algorithms are combined and applied at the sentence level. GTM is an unsupervised corpus-based approach for measuring semantic relatedness between texts. LCS focuses on the word order of sentences. Cosine Similarity provides bag-of-word similarity. GTM, LCS, and Cosine Similarity are used to filter out the pairs based on semantics, sentence structure, and word frequency, respectively.

The filtering thresholds were set by running experiments at the sentence level on the iPhone user manual (Apple Inc., 2015). Algorithm 1 is used to set the filtering threshold for each average sentence length[3].

We utilize a sentence detector and a tokenizer[4] to divide the text of the manual into a sentence set of n sentence pairs (Line 2). We separately run Algorithm 1 three times to set the threshold sets for GTM, LCS, and Cosine. The thresholds are set based on the lengths of both sentences of a sentence pair. The average length starts from 2 and is increased by one once the threshold for the current length is set. We discovered that once the sentence length goes above 10, the thresholds vary little. Therefore, we stop the algorithm when the threshold for pairs of average length equal to 10 is found (Line 6).

For each different average length, the algorithm starts by asking the user to input an initial similarity threshold and an increasing step value (Line 4-5). An initial threshold range is generated based on the user setting. The lower bound of the range is T and the upper bound of the range is T+*Step* (Line 9-10). Then the algorithm would loop over all the sentence pairs (Line 11-20) and add the pairs within the current threshold range into set C (Line 14-16).

[3]See the Example (4) in Section 1, where two sentences within one sentence pair could be unequal in length, thus we compute the average length to represent the length of each candidate pair.

[4]OpenNLP: `https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html`

Input : User Manual
Output: Threshold-Length_List $[(T_1, L_1), ...]$

1 **begin**
2 $S[n] \longleftarrow$ *SentenceDetector*(User Manual)
3 $L \longleftarrow 2$ /*Initial average length of a sentence pair*/
4 $T \longleftarrow$ Similarity threshold
5 $Step \longleftarrow$ Threshold increasing step
6 **while** $(L \leq 10)$ **do**
7 $C \longleftarrow \emptyset$ /*Initialize the output sentence container.*/
8 **do**
9 $T_{low} \longleftarrow T$
10 $T_{up} \longleftarrow T + Step$
11 **for** *(i=0; i<n; i++)* **do**
12 **for** *(j=0; j<n; j++)* **do**
13 $AvgL \longleftarrow (S[i] + S[j])/2$
14 **if** $AvgL \in [L-1, L)$ **then**
15 **if** $(T_{low} \leq Sim(S[i], S[j]))$ **and** $(Sim(S[i], S[j]) \leq T_{up})$ **then**
16 $C \overset{add}{\longleftarrow} (S[i], S[j])$
17 **end**
18 **end**
19 **end**
20 **end**
21 $T \longleftarrow T+Step$
22 **while** *(Check(C)=True)* /*Checked by human, True when all the sentence pairs are not instances of non-uniform language.*/;
24 *Threshold-Length_List* $\overset{add}{\longleftarrow} (T_{low}, L)$
25 $L++;$
26 **end**
27 **end**

Algorithm 1: Setting similarity thresholds

The similarity of sentence pairs above the previous threshold and below the current threshold are captured and analyzed (Line 15-16). If they consist of all false non-uniform language candidates, we repeat the loop with a higher threshold to filter more false candidates. Once we discover that a true candidate is filtered by the current thresholds, we stop increasing and set the prior value as the threshold to maximize the recall ratio. The whole experiment is repeated for different sentence pair lengths. The final thresholds for different similarity methods are shown in Figure 2.

To filter the sentence pairs, we applied the thresholds of the three text similarity algorithms. For example, assume there are two sentences of nine-word length on average. The similarity scores of this pair have to be above all the GTM, LCS and Cosine thresholds (which are 0.943, 0.836, and 0.932, according to Figure 2) to make it a candidate instance.

By applying the thresholds shown in Figure 2, candidate pairs could be detected in reasonable scale in terms of the size of the corpus, and achieve good

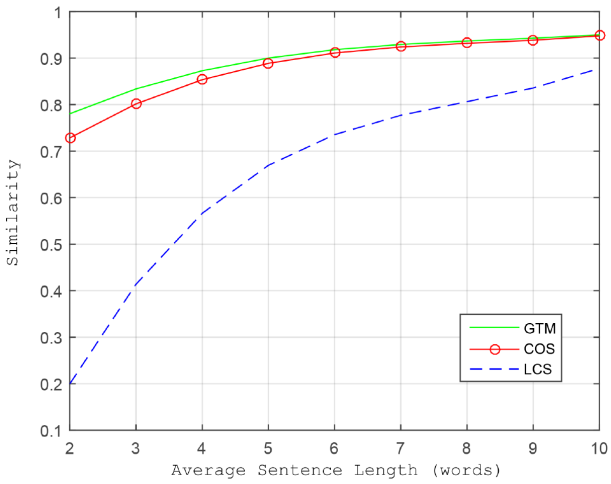


Figure 2: Candidate filtering thresholds

recall ratio as well. As for precision, around 40% of the candidates are true non-uniform language cases, where the remaining candidates are supposed to be filtered in the second stage.

3.2 Stage 2: Sentence Pair Analysis

In this stage, we aim to determine for the two sentences of a candidate pair whether they describe the same object or operation using different words or writing style (i.e., true non-uniform language) or they just appear similar but actually have different intended meanings, by using the following features.

3.2.1 Part-of-Speech Tagging Analysis

POS tags are added for each candidate pair using NLTK (Bird et al., 2009) tagger to gain a grammatical view over the sentences.

As Table 1 shows, some differences in sentence content can be captured using POS tags, but some cannot. Thus, it is necessary to make further syntactic and semantic analysis to distinguish true candidates from false ones.

We categorized the different POS tags into the following groups shown in Table 2. The different POS tags are mapped to different categories, which are then used as one more feature of the sentence pair representation.

3.2.2 Character N-gram Analysis

In the character N-gram analysis, the relatedness between the different words of each candidate pair is calculated in terms of character unigram, bigram

Candidate Sentence Pair with POS Tag	Ground Truth
Link/NNP your/PRP device/NN to/TO iTunes/NNS **stores/NNS** Link/NNP your/PRP device/NN to/TO iTunes/NNS **store/NN**	True Candidate
go/VB to/TO settings/NNS >/SYS general/JJ >/SYS accessibility/NN >/SYS **audio/NN** go/VB to/TO settings/NNS >/SYS general/JJ >/SYS accessibility/NN >/SYS **video/NN**	False Candidate
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shutdown/NN** the/DT device/NN Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shut/VBN down/RB** the/DT device/NN	True Candidate
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **turn/VBN off/IN** the/DT device/NN Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shut/VBN down/RB** the/DT device/NN	True Candidate

Table 1: POS analysis on candidate sentence pairs

Label	Description	Example
1	Equal length, same POS tag	**/NN** vs. **/NN**, **/VB** vs. **/VB**
2	Equal length, plural noun with singular noun	**/NN** vs. **/NNS**
3	Equal length, different POS	**/NN** vs. **/VB**
4	Unequal length, extra article	**/NN** vs. **/DT/NN**
5	Unequal length, extra conjunction	**/NN** vs. **/CC/NN**
6	Unequal length, extra adjective	**/NN** vs. **/JJ/NN**
7	Other POS tag types.	**/NN** vs. **N/A**

Table 2: POS tag categorizing

and trigram similarity. The character N-gram frequencies with a window size from 1 to 3 is firstly calculated. Then, the N-gram distance based on the frequencies is calculated using the Common N-gram distance (CNG) (Kešelj and Cercone, 2004):

$$d(f_1, f_2) = \sum_{n \in dom(f_1) \cup dom(f_2)} \left(\frac{f_1(n) - f_2(n)}{\frac{f_1(n)+f_2(n)}{2}}\right)^2 \quad (1)$$

where $dom(f_i)$ is the domain of function f_i. In the equation above, n represents a certain N-gram unit. $f_i(n)$ represents the frequency of n in sentence i *(i=1,2)*. If n does not appear in sentence i, $f_i(n)$=0. The lower bound of the N-gram distance is 0 (when the two units to be compared are exactly the same). The higher the value of N-gram distance, the larger the difference, thus there is no upper bound. CNG was demonstrated to be a robust measure of dissimilarity for character N-grams in different domains (Wołkowicz and Kešelj, 2013).

3.2.3 WordNet Lexical Relation Analysis

For a given candidate sentence pair, if the different wordingF are synonymous to each other, there is a high likelihood that the two sentences try to convey the same meaning but using different expressions. On the other hand, if the different parts of a candidate pair are not related at the lexical level, then it is reasonable to assume that this pair is describing different objects/actions and thus they might not be instances of non-uniform language.

WordNet is utilized here to analyze the lexical relationship within each candidate pair to determine whether they are synonyms to each other. To perform this analysis, we only used synset information from WordNet, and we only considered words as synonyms if they belong to a same synset. The rationale is that a similar sentence pair tends to be an instance of non-uniform language if the different words are synonyms, rather than having other

relationships such as hypernymy, hyponymy, and antonymy. Therefore, we do not deem necessary to include these relationships into our analysis. For example, given a similar sentence pair:

(6) if the **photo** hasn't been downloaded yet, tap the download notice first.
if the **video** hasn't been downloaded yet, tap the download notice first.

The sentence pair above is not a non-uniform language instance. However, the relatedness score between '**photo**' and '**video**' given by Wu-Palmer metric (Wu and Palmer, 1994) using WordNet is 0.6, which is fairly high compared to a random word pair. Yet we do not know how these words are related, e.g., "photo is a kind of video", "photo is a part of video", or "photo and video are examples of media content". Thus, we might make wrong judgments based on such a similarity score. However, using synset information, we know that these words are not synonyms and thus probably not suggesting a non-uniform language instance. Therefore, we considered as one more feature of our classifier whether mismatching words belong to the same synset.

3.2.4 GTM Word Relatedness Analysis

Besides text similarity, GTM also measures semantic relatedness between words. To find the relatedness between a pair of words, GTM takes into account all the trigrams that start and end with the given pair of words and then normalizes their mean frequency using unigram frequency of each of the words as well as the most frequent unigram in the Google Web 1T N-gram corpus (Brants and Franz, 2006), and extends the word relatedness method to measure document relatedness.

3.2.5 Flickr Related Concept Analysis

In some cases, word to word relatedness exists that goes beyond dictionary definitions, such as metonymy, in which a thing or concept is called not by its own name but rather by the name of something associated in meaning with that thing or concept (Kövecses and Radden, 1998). Metonymy detection is actually a task at the pragmatic level of NLP area, which can be appied for NLD in technical writing.

Flickr is a popular photo sharing website that supports time and location metadata and user tagging for each photo. Since the tags are added by humans and aim to describe or comment on a certain photo, the tags are somehow related from a human perspective. As a result, Flickr becomes a large online resource with the potential to find metonymy relationships in text.

Flickr made available statistical information about their dataset that can be used to query related concepts of a certain word or phrase online. We utilized this resource to detect whether the different parts within a candidate sentence pair are related at the pragmatic level. A boolean value that indicates metonymy relationship is obtained and regarded as another feature of our sentence pair representation for our NLD analysis. Table 3 gives some examples of relatedness that could be discovered in this stage.

Different Content	Is Metonymy
aeroplane, A380	True
film, hollywood	True
apple, iPhone	True
audio, grayscale	False

Table 3: Example of analysis using Flickr

3.3 Stage 3: SVM Classification

All the metrics described above are regarded as features of our candidate sentence pairs. To make a comprehensive judgment based on these different signals, a classification method based on SVM (Vladimir and Vapnik, 1995) is applied. We implemented the SVM classification using "e1071" package[5] in R.

Using our labeled corpus, we trained an SVM model on 61.5% of the data and used the remaining for testing.

4 Experiments and Evaluation

In this section, we present the dataset, experimental work and results, including results using other baseline methods for comparative purposes.

4.1 Experiment Data

We downloaded smart phone user manuals of iPhone (Apple Inc., 2015), LG (LG, 2009) and Samsung (Samsung, 2011), which are available online

[5]https://cran.r-project.org/web/packages/e1071/

as three raw datasets. Then, we performed Stage 1 three times on the three different datasets, and identified 325 candidate sentence pairs (650 sentences) as part of Stage 1, which is considered as our candidate dataset. Before applying the sentence analysis and classification stages, each candidate sentence pair in the dataset was labeled by three different annotators as true or false. Then the ground truth for each instance is generated by annotators' voting. The annotators worked separately to label the sentence pairs. Cases of disagreement were sent again to the annotators to double-check their judgement. Some statistics from the manuals are shown in Table 4.

Data Source	Data Volume (Pages)	Candidate Pairs (True, False)
iPhone	196	208 (102, 106)
LG	274	54 (16, 38)
Samsung	190	63 (32, 31)

Table 4: Experiment data distribution

To prepare for the SVM based classification stage, we split the dataset into a training set DS_{train}, and a testing set DS_{test}. Considering that the data distribution is nearly balanced in terms of true and false instances, DS_{train} was formed by randomly selecting 200 instances from the dataset and the remaining 125 instances were used for DS_{test}.

4.2 Evaluation Methods and Results

The performance of each annotator against the majority voting is evaluated in terms of Precision, Recall, Accuracy, and F-measure. These results along with the number of true/ false, positive/ negative cases for each annotator are presented in Table 5.

Parameters	Expert 1	Expert 2	Expert 3
True-positive	130	99	125
True-negative	161	164	166
False-positive	20	51	25
False-negative	14	11	9
Precision	**86.67**	66.00	83.33
Recall	90.27	90.00	**93.28**
Accuracy	**89.54**	80.92	**89.54**
F-Measure	**88.43**	76.15	88.03

Table 5: Evaluation of annotators performance

To measure the agreement among annotators, the Fleiss' Kappa test (Fleiss and Cohen, 1973) is used. Fleiss' Kappa is an extension of Cohen's Kappa (Cohen, 1968). Unlike Cohen's Kappa, which only measures the agreement between two annotators, Fleiss' Kappa measures the agreement among three or more annotators. In our case, we have 3 annotators (the annotator number n is 3), each annotator labeled 325 candidate pairs (the subject volume N is 325), each candidate pair is labeled either 0 or 1 (the value of category k is 2). The final Fleiss' Kappa Value is 0.545, which indicates a moderate agreement level (0.41-0.60) based on the Kappa Interpretation Model (Fleiss and Cohen, 1973). In other words, the performance of the annotators reveal that the NLD task is not simple, since there are many cases that are ambiguous and hard to make accurate judgments on, even for humans.

As Table 5 shows, the best performance of annotators is highlighted and regarded as the upper bound performance (UB) of the NLD task on our dataset. The state-of-the-art unsupervised PD system named STS (Islam and Inkpen, 2008), as well as the state-of-the-art supervised PD system named RAE (Socher et al., 2011), are utilized to generate the baselines of the NLD task. STS uses the similarity score of 0.5 as the threshold to evaluate their method in the PD task. RAE applies supervised learning to classify a pair as a true or false instance of paraphrasing. These approaches are utilized on our evaluation as baselines for the NLD task.

After defining the upper bound and baseline performances, we evaluated our proposed method, which we name as Non-uniform Language Detecting System (NLDS), by training the SVM classifier on DS_{train}, and then performing classification using the SVM classifier on DS_{test}. The result is shown in Table 6 as the NLDS method. The first row presents the upper bound performance and the following two rows present the baseline performances.

To assess the importance of each feature utilized in the proposed framework, we performed a feature ablation study (Cohen and Howe, 1988) on N-gram, POS analysis, lexical analysis (GTM and WordNet), and Flickr, separately on the DS_{test} dataset. The results are listed in Table 6.

A series of cross-validation and Student's t-tests are applied after running NLDS, STS, RAE, and UB

Method	R(%)	P (%)	A(%)	F1(%)
UB	92.38	86.67	**89.54**	**88.43**
STS	**100**	46.15	46.15	63.16
RAE	**100**	46.40	46.40	63.39
Uni-gram	11.11	35.29	52.80	16.90
Bi-gram	44.44	61.54	64.00	51.61
Tri-gram	50.00	62.79	65.60	55.67
POS	77.78	72.77	78.40	76.52
Lexical	85.18	59.74	68.80	70.23
Flickr	48.96	94.00	74.00	64.38
NLDS	80.95	**96.22**	88.80	87.93

Table 6: Evaluation of NLDS

methods on the F-measure metric. The tests reveal that the performance of NLDS is significantly better than STS and RAE, no significant differences could be found between UB and NLDS. These results demonstrate that NLDS would represent an effective approach for NLD that is on pair with annotator judgement and overcomes state-of-the-art approaches for related tasks.

4.3 Discussion

As Table 6 shows, the PD systems STS and RAE regard all the test cases as true non-uniform language cases, so the recall ratio is 1 but the precision is low.

It is worth noting that by using character N-gram analysis alone, it is not possible to obtain good results. This is because the character N-gram analysis using a probabilistic method is unable to capture any difference or relatedness in the meaning, while the NLD task relies heavily on discovering such relatedness. The reason we applied the N-gram analysis is to use it as a supplementary method to catch differences such as between '**shut down**' (two words) and '**shutdown**' (one word), or some spelling errors.

POS analysis provides a syntactic perspective for the text instances. For instances, '**then(/RB)**' versus '**and(/CC)**', and '**store(/NN)**' versus '**stores(/NNS)**', the differences can be reflected in POS tags. Yet, POS analysis alone could not capture the difference between words such as '**writing(/VBG)**' versus '**entering(/VBG)**' since they share the same POS tag. These features make POS analysis outperform the character N-gram analysis, but not semantic-based approaches.

Lexical analysis (GTM and WordNet) achieves the best recall ratio since it can provide semantic relatedness, which is the most important aspect for the NLD task. Flickr is utilized as a supplementary resource to provide pragmatic relatedness.

By combining the different types of analyses above, the differences of each sentence pair are analyzed at different NLP levels and thus, the relatedness and difference from structural, grammatical, syntactic, semantic and pragmatic perspectives can be captured and integrated by the classification method.

5 Conclusions

This paper proposes NLDS to detect non-uniform language for technical writings at sentence level. Text, stream-based, and word similarity algorithms at the lexical, syntactic, semantic, and pragmatic levels are integrated through an SVM-based classification method. To evaluate the proposed method, three annotators manually labeled all the candidate instances identified in Stage 1. Then we assigned the ground truth for each instance pair by annotators' voting. Fleiss' Kappa test is applied to reflect the difference of human judgments and thus to reveal the difficulty of this task.

We also evaluated each annotator against the ground truth, and defined the best performance of human as the upper bound performance for this task. With the generated ground truth, a series of experiments using our implemented system were carried out with different smart phone user manuals data. We evaluated the results by comparing the outcome of the classifier with the results using each single feature, as well as the state-of-the-art PD methods.

Considering the different annotators' judgments as reflected by Fleiss' Kappa Value, the NLD task is fairly difficult. Yet, the performance of our system is close to human performance. The experiments reveal that our solution is the most effective method to date and the performance is close to the upper bound that we defined. As for future work, we would apply deeper analysis on true non-uniform language pairs to indicate which sentence of the pair fits better with the style and language of the rest of the document. We would then provide a semi-automatic correction function to facilitate authors with the task of removing non-uniform language occurrences.

Acknowledgments

This research work was supported by Innovatia Inc. and NSERC. We are thankful to our colleagues Andrew Albert, David Crowley, and Erika Allen who proposed and defined this NLD task, and provided expertise that contributed on the preparation of this paper.

References

Apple Inc. 2015. iPhone User Guide For iOS 8.4 Software. https://manuals.info.apple.com/MANUALS/1000/MA1565/en_US/iphone_user_guide.pdf.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. " O'Reilly Media, Inc.".

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1.

Paul R Cohen and Adele E Howe. 1988. How evaluation guides AI research: The message still counts more than the medium. *AI magazine*, 9(4):35.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

David K Farkas. 1985. The concept of consistency in writing and editing. *Journal of Technical Writing and Communication*, 15(4):353–364.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*.

Caichun Gong, Yulan Huang, Xueqi Cheng, and Shuo Bai. 2008. Detecting near-duplicates in large-scale short text databases. In *Advances in Knowledge Discovery and Data Mining*, pages 877–883. Springer.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

Aminul Islam, Evangelos Milios, and Vlado Kešelj. 2012. Text similarity using google tri-grams. In *Advances in Artificial Intelligence*, pages 312–317. Springer.

Vlado Kešelj and Nick Cercone. 2004. CNG method with weighted voting. In *Ad-hoc Authorship Attribution Competition. Proceedings 2004 Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities (ALLC/ACH 2004), Göteborg, Sweden*.

Zoltán Kövecses and Günter Radden. 1998. Metonymy: Developing a cognitive linguistic view. *Cognitive Linguistics (includes Cognitive Linguistic Bibliography)*, 9(1):37–78.

LG. 2009. LG600G User Guide. https://www.tracfone.com/images/en/phones/TFLG600G/manual.pdf.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150. ACM.

Jie Mei, Xinxin Kou, Zhimin Yao, Andrew Rau-Chaplin, Aminul Islam, Abidalrahman Moh'd, and Evangelos E. Milios. 2015. Efficient computation of co-occurrence based word relatedness. DemoURL: http://ares.research.cs.dal.ca/gtm/.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244.

Samsung. 2011. Samsung 010505d5 cell phone user manual. http://cellphone.manualsonline.com/manuals/mfg/samsung/010505d5.html?p=53.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

Yifang Sun, Jianbin Qin, and Wei Wang. 2013. Near duplicate text detection using frequency-biased signatures. In *Web Information Systems Engineering–WISE 2013*, pages 277–291. Springer.

Vapnik N Vladimir and V Vapnik. 1995. The nature of statistical learning theory.

Jacek Wołkowicz and Vlado Kešelj. 2013. Evaluation of n-gram-based classification approaches on classical music corpora. In *Mathematics and computation in music*, pages 213–225. Springer.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. DemoURL:http://ws4jdemo.appspot.com/?mode=w&s1=&w1=photo&s2=&w2=video.

Adapting Grammatical Error Correction Based on the Native Language of Writers with Neural Network Joint Models

Shamil Chollampatt[1] and **Duc Tam Hoang**[2] and **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science
National University of Singapore
shamil@u.nus.edu, {hoangdt,nght}@comp.nus.edu.sg

Abstract

An important aspect for the task of grammatical error correction (GEC) that has not yet been adequately explored is adaptation based on the native language (L1) of writers, despite the marked influences of L1 on second language (L2) writing. In this paper, we adapt a neural network joint model (NNJM) using L1-specific learner text and integrate it into a statistical machine translation (SMT) based GEC system. Specifically, we train an NNJM on general learner text (not L1-specific) and subsequently train on L1-specific data using a Kullback-Leibler divergence regularized objective function in order to preserve generalization of the model. We incorporate this adapted NNJM as a feature in an SMT-based English GEC system and show that adaptation achieves significant $F_{0.5}$ score gains on English texts written by L1 Chinese, Russian, and Spanish writers.

1 Introduction

Grammatical error correction (GEC) deals with the automatic correction of errors (spelling, grammar, and collocation errors), particularly in non-native written text. The native language (L1) background of the writer has a noticeable influence on the errors made in second language (L2) writing, and considering this factor can potentially improve the performance of GEC systems. For example, consider the following sentence written by a Finnish writer (Jarvis and Odlin, 2000): *"When they had escaped **in** the police car they sat under the tree.*" The preposition *in* appears to be grammatically correct. However, in the given context, the preposition '*from*' is the correct choice in place of the preposition '*in*'. Finnish learners of English tend to overgeneralize the use of the preposition '*in*'. Knowledge of L1 makes the correction more probable whenever the preposition *in* appears in texts written by Finnish writers. Similarly, Chinese learners of English tend to make frequent verb tense and verb form errors, since Chinese lacks verb inflection (Shaughnessy, 1977). The cross-linguistic influence of L1 on L2 writing is a highly complex phenomenon, and the errors made by learners cannot be directly attributed to the similarities or differences between the two languages. As Ortega (2009) points out, learners seem to operate on two complementary principles: "what works in L1 may work in L2 because human languages are fundamentally alike; but if it sounds too L1-like, it will probably not work in L2". In this paper, we follow a data-driven approach to model these influences and adapt GEC systems using L2 texts written by writers of the same L1 background.

The two most popular approaches for grammatical error correction are the classification approach (Dahlmeier et al., 2012; Rozovskaya et al., 2014) and the statistical machine translation (SMT) approach (Chollampatt et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2014). The SMT approach has emerged as a popular paradigm for GEC because of its ability to learn text transformations from ill-formed to well-formed text enabling it to correct a wide variety of errors including complex errors that are difficult to handle for the classification approach (Rozovskaya and Roth, 2016). The phrase-based SMT approach has been used in state-of-the-art GEC systems (Rozovskaya and Roth, 2016;

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1901–1911,
Austin, Texas, November 1-5, 2016.

Chollampatt et al., 2016; Hoang et al., 2016). The SMT approach does not model error types specifically, nor does it require linguistic analysis like parsing and part-of-speech (POS) tagging. We adopt a phrase-based SMT approach to GEC in this paper. Additionally, we implement and incorporate a neural network joint model (NNJM) (Devlin et al., 2014) as a feature in our SMT-based GEC system. It is easy to integrate an NNJM into the SMT decoding framework as it uses a fixed-window context and it has shown to improve SMT-based GEC (Chollampatt et al., 2016). We adapt the NNJM to L1-specific data (i.e., English text written by writers of a particular L1) and obtain significant improvements over the baseline which uses an unadapted NNJM. Adaptation is done by using the unadapted NNJM trained on general domain data (i.e., not L1-specific) using a log likelihood objective function with self-normalization (Devlin et al., 2014) as the starting point, and training for subsequent iterations using the smaller L1-specific in-domain data with a modified objective function which includes a Kullback-Leibler (KL) divergence regularization term. This modified objective function prevents overfitting on the smaller in-domain data and preserves the generalization capability of the NNJM. We show that this method of adaptation works on very small and high-quality L1-specific data as well (50–100 essays).

In summary, the two major contributions of this paper are as follows. (1) This is the first work that performs L1-based adaptation for GEC using the SMT approach and covering all error types. (2) We introduce a *novel* method of NNJM adaptation and demonstrate that this method can work with in-domain data that are much smaller than the general domain data.

2 Related Work

In the past decade, there has been increasing attention on GEC in English, mainly due to the growing number of English as second language (ESL) learners around the world. The popularity of this problem grew further through Helping Our Own (HOO) (Dale and Kilgarriff, 2011; Dale et al., 2012) and CoNLL shared tasks (Ng et al., 2013; Ng et al., 2014). The majority of the published work on GEC aimed at building classifiers or rule-based systems for specific error types and combined them to build hybrid systems (Dahlmeier et al., 2012; Rozovskaya et al., 2014).

The cross-linguistic influences between L1 and L2 have been mainly used for the task of native language identification (Massung and Zhai, 2016). It has also been used in typology prediction (Berzak et al., 2014) and predicting error distributions in ESL data (Berzak et al., 2015). L1-based adaptation has previously shown to improve GEC for specific error types using the classification approach. Rozovskaya and Roth (2010) used an approach to correct preposition errors by restricting the candidate corrections to those observed in L1-specific data. They further added artificial training data that mimic the error frequency in L1-specific text to improve accuracy. In their later work, Rozovskaya and Roth (2011) focused on L1-based adaptation for preposition and article correction, by modifying the prior probabilities in the naïve Bayes classifier during decision time based on L1-specific ESL learner text. Both approaches use native data for training, but rely on non-native L1-specific text to introduce artificial errors or to modify the prior probabilities. Dahlmeier and Ng (2011) implemented a system to correct collocation errors, by adding paraphrases derived from L1 into the confusion set. Specifically, they use a bilingual L1-L2 corpus, to obtain L2 paraphrases, which are likely to be translated to the same phrase in L1. There is no prior work on L1-based adaptation for GEC using the machine translation approach, which is a one of the most popular approaches for GEC.

With the availability of large-scale error corrected data (Mizumoto et al., 2011), the statistical machine translation (SMT) approach to GEC became popular and was employed in state-of-the-art GEC systems. Comparison of the classification approach and the machine translation approach can be found in (Rozovskaya and Roth, 2016) and (Susanto et al., 2014). Recently, an end-to-end neural machine translation framework was proposed for GEC (Yuan and Briscoe, 2016), which was shown to achieve competitive results. Neural network joint models have shown to be improve SMT-based GEC systems (Chollampatt et al., 2016) due to their ability to model words and phrases in a continuous space, access to larger contexts from source side, and abil-

ity to learn non-linear mappings from input to output. In this paper, we exploit the advantages of the SMT approach and neural network joint models (NNJMs) by adapting an NNJM based on the L1 background of the writers and integrating it into the SMT framework. We perform KL divergence regularized adaptation to prevent overfitting on the smaller in-domain data. KL divergence regularization was previously used by Yu et al. (2013) for speaker adaptation. Joty et al. (2015) proposed another NNJM adaptation method, which uses a regularized objective function that encourages a network trained on general-domain data to be closer to an in-domain NNJM. Other adaptation techniques used in SMT include mixture modeling (Foster and Kuhn, 2007; Moore and Lewis, 2010; Sennrich, 2012) and alternative decoding paths (Koehn and Schroeder, 2007).

3 A Machine Translation Framework for Grammatical Error Correction

We formulate GEC as a translation task from a possibly erroneous input sentence to a corrected sentence. We use the popular phrase-based SMT system, Moses (Koehn et al., 2007), which employs a log linear model to find the best correction hypothesis T^* given an input sentence S:

$$T^* = \underset{T}{\operatorname{argmax}}\, P(T|S) = \underset{T}{\operatorname{argmax}} \sum_{i=1}^{N} \mu_i f_i(T, S)$$

where μ_i and $f_i(T, S)$ are the i^{th} feature weight and feature function, respectively. We use the standard features in Moses, without any re-ordering models. The two main components of an SMT system are the translation model (TM) and the language model (LM). The TM (typically, a phrase table), responsible for generating hypotheses, is trained using parallel data, i.e., learner-written sentences (source data) and their corresponding corrected sentences (target data). It also scores the hypotheses using features like forward and inverse phrase translation probabilities and lexical weights. The LM is trained on well-formed text and ensures the fluency of the corrected output. The feature weights μ_i are computed by minimum error rate training (MERT), optimizing the $F_{0.5}$ measure (Junczys-Dowmunt and Grundkiewicz, 2014) using a development set. The $F_{0.5}$ measure computed using the MaxMatch scorer (Dahlmeier and Ng, 2012) is the standard evaluation metric for GEC used in the CoNLL-2014 shared task (Ng et al., 2014), weighting precision twice as much as recall.

Apart from the TM and the n-gram LM, we add a neural network joint model (NNJM) (Devlin et al., 2014) as a feature, following Chollampatt et al. (2016), who reported that NNJM improves the performance of a state-of-the-art SMT-based GEC system. Unlike Recurrent Neural Networks (RNNs) and Long Short Term Memory networks (LSTMs), NNJMs have a feed-forward architecture which relies on a fixed context. This makes it easy to integrate NNJMs into a machine translation decoder as a feature. The feature value is given by $\log P(T|S)$, which is the sum of the log probabilities of individual target words in the hypothesis T given the context:

$$\log P(T|S) \approx \sum_{i=1}^{|T|} \log P(t_i|h_i) \tag{1}$$

where $|T|$ is the number of words in the target hypothesis (corrected sentence), t_i is the i^{th} target word, and h_i is the context of t_i. The context h_i consists of $n-1$ previous target words and m source words surrounding the source word that is aligned to the target word t_i.

Each dimension in the output layer of the neural network (Chollampatt et al., 2016) gives the probability of a word t in the output vocabulary given its context h:

$$P(y = t|h) = \frac{\exp(U_t(h))}{Z(h)} = \frac{\exp(U_t(h))}{\sum_{t' \in V_o} \exp(U_{t'}(h))}$$

where $U_t(h)$ is the unnormalized output score before the softmax, and V_o is the output vocabulary.

The neural network parameters which include the weights, biases, and embedding matrices are trained using back propagation with stochastic gradient descent (LeCun et al., 1998). Instead of using the noise contrastive estimation (NCE) loss as done in (Chollampatt et al., 2016), we use the log likelihood objective function with a self-normalization term similar to Devlin et al. (2014):

$$L = \frac{1}{N} \sum_{i=1}^{N} \left[\log p(y = t_i|h_i) - \alpha \log^2(Z(h_i))\right] \tag{2}$$

where N is the number of training instances, and t_i is the target word in the i^{th} training instance. Each training instance consists of a target word t and its context h. α is the self-normalization coefficient which we set to 0.1, following Devlin et al. (2014). The training can be done efficiently on GPUs. We adapt this NNJM using L1-specific learner text using a Kullback-Leibler divergence regularized objective function as described in Section 4.

4 KL Divergence Regularized Adaptation

We first train an NNJM with the general-domain data (the erroneous and corrected texts, not considering the L1 of the writers) as described in the previous section. Let $p^{GD}(y|h)$ be the probability distribution estimated by the general-domain NNJM. Starting from this NNJM, subsequent iterations of training are done using the L1-specific in-domain data alone. The in-domain data consists of the erroneous texts written by writers of a specific L1 and their corresponding corrected texts. This adaptive training is done using a modified objective function having a regularization term K, which is used to minimize the KL divergence between $p^{GD}(y|h)$ and the network's output probability distribution $p(y|h)$ (Yu et al., 2013):

$$K = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{V_o} p^{GD}(y = t_j|h_i)\log p(y = t_j|h_i)$$

The term K will prevent the estimated probability distribution from deviating too much from that of the general domain NNJM during training. Our final objective function for the adaptation step is a linear combination of the terms in L and K, with a regularization weight λ and a self-normalization coefficient α:

$$\begin{aligned} L' =& \lambda K + (1-\lambda)\frac{1}{N}\sum_{i=1}^{N}\log p(y = t_i|h_i) \\ & - \alpha\frac{1}{N}\sum_{i=1}^{N}\log^2(Z(h_i)) \end{aligned}$$

We integrate the unadapted NNJM and adapted NNJM independently into our SMT-based GEC system in order to compare the effect of adaptation.

5 Other Adaptation Methods

We compare our method against two other adaptation methods previously used in SMT.

Translation Model Interpolation: Following Sennrich (2012), we linearly interpolate the features in two phrase tables, one trained on in-domain data (L1-specific learner text) and the other on out-of-domain data. The interpolation weights are set by minimization of perplexity using phrase pairs extracted from our in-domain development set. The lexical weights are re-computed from the lexical counts and the interpolation weights are re-normalized if a phrase pair exists only in one of the phrase tables.

Neural Domain Adaptation Model: Joty et al. (2015) proposed an adaptation of NNJM for SMT. They first train an NNJM using in-domain data, and then perform regularized adaptation on the general domain data (concatenation of in-domain and out-of-domain data) which restricts the model from drifting away from the in-domain NNJM. Specifically, they add a regularization term J to the objective function in their adaptive training step:

$$J = \frac{1}{N}\sum_{i=1}^{N} p^{ID}(y = t_i|h_i)\log p(y = t_i|h_i)$$

where $p^{ID}(y|h)$ id probability distribution estimated by the in-domain NNJM.

NDAM has the following drawbacks compared to our method: (1) Regularization is done using probabilities of the target words alone and not on the entire probability distribution over all words, leading to a weak regularization. (2) If the in-domain data is too small, the probability distribution learnt by the in-domain NNJM will be overfitted. Therefore, encouraging adaptation to be closer to this probability distribution may not yield a good model. Our method, on the other hand, can utilize in-domain data of very small sizes to fine tune a general NNJM. (3) Their method requires retraining of the model on complete training data in order to adapt to each domain. On the contrary, our method can adapt a single general model to different domains using small in-domain data, leading to a considerable reduction in training time.

We re-implement their method by incorporating this regularization term into the log likelihood objec-

tive function with self-normalization, L (Equation 2), during adaptive training.

6 Data and Evaluation

The training data consist of two corpora: the NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) and the Lang-8 Learner Corpora v2 (Mizumoto et al., 2011). We extract texts written by learners who learn only English from Lang-8. A language identification tool `langid.py`[1] (Lui and Baldwin, 2011) is then used to obtain purely English sentences. In addition, we remove noisy source-target sentence pairs in Lang8 where the ratio of the lengths of the source and target sentences is outside $[0.5, 2.0]$, or their word overlap ratio is less than 0.2. A sentence pair where the source or target sentence has more than 80 words is also removed from both NUCLE and Lang-8. The statistics of the data after pre-processing are shown in Table 1.

Corpus	#sents	#src tokens	#tgt tokens
NUCLE	57,063	1,156,460	1,151,278
LANG-8	2,048,654	24,649,768	25,912,707
CONCAT	2,105,717	25,806,228	27,063,985

Table 1: Statistics of training data

We obtain L1-specific in-domain data for adaptation based on the L1 information provided in Lang-8. Adaptation is performed on English texts written by learners of three different L1 backgrounds: Chinese, Russian, and Spanish. The statistics of the in-domain data from Lang-8 for each L1 are given in Table 2. For each L1, its out-of-domain data are obtained by excluding the L1-specific in-domain data (from Table 2) from the combined training data (CONCAT).

L1	#sents	#src tokens	#tgt tokens
Chinese	260,872	3,521,336	3,688,098
Russian	43,488	566,517	596,692
Spanish	19,357	292,257	309,236

Table 2: Statistics of L1-specific data in Lang-8

We use the publicly available CLC-FCE (Yannakoudakis et al., 2011) corpus to obtain the development and test data. The FCE corpus contains 1,244 scripts written by 1,244 distinct candidates sitting the Cambridge ESOL First Certificate in English (FCE) examination in 2000 and 2001. The corpus identifies the L1 of each writer. We extract the scripts written by Chinese, Russian, and Spanish native writers. We split the data for each L1 into two roughly equal parts based on the number of scripts, of which one part is used as the development data and other part is used as the test data. Splitting based on the number of scripts ensures that there is no overlap between the writers of the development and test data, as each script is written by a unique learner. The details of the FCE dataset corresponding to each L1 are given in Table 3.

	#scripts	#sents	#src tokens	#tgt tokens	#errors
L1: *Chinese*					
DEV	33	1,041	15,424	15,601	1,751
TEST	33	1,078	15,640	15,816	1,487
L1: *Russian*					
DEV	41	1,125	17,021	17,267	1,782
TEST	42	1,263	18,738	18,920	1,934
L1: *Spanish*					
DEV	100	2,281	41,375	41,681	4,133
TEST	100	2,431	41,557	42,035	4,237

Table 3: Statistics of the FCE dataset for each L1

For evaluation, we use the $F_{0.5}$ measure, computed by the M^2scorer v3.2 (Dahlmeier and Ng, 2012), as our evaluation metric. The error annotations in FCE are converted to the format required by the M^2scorer. The statistics of error annotations after converting to this format are given in Table 3. To deal with the instability of parameter tuning in SMT, we perform five runs of tuning and calculate the statistical significance by stratified approximate randomization test, as recommended by (Clark et al., 2011).

7 Experiments and Results

7.1 Baseline SMT-based GEC system

We use Moses (Version 3) to build all our SMT-based GEC systems. The phrase table of the baseline system (S_{CONCAT}) is trained using the complete

[1] https://github.com/saffsd/langid.py

	L1: *Chinese*			L1: *Russian*			L1: *Spanish*		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
S_{IN}	50.03	16.11	35.09	38.11	16.99	30.52	43.40	12.74	29.28
S_{OUT}	49.88	17.34	36.23	54.78	21.15	41.54	57.18	16.10	37.83
S_{CONCAT}	51.72	17.56	37.23	54.17	21.70	41.62	55.45	16.93	38.09
S_{CONCAT} + $NNJM_{BASELINE}$	50.47	18.75	37.63	55.22	21.73	42.15	58.30	16.42	38.60
NNJM adaptation using KL divergence regularization									
S_{CONCAT} + $NNJM_{ADAPTED}$	56.02	17.59	38.90	55.71	22.53	43.03	59.05	16.77	39.24
S_{CONCAT} + $NNJM_{ADAPTED\ (FCE)}$	53.82	18.60	**38.91**	56.03	22.46	**43.13**	58.88	16.95	**39.38**
Comparison to other adaptation techniques									
TM_{INT} + $NNJM_{BASELINE}$	55.70	17.18	38.38	54.97	21.90	42.21	58.32	16.44	38.59
S_{CONCAT} + NDAM	56.56	16.76	38.31	54.60	22.03	42.11	58.28	16.64	38.83
TM_{INT} + $NNJM_{ADAPTED}$	55.89	17.62	38.81	56.30	21.75	42.70	57.04	16.97	38.73
Using smaller general domain data									
S_{CONCAT} + $NNJM_{SMALL\text{-}BASELINE}$	53.29	17.47	37.75	55.34	20.87	41.55	58.05	16.46	38.55
S_{CONCAT} + $NDAM_{SMALL}$	53.89	17.36	37.87	55.29	21.09	41.70	56.82	16.69	38.36
S_{CONCAT} + $NNJM_{SMALL\text{-}ADAPTED}$	52.41	17.40	37.37	56.03	21.17	42.09	58.34	16.79	39.01

Table 4: Precision (P), recall (R), and $F_{0.5}$ of L1-based adaptation of GEC systems. All results are averaged over 5 runs of tuning and evaluation.

training data. We use two 5-gram language models (LMs) trained using KenLM (Heafield et al., 2013). One LM is trained on the English Wikipedia (about 1.78 billion tokens) and another on the target side of the complete training data. The system is tuned using MERT, optimizing the $F_{0.5}$ measure on the L1-specific development data in Table 2.

For comparison, we show two other baselines S_{IN} and S_{OUT}, where the phrase tables for each L1 are trained on the in-domain data only (Table 2) and the out-of-domain data only, respectively. The results of the above baseline GEC systems on L1 Chinese, Russian, and Spanish FCE test data are summarized in Table 4. We enhance the baseline S_{CONCAT} with an NNJM feature, as described in following subsection.

7.2 NNJM Adaptation

We implement NNJM in Python using the deep learning library Theano[2] (Bergstra et al., 2010) in order to use the massively parallel processing power of GPUs for training. We first train an NNJM ($NNJM_{BASELINE}$) with complete training data for 10 epochs. The source context window size is set to 5 and the target context window size is set to 4, making it a (5+5)-gram joint model. Training is done using stochastic gradient descent with a mini-batch size of 128 and learning rate of 0.1. To speed up training and decoding, a single hidden layer neural network is used with an input embedding dimension of 192 and 512 hidden units. We use a self-normalization coefficient of 0.1. We pick 16,000 and 32,000 most frequent words on the source and target sides as our source context vocabulary and target context vocabulary, respectively. The output vocabulary is set to be the same as the target vocabulary. The vocabulary is selected from the complete training data, and not based on the L1-specific in-domain data. We add the self-normalized NNJM as a feature to our baseline GEC system, S_{CONCAT} to build a stronger baseline. This is referred to as S_{CONCAT} + $NNJM_{BASELINE}$ in Table 4.

We perform adaptation on $NNJM_{BASELINE}$ by training for 10 additional epochs using the in-domain training data alone. We use the same hyper-parameters, network structure, and vocabulary, but with the KL-divergence regularized objective function (regularization weight $\lambda = 0.5$). We train the adapted NNJM ($NNJM_{ADAPTED}$) specific to each L1. We integrate these to our baseline GEC system, and the adapted systems are referred to as S_{CONCAT} + $NNJM_{ADAPTED}$ in Table 4. The results are averaged over five runs of tuning and evaluation. Our evaluation shows that each adapted system S_{CONCAT}

[2]http://deeplearning.net/software/theano

+ $\text{NNJM}_{\text{ADAPTED}}$ outperforms every baseline system (S_{IN}, S_{OUT}, S_{CONCAT}, and S_{CONCAT} + $\text{NNJM}_{\text{BASELINE}}$) significantly on all three L1s ($p < 0.01$).

7.3 Comparison to Other Adaptation Techniques

We compare our method to two different adaptation techniques described in Section 5: Translation Model Interpolation (TM_{INT}) (Sennrich, 2012) and Neural Domain Adaptation Model (NDAM) (Joty et al., 2015)[3]. The optimization of interpolation weights for TM_{INT} is done using the L1-specific FCE development data. NDAM is trained on the complete training data (CONCAT) for 10 epochs by regularizing using an in-domain NNJM also trained for 10 epochs on L1-specific in-domain data from Lang-8. For NDAM, we use the same vocabulary and hyperparameters as our NNJMs.

The results are shown in the rows TM_{INT} + $\text{NNJM}_{\text{BASELINE}}$ and S_{CONCAT} + NDAM in Table 4. Our evaluation shows that for L1 Russian and L1 Spanish, our adapted system S_{CONCAT} + $\text{NNJM}_{\text{ADAPTED}}$ significantly outperforms both TM_{INT} + $\text{NNJM}_{\text{BASELINE}}$ and S_{CONCAT} + NDAM ($p < 0.01$), but the improvement is not statistically significant for L1 Chinese.

Our evaluation also shows that the combination of TM_{INT} and adapted NNJM is similar (for L1 Chinese and Russian) or worse (for Spanish) in performance compared to S_{CONCAT} + $\text{NNJM}_{\text{ADAPTED}}$. This is because $\text{NNJM}_{\text{ADAPTED}}$ by itself is a translation model adaptation (because it considers source and target side contexts) and hence using TM_{INT} along with it does not bring in any newer information and may even hurt the performance when the in-domain data is very small (in the case of Spanish).

7.4 Effect of Adaptation Data

We also perform adaptation on the L1-specific FCE development set in Table 3 (which is also our development data for the GEC systems), instead of the in-domain data from Lang-8 in Table 2. Our neural network overfits easily on the FCE development set due to its much smaller size. Hence, we perform adaptive training for only 2 epochs on top of $\text{NNJM}_{\text{BASELINE}}$. The results are shown in the row S_{CONCAT} + $\text{NNJM}_{\text{ADAPTED (FCE)}}$ in Table 4. Although the FCE development data is much smaller in size than the L1-specific in-domain data from Lang-8, we observe similar improvements on all three L1s. This is likely due to the similarity of the development and test sets, which are obtained from the same FCE corpus. These experiments show that smaller high-quality L1-specific error annotated data (1,000–2,000 sentences) similar to the target data can be used for adaptation to give competitive results compared to using much larger in-domain data (20,000–250,000 sentences) from other sources.

We perform experiments with smaller general domain data. In order to do this, we select a subset of CONCAT composed of the in-domain data of the three L1s along with 300,000 sentences randomly selected from the rest of CONCAT. This corpus is referred to as SMALL-CONCAT (623,717 sentences and 7,990,659 source tokens). We perform both KL-divergence regularized NNJM adaptation ($\text{NNJM}_{\text{SMALL-ADAPTED}}$) as well as Neural Domain Adaptation Model (Joty et al., 2015) ($\text{NDAM}_{\text{SMALL}}$) and compare them to NNJM trained with SMALL-CONCAT ($\text{NNJM}_{\text{SMALL-BASELINE}}$). We use these NNJMs with our S_{CONCAT} baseline. The results are summarized in Table 4. When the ratio between in-domain data and general domain data is high, both adaptation methods do not significantly improve over an unadapted NNJM. In the case of L1 Spanish, KL-divergence regularized adaptation significantly outperforms the unadapted NNJM and NDAM as the size of in-domain data is smaller.

7.5 Effect of Regularization

To analyze the effect of regularization when smaller data are used, we vary the regularization weight λ in our objective function (Section 4). The results are shown in Figure 1. $\lambda = 0$ corresponds to no regularization and training completely depends on the in-domain data apart from using the general NNJM as a starting point. On the other hand, setting $\lambda = 1$ forces the distribution learnt by the network to be similar to that of the unadapted model. We see that having no regularization ($\lambda = 0$) fails on all three L1s, due to overfitting on the smaller in-domain data. However, the effect of varying regularization is more significant on L1 Russian and Spanish, as the general NNJM has seen much smaller in-domain

[3] We use the NDAM_{vl} (Joty et al., 2015) trained using the log likelihood objective function with self-normalization.

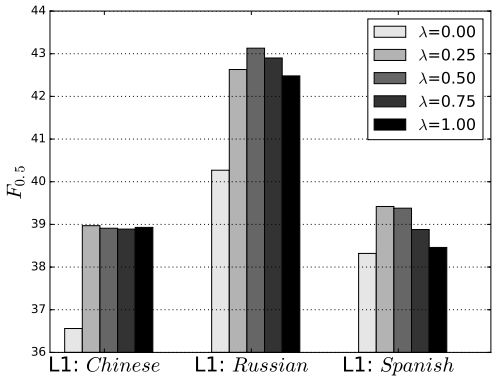


Figure 1: Effect of regularization for S_{CONCAT} + $NNJM_{ADAPTED\ (FCE)}$

data compared to L1 Chinese.

7.6 Evaluation on Benchmark Dataset

We also evaluate our system on the benchmark CoNLL-2014 shared task (Ng et al., 2014) test set for GEC in English. The CoNLL-2014 shared task consists of 1,312 sentences with two annotators. We also perform evaluation on the extension of CoNLL-2014 test set (Bryant and Ng, 2015), which contains eight additional sets of annotations over the two sets of annotations provided in the original test set. Following the settings of the CoNLL-2014 shared task, we tune our unadapted baseline system and the L1-adapted systems on the CoNLL-2013 shared task test set consisting of 1,381 test sentences. The results are summarized in Table 5.

We find that only the systems adapted based on L1 Chinese improves over the unadapted baseline system (S_{CONCAT} + $NNJM_{BASELINE}$). When the smaller-sized, high-quality FCE data is used for adaptation the margin of improvement is higher. This could be due to large proportion of Chinese learner written text in CoNLL-2014 test set, as the essays are written by the students of National University of Singapore comprising mostly of native Chinese speakers. Adaptation to L1 Russian and Spanish, does not help the system on CoNLL-2014 test set. We also compare our baseline SMT-based system with other state-of-the-art GEC systems. Our baseline system which is SMT-based, achieves the best $F_{0.5}$ score compared to other systems using the SMT approach alone, making it a competitive SMT-based GEC baseline. Overall, (Rozovskaya and Roth, 2016)

System	CoNLL-2014	
	ST	**10ANN**
S_{CONCAT} + $NNJM_{BASELINE}$	42.80	59.14
Adaptation based on L1 Chinese		
S_{CONCAT} + $NNJM_{ADAPTED}$	43.06	59.27
S_{CONCAT} + $NNJM_{ADAPTED\ (FCE)}$	44.27	60.36
Adaptation based on L1 Russian		
S_{CONCAT} + $NNJM_{ADAPTED}$	42.73	58.90
S_{CONCAT} + $NNJM_{ADAPTED\ (FCE)}$	42.12	58.30
Adaptation based on L1 Spanish		
S_{CONCAT} + $NNJM_{ADAPTED}$	41.88	58.32
S_{CONCAT} + $NNJM_{ADAPTED\ (FCE)}$	42.36	58.54
Best Published Results		
Rozovskaya and Roth (2016)		
(classifiers + spelling + SMT)	47.40	-
(SMT)	39.48	-
Chollampatt et al. (2016) (SMT)	41.75	57.19
Shared Task Teams		
CAMB (classifiers, rules, SMT)	37.33	54.26
CUUI (classifiers)	36.79	51.79
AMU (SMT)	35.01	50.17

Table 5: ST denotes $F_{0.5}$ scores on the shared task test set and **10ANN** denotes the $F_{0.5}$ scores on the extended test set with 10 sets of annotations.

achieves the best $F_{0.5}$ score (47.40) after adding classifier components, spelling checker, punctuation and capitalization correction components in a pipeline with their SMT-based system. However, their SMT-based system alone achieves an $F_{0.5}$ score of 39.48 only.

8 Discussion and Error Analysis

Our results show that L1-based adaptation of the NNJM using L1-specific in-domain data from Lang-8 significantly improves the $F_{0.5}$ score of the GEC system on the three L1s by 1.27 (Chinese), 0.88 (Russian), and 0.64 (Spanish). We observe similar gains when smaller in-domain development data from FCE is used for adaptation. These results show that adaptation based on L1 is beneficial for targeted error correction based on the native language of the writers. Our results also show that the proposed method of NNJM adaptation is scalable to different sizes of in-domain and general domain data and outperforms other methods of adaptation like phrase table interpolation (Sennrich, 2012) and Neural Domain Adaptation Model (NDAM) (Joty et al., 2015).

We perform error analysis on four error types

Error type	Δ $F_{0.5}$		
	Chinese	*Russian*	*Spanish*
verb form/tense	+0.394	+0.298	-0.124
determiner	**+2.892**	**+2.440**	+1.648
preposition	+0.084	+2.010	**+1.806**
noun number	+0.130	-0.706	+0.822
all	+0.400	+1.068	+0.586

Table 6: Differences between per error type $F_{0.5}$ scores of *system* and *baseline* for the three L1s

which are difficult for non-native learners of English.

We compare the outputs produced by our adapted *system*: S_{CONCAT} + $NNJM_{ADAPTED}$ and the *baseline*: S_{CONCAT} + $NNJM_{BASELINE}$. We perform per error type quantitative analysis of our results by observing the difference in the per error type $F_{0.5}$ scores averaged over five runs of tuning and evaluation of *baseline* and *system*. Computing per error type $F_{0.5}$ scores is difficult for SMT-based GEC systems, as the error types for corrections proposed by the SMT-based GEC system cannot be determined trivially. To overcome this difficulty, we attempt to determine the error type of the proposed corrections by matching them to the available human annotations (the source/target phrase without the surrounding context) in the complete FCE dataset (1,244 scripts). We select those sentences from the test data where the error type of every correction proposed by the *baseline* and the *system* can be determined. This process selects 928, 1102, and 2179 sentences for L1 Chinese, Russian, and Spanish, respectively. The differences in per error type $F_{0.5}$ scores between *system* and *baseline* are shown in Table 6. For Chinese, the largest gain in $F_{0.5}$ is observed for determiner errors. Determiner errors are frequent in our L1 Chinese FCE test set (10.02%) . Moreover, we see that adaptation improves verb form errors by approximately 0.4% $F_{0.5}$. Verb form errors are the most frequent error type in our L1 Chinese test set (14.46%). Also, the highest gain for L1 Russian comes from determiner errors which is the most frequent error type in our FCE test data for L1 Russian (13.55%). Similarly, the highest gain for L1 Spanish comes from preposition errors which is the most frequent error type for L1 Spanish (12.51%).

From a practical standpoint, the adapted system can be used as an educational aid in English classes of local students in non-English-speaking countries, where all the students share the same L1 and their L1 is known in advance. The adapted system can give focused feedback to learners by correcting mistakes frequently made by learners having the same L1. Also, NNJM adaptation proposed in this paper can be done using a small number of essays (50–100 essays) in a relatively short time (20–30 minutes), making it easy to adapt GEC systems in practice.

9 Conclusion

In this paper, we perform NNJM adaptation using L1-specific learner text with a KL divergence regularized objective function. We integrate adaptation into an SMT-based GEC system. The systems with adapted NNJMs outperform unadapted baselines significantly. We also demonstrate the necessity for regularization when adapting on smaller in-domain data. Our method of adaptation performs better compared to other adaptation methods, especially when smaller in-domain data is used. Our results show that adapting GEC systems for learners of similar L1 background gives significant improvement and can be adopted in practice to improve GEC systems.

Acknowledgments

We thank Kaveh Taghipour for insightful comments and discussions throughout this work. We are also grateful to the anonymous reviewers for their feedback which helped in revising and improving the paper. This research is supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2013-T2-1-150.

References

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2014. Reconstructing native language typology from foreign language usage. In *Proceedings of the 19th Conference on Computational Natural Language Learning*.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2015. Contrastive analysis with predictive power: Typology

driven estimation of grammatical error distributions in ESL. In *Proceedings of the 19th Conference on Computational Natural Language Learning*.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with L1-induced paraphrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. 2012. NUS at the HOO 2012 shared task. In *Proceedings of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an SMT approach to grammatical error correction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Scott Jarvis and Terence Odlin. 2000. Morphological type, spatial reference, and language transfer. *Studies in Second Language Acquisition*, 22:535–556.

Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali, and Stephan Vogel. 2015. How to avoid unwanted pregnancies: Domain adaptation using neural network models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (Interactive Poster and Demonstration Sessions)*.

Yann LeCun, Leon Bottou, Genevieve Orr, and Klaus Müller. 1998. Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 9–50.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*.

Sean Massung and Chengxiang Zhai. 2016. Non-native text analysis: A survey. *Natural Language Engineering*, 22:163–186.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of

language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Lourdes Ortega. 2009. *Understanding Second Language Acquisition*. Hodder Education.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Mina Shaughnessy. 1977. *Errors and Expectations*. New York: Oxford University Press.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Orthographic Syllable as basic unit for SMT between Related Languages

Anoop Kunchukuttan, Pushpak Bhattacharyya
Center For Indian Language Technology,
Department of Computer Science & Engineering
Indian Institute of Technology Bombay
{anoopk,pb}@cse.iitb.ac.in

Abstract

We explore the use of the *orthographic syllable*, a variable-length consonant-vowel sequence, as a basic unit of translation between *related* languages which use abugida or alphabetic scripts. We show that orthographic syllable level translation significantly outperforms models trained over other basic units (word, morpheme and character) when training over small parallel corpora.

1 Introduction

Related languages exhibit lexical and structural similarities on account of sharing a **common ancestry** (Indo-Aryan, Slavic languages) or being in **prolonged contact** for a long period of time (Indian subcontinent, Standard Average European linguistic areas) (Bhattacharyya et al., 2016). Translation between *related* languages is an important requirement due to substantial government, business and social communication among people speaking these languages. However, most of these languages have few parallel corpora resources, an important requirement for building good quality SMT systems.

Modelling the lexical similarity among related languages is the key to building good-quality SMT systems with limited parallel corpora. *Lexical similarity* implies that the languages share many words with the similar form (spelling/pronunciation) and meaning *e.g.* `blindness` is `andhapana` in Hindi, `aandhaLepaNaa` in Marathi. These words could be cognates, lateral borrowings or loan words from other languages. Translation for such words can be achieved by sub-word level transformations. For instance, lexical similarity can be modelled in the standard SMT pipeline by transliteration of words while decoding (Durrani et al., 2010) or post-processing (Nakov and Tiedemann, 2012; Kunchukuttan et al., 2014).

A different paradigm is to drop the notion of word boundary and consider the character n-gram as the basic unit of translation (Vilar et al., 2007; Tiedemann, 2009a). Such character-level SMT bas been explored for closely related languages like *Bulgarian-Macedonian, Indonesian-Malay* with modest success, with the short context of unigrams being a limiting factor (Tiedemann, 2012). The use of character n-gram units to address this limitation leads to data sparsity for higher order n-grams and provides little benefit (Tiedemann and Nakov, 2013).

In this work, we present a linguistically motivated, variable length unit of translation — **orthographic syllable (OS)** — which provides more context for translation while limiting the number of basic units. The OS consists of one or more consonants followed by a vowel and is inspired from the *akshara*, a consonant-vowel unit, which is the fundamental organizing principle of Indic scripts (Sproat, 2003; Singh, 2006). It can be thought of as an *approximate syllable* with the onset and nucleus, but no coda. While true syllabification is hard, orthographic syllabification can be easily done. Atreya et al. (2016) and Ekbal et al. (2006) have shown that the OS is a useful unit for transliteration involving Indian languages.

We show that orthographic syllable-level trans-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1912–1917,
Austin, Texas, November 1-5, 2016.

lation significantly outperforms character-level and strong word-level and morpheme-level baselines over multiple related language pairs (Indian as well as others). Character-level approaches have been previously shown to work well for language pairs with high lexical similarity. Our major finding is that OS-level translation outperforms other approaches even when the language pairs have relatively less lexical similarity or belong to different language families (but have sufficient contact relation).

2 Orthographic Syllabification

The *orthographic syllable* is a sequence of one or more consonants followed by a vowel, *i.e* a C^+V unit. We describe briefly procedures for orthographic syllabification of Indian scripts and non-Indic alphabetic scripts. Orthographic syllabification cannot be done for languages using *logographic* and *abjad* scripts as these scripts do not have vowels.

Indic Scripts: Indic scripts are *abugida* scripts, consisting of consonant-vowel sequences, with a consonant core (C^+) and a dependent vowel (*matra*). If no vowel follows a consonant, an implicit *schwa* vowel [IPA: ə] is assumed. Suppression of *schwa* is indicated by the *halanta* character following a consonant. This script design makes for a straightforward syllabification process as shown in the following example. *e.g.* लक्षमी $\left(\frac{lakShamI}{CVCCCVCV}\right)$ is segmented as ल क्ष मी $\left(\frac{la\ \ kSha\ \ mI}{CV\ \ CCV\ \ CV}\right)$. There are two exceptions to this scheme: (i) Indic scripts distinguish between dependent vowels (vowel diacritics) and independent vowels, and the latter will constitute an OS on its own. *e.g.* मुम्बई $(mumbaI) \rightarrow$ मु म्ब ई $(mu\ \ mba\ \ I)$ (ii) The characters *anusvaara* and *chandrabindu* are part of the OS to the left if they represents nasalization of the vowel/consonant or start a new OS if they represent a nasal consonant. Their exact role is determined by the character following the *anusvaara*.

Non-Indic Alphabetic Scripts: We use a simpler method for the alphabetic scripts used in our experiments (Latin and Cyrillic). The OS is identified by a C^+V^+ sequence. *e.g. lakshami→la ksha mi*, *mumbai→mu mbai*. The OS could contains multiple terminal vowel characters representing long vowels (*oo* in *cool*) or diphthongs (*ai* in *mumbai*). A vowel starting a word is considered to be an OS.

Basic Unit	Example	Transliteration
Word	घरासमोरचा	gharAsamoracA
Morph Segment	घरा समोर चा	gharA samora cA
Orthographic Syllable	घ रा स मो र चा	gha rA sa mo racA
Character unigram	घ र ा स म ो र च ा	gha r A sa m o ra c A
Character 3-gram	घरा समो रचा	gharA samo rcA

something that is in front of home: ghara=home, samora=front, cA=of

Table 1: Various translation units for a Marathi word

3 Translation Models

We compared the orthographic syllable level model (O) with models based on other translation units that have been reported in previous work: word (W), morpheme (M), unigram (C) and trigram characters. Table 1 shows examples of these representations.

The first step to build these translation systems is to transform sentences to the correct representation. Each word is segmented as the per the unit of representation, punctuations are retained and a special *word boundary marker* character (_) is introduced to indicate word boundaries as shown here:

W: राजू , घराबाहेर जाऊ नको .
O: रा जू _ , _ घ रा बा हे र _ जा ऊ _ न को _ .

For all units of representation, we trained phrase-based SMT (PBSMT) systems. Since related languages have similar word order, we used distance based distortion model and monotonic decoding. For character and orthographic syllable level models, we use higher order (10-gram) languages models since data sparsity is a lesser concern due to small vocabulary size (Vilar et al., 2007). As suggested by Nakov and Tiedemann (2012), we used word-level tuning for character and orthographic syllable level models by post-processing n-best lists in each tuning step to calculate the usual word-based BLEU score.

While decoding, the word and morpheme level systems will not be able to translate OOV words. Since the languages involved share vocabulary, we transliterate the untranslated words resulting in the post-edited systems W_X and M_X corresponding to the systems W and M respectively. Following decoding, we used a simple method to regenerate words from sub-word level units: Since we represent word boundaries using a word boundary marker, we

IA→IA		DR→DR		IA→DR	
ben-hin	52.30	mal-tam	39.04	hin-mal	33.24
pan-hin	67.99	tel-mal	39.18	**DR→IA**	
kok-mar	54.51			mal-hin	33.24

IA: Indo-Aryan, DR: Dravidian

Table 2: Language pairs used in experiments along with Lexical Similarity between them, in terms of LCSR between training corpus sentences

simply concat the output units between consecutive occurrences of the marker character.

4 Experimental Setup

Languages: Our experiments primarily concentrated on multiple language pairs from the two major language families of the Indian sub-continent (Indo-Aryan branch of Indo-European and Dravidian). These languages have been in contact for a long time, hence there are many lexical and grammatical similarities among them, leading to the subcontinent being considered a *linguistic area* (Emeneau, 1956). Specifically, there is overlap between the vocabulary of these languages to varying degrees due to cognates, language contact and loanwords from Sanskrit (throughout history) and English (in recent times). Table 2 lists the languages involved in the experiments and provides an indication of the lexical similarity between them in terms of the Longest Common Subsequence Ratio (LCSR) (Melamed, 1995) between the parallel training sentences at character level. All these language have a rich inflectional morphology with Dravidian languages, and Marathi and Konkani to some degree, being agglutinative. *kok-mar* and *pan-hin* have a high degree of lexical similarity.

Dataset: We used the multilingual ILCI corpus for our experiments (Jha, 2012), consisting of a modest number of sentences from tourism and health domains. The data split is as follows – *training: 44,777, tuning 1K, test: 2K* sentences. Language models for word-level systems were trained on the target side of training corpora plus monolingual corpora from various sources [hin: 10M (Bojar et al., 2014), tam: 1M (Ramasamy et al., 2012), mar: 1.8M (news websites), mal: 200K (Quasthoff et al., 2006) sentences]. We used the target language side of the parallel corpora for character, morpheme and OS level LMs.

System details: PBSMT systems were trained using the *Moses* system (Koehn et al., 2007), with the *grow-diag-final-and* heuristic for extracting phrases, and Batch MIRA (Cherry and Foster, 2012) for tuning (default parameters). We trained 5-gram LMs with Kneser-Ney smoothing for word and morpheme level models and 10-gram LMs for character and OS level models. We used the *BrahmiNet* transliteration system (Kunchukuttan et al., 2015) for post-editing, which is based on the transliteration Module in Moses (Durrani et al., 2014). We used unsupervised morphological segmenters trained with *Morfessor* (Virpioja et al., 2013) for obtaining morpheme representations. The unsupervised morphological segmenters were trained on the ILCI corpus and the Leipzig corpus (Quasthoff et al., 2006).The morph-segmenters and our implementation of orthographic syllabification are made available as part of the *Indic NLP Library*[1].

Evaluation: We use BLEU (Papineni et al., 2002) and Le-BLEU (Virpioja and Grönroos, 2015) for evaluation. Le-BLEU does fuzzy matches of words and hence is suitable for evaluating SMT systems that perform transformation at the sub-word level.

5 Results and Discussion

This section discusses the results on Indian and non-Indian languages and cross-domain translation.

Comparison of Translation Units: Table 3 compares the BLEU scores for various translation systems. The orthographic syllable level system is clearly better than all other systems. It significantly outperforms the character-level system (by 46% on an average). The character-based system is competitive only for highly lexically similar language pairs like *pan-hin* and *kok-mar*. The system also outperforms two strong baselines which address data sparsity: (a) a word-level system with transliteration of OOV words (10% improvement), (b) a morph-level system with transliteration of OOV words (5% improvement). The OS-level representation is more beneficial when morphologically rich

[1] http://anoopkunchukuttan.github.io/indic_nlp_library

	W	W_X	M	M_X	C	O
ben-hin	31.23	32.79	32.17	32.32	27.95	**33.46**
pan-hin	68.96	71.71	71.29	71.42	71.26	**72.51**
kok-mar	21.39	21.90	22.81	22.82	19.83	**23.53**
mal-tam	6.52	7.01	7.61	7.65	4.50	**7.86**
tel-mal	6.62	6.94	7.86	7.89	6.00	**8.51**
hin-mal	8.49	8.77	9.23	9.26	6.28	**10.45**
mal-hin	15.23	16.26	17.08	17.30	12.33	**18.50**

Table 3: Results - ILCI corpus (% BLEU). The reported scores are:- **W**: word-level, $\mathbf{W}_X$: word-level followed by transliteration of OOV words, **M**: morph-level, $\mathbf{M}_X$: morph-level followed by transliteration of OOV morphemes, **C**: character-level, **O:** orthographic syllable. The values marked in bold indicate the best scores for the language pair.

	C	O	M	W
ben-hin	0.71	0.63	0.58	0.40
pan-hin	0.72	0.70	0.64	0.50
kok-mar	0.74	0.68	0.63	0.64
mal-tam	0.77	0.71	0.56	0.46
tel-mal	0.78	0.65	0.52	0.45
hin-mal	0.79	0.59	0.46	-0.02
mal-hin	0.71	0.61	0.45	0.37

Table 4: Pearson's correlation coefficient between lexical similarity and translation accuracy (both in terms of LCSR at character level). *This was computed over the test set between: (ii) sentence level lexical similarity between source and target sentences and (ii) sentence level translation match between hypothesis and reference.*

languages are involved in translation. Significantly, OS-level translation is also the best system for translation between languages of different language families. The Le-BLEU scores also show the same trend as BLEU scores, but we have not reported it due to space limits. There are a very small number of untranslated OSes, which we handled by simple mapping of untranslated characters from source to target script. This barely increased translation accuracy (0.02% increase in BLEU score).

Why is OS better than other units?: The improved performance of OS level representation can be attributed to the following factors:

One, the number of basic translation units is limited and small compared to word-level and morpheme-level representations. For word-level representation, the number of translation units can increase with corpus size, especially for morphologically rich languages which leads to many OOVs. Thus, OS-level units **address data sparsity**.

	W_X	M_X	C	O
ben-hin	*Corpus not available*			
pan-hin	61.56	**59.75**	58.07	58.48
kok-mar	19.32	18.32	17.97	**19.65**
mal-tam	**5.88**	6.02	4.12	**5.88**
tel-mal	3.19	**4.07**	3.11	3.77
hin-mal	5.20	6.00	3.85	**6.26**
mal-hin	9.68	11.44	8.42	**13.32**

Table 5: Results: Agricuture Domain (% BLEU)

Two, while character level representation too does not suffer from data sparsity, we observe that the translation accuracy is highly correlated to lexical similarity (Table 4). The high correlation of character-level system and lexical similarity explains why character-level translation performs nearly as well other methods for language pairs which have high lexical similarity, but performs badly otherwise. On the other hand, the OS-level representation has lesser correlation with lexical similarity and sits somewhere between character-level and word/morpheme level systems. Hence it is able to make **generalizations beyond simple character level mappings**. We observed that OS-level representation was able to correctly generate words whose translations are not cognate with the source language. This is an important property since function words and suffixes tend to be less similar lexically across languages.

Can improved translation performance be explained by longer basic translation units? To verify this, we trained translation systems with character trigrams as basic units. We chose trigrams since the average length of the OS was 3-5 characters for the languages we tested with. The translation accuracies were far less than even unigram representation. The number of unique basic units was about 8-10 times larger than orthographic syllables, thus making data sparsity an issue again. So, improved translation performance **cannot be attributed to longer n-gram units alone**.

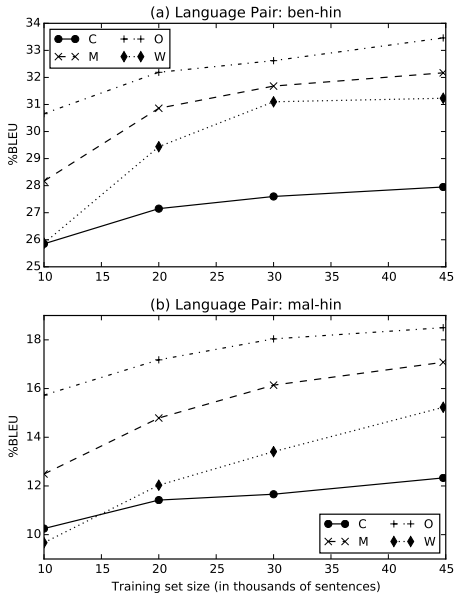


Figure 1: Effect of training data size on translation accuracy for different basic units

	Corpus Stats	Lex-Sim	W	C	O
bul-mac	(150k,1k,2k)	62.85	21.20	20.61	**21.38**
dan-swe	(150k,1k,2k)	63.39	35.13	35.36	**35.46**
may-ind	(137k,1k,2k)	73.54	61.33	60.50	**61.24**

Table 6: Translation among non-Indic languages (%BLEU). Corpus Stats show (train,tune,test) split

Robustness to Domain Change: We also tested the translation models trained on tourism & health domains on an agriculture domain test set of 1000 sentences. In this cross-domain translation scenario too, the OS level model outperforms most units of representation. The only exceptions are the *pan-hin* and *tel-mal* language pairs for the system $\mathbf{M}_X$ (accuracies of the OS-level system are within 10% of the $\mathbf{M}_X$ system). Since the word level model depends on coverage of the lexicon, it is highly domain dependent, whereas the sub-word units are not. So, even unigram-level models outperform word-level models in a cross-domain setting.

Experiments with non-Indian languages: Table 6 shows the corpus statistics and our results for translation between some related non-Indic language pairs (Bulgarian-Macedonian, Danish-Swedish, Malay-Indonesian). OS level representation outperforms character and word level representation, though the gains are not as significant as Indic language pairs. This could be due to short length of sentences in training corpus [OPUS movie subtitles (Tiedemann, 2009b)] and high lexical similarity between the language pairs. Further experiments between less lexically related languages on general parallel corpora will be useful.

Effect of training data size: For different training set sizes, we trained SMT systems with various representation units (Figure 1 shows the learning curves for two language pairs). BPE level models are consistently better than word as well as morph-level models, and are competitive or better than OS level models. Note that *bn-hi* is a relatively morphologically simpler language where BPE is just competitive with OS over the complete dataset too as discussed earlier.

6 Conclusion & Future Work

We focus on the task of translation between *related languages*. This aspect of MT research is important to make available translation technologies to language pairs with limited parallel corpus, but huge potential translation requirements. We propose the use of the *orthographic syllable*, a variable-length, linguistically motivated, approximate syllable, as a basic unit for translation between related languages. We show that it significantly outperforms other units of representation, over multiple language pairs, spanning different language families, with varying degrees of lexical similarity and is robust to domain changes too. This opens up the possibility of further exploration of sub-word level translation units *e.g.* segments learnt using byte pair encoding (Sennrich et al., 2016).

Acknowledgments

We thank Arjun Atreya for inputs regarding orthographic syllables. We thank the Technology Development for Indian Languages (TDIL) Programme and the Department of Electronics & Information Technology, Govt. of India for their support.

References

Arjun Atreya, Swapnil Chaudhari, Pushpak Bhattacharyya, and Ganesh Ramakrishnan. 2016. Value the vowels: Optimal transliteration unit selection for machine. In *Unpublished, private communication with authors*.

Pushpak Bhattacharyya, Mitesh Khapra, and Anoop Kunchukuttan. 2016. Statistical machine translation between related languages. In *NAACL Tutorials*.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp – Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2010. Hindi-to-Urdu machine translation through transliteration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Nadir Durrani, Hieu Hoang, Philipp Koehn, and Hassan Sajjad. 2014. Integrating an unsupervised transliteration model into Statistical Machine Translation. *EACL 2014*.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL on Main conference poster sessions*.

Murray B Emeneau. 1956. India as a lingustic area. *Language*.

Girish Nath Jha. 2012. The TDIL program and the Indian Language Corpora Initiative. In *Language Resources and Evaluation Conference*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*.

Anoop Kunchukuttan, Ratish Pudupully, Rajen Chatterjee, Abhijit Mishra, and Pushpak Bhattacharyya. 2014. The IIT Bombay SMT System for ICON 2014 Tools contest. In *NLP Tools Contest at ICON 2014*.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-Net: A transliteration and script conversion system for languages of the Indian subcontinent.

I Dan Melamed. 1995. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Third Workshop on Very Large Corpora*.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics*.

Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on language resources and evaluation*.

Loganathan Ramasamy, Ondřej Bojar, and Zdeněk Žabokrtský. 2012. Morphological Processing for English-Tamil Statistical Machine Translation. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *ACL*.

Anil Kumar Singh. 2006. A computational phonetic model for Indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*.

Richard Sproat. 2003. A formal computational analysis of Indic scripts. In *International symposium on indic scripts: past and future, Tokyo*.

Jörg Tiedemann and Preslav Nakov. 2013. Analyzing the use of character-level translation with sparse and noisy datasets. In *RANLP*.

Jörg Tiedemann. 2009a. Character-based PBSMT for closely related languages. In *Proceedings of the 13th Conference of the European Association for Machine Translation*.

Jörg Tiedemann. 2009b. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*.

Jörg Tiedemann. 2012. Character-based pivot translation for under-resourced languages and domains. In *EACL*.

David Vilar, Jan-T Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Sami Virpioja and Stig-Arne Grönroos. 2015. Lebleu: N-gram-based translation evaluation score for morphologically complex languages. In *WMT 2015*.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge

Nicholas Locascio
CSAIL, MIT
njl@mit.edu

Karthik Narasimhan
CSAIL, MIT
karthikn@mit.edu

Eduardo DeLeon
CSAIL, MIT
edeleon4@mit.edu

Nate Kushman
Microsoft
nate@kushman.org

Regina Barzilay
CSAIL, MIT
regina@csail.mit.edu

Abstract

This paper explores the task of translating natural language queries into regular expressions which embody their meaning. In contrast to prior work, the proposed neural model does not utilize domain-specific crafting, learning to translate directly from a parallel corpus. To fully explore the potential of neural models, we propose a methodology for collecting a large corpus[1] of regular expression, natural language pairs. Our resulting model achieves a performance gain of 19.6% over previous state-of-the-art models.

1 Introduction

This paper explores the task of translating natural language text queries into regular expressions which embody their meaning. Regular expressions are built into many application interfaces, yet most users of these applications have difficulty writing them (Friedl, 2002). Thus a system for automatically generating regular expressions from natural language would be useful in many contexts. Furthermore, such technologies can ultimately scale to translate into other formal representations, such as program scripts (Raza et al., 2015).

Prior work has demonstrated the feasibility of this task. Kushman and Barzilay (2013) proposed a model that learns to perform the task from a parallel corpus of regular expressions and the text descriptions. To account for the given representational disparity between formal regular expressions and natural language, their model utilizes a domain specific component which computes the semantic equivalence between two regular expressions. Since their model relies heavily on this component, it cannot be readily applied to other formal representations where such semantic equivalence calculations are not possible.

In this paper, we reexamine the need for such specialized domain knowledge for this task. Given the same parallel corpus used in Kushman and Barzilay (2013), we use an LSTM-based sequence to sequence neural network to perform the mapping. Our model does not utilize semantic equivalence in any form, or make any other special assumptions about the formalism. Despite this and the relatively small size of the original dataset (824 examples), our neural model exhibits a small 0.1% boost in accuracy.

To further explore the power of neural networks, we created a much larger public dataset, **NL-RX**. Since creation of regular expressions requires specialized knowledge, standard crowd-sourcing methods are not applicable here. Instead, we employ a two-step generate-and-paraphrase procedure that circumvents this problem. During the generate step, we use a small but expansive manually-crafted grammar that translates regular expression into natural language. In the paraphrase step, we rely on crowd-sourcing to paraphrase these rigid descriptions into more natural and fluid descriptions. Using this methodology, we have constructed a corpus of 10,000 regular expressions, with corresponding verbalizations.

Our results demonstrate that our sequence to sequence model significantly outperforms the domain specific technique on the larger dataset, reaching a

[1] The corpus and code used in this paper is available at https://github.com/nicholaslocascio/deep-regex

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1918–1923,
Austin, Texas, November 1-5, 2016.

gain of 19.6% over of the state-of-the-art technique.

2 Related Work

Regular Expressions from Natural Language There have been several attempts at generating regular expressions from textual descriptions. Early research into this task used rule-based techniques to create a natural language interface to regular expression writing (Ranta, 1998). Our work, however, is closest to Kushman and Barzilay (2013). They learned a semantic parsing translation model from a parallel dataset of natural language and regular expressions. Their model used a regular expression-specific semantic unification technique to disambiguate the meaning of the natural language descriptions. Our method is similar in that we require only description and regex pairs to learn.r However, we treat the problem as a direct translation task without applying any domain-specific knowledge.

Neural Machine Translation Recent advances in neural machine translation (NMT) (Bahdanau et al., 2014; Devlin et al., 2014) using the framework of sequence to sequence learning (Sutskever et al., 2014) have demonstrated the effectiveness of deep learning models at capturing and translating language semantics. In particular, recurrent neural networks augmented with attention mechanisms (Luong et al., 2015) have proved to be successful at handling very long sequences. In light of these successes, we chose to model regular expression generation as a neural translation problem.

3 Regex Generation as Translation

We use a Recurrent Neural Network (RNN) with attention (Mnih et al., 2014) for both encoding and decoding (Figure 1).

Let $W = w_1, w_2 ... w_m$ be the input text description where each w_i is a word in the vocabulary. We wish to generate the regex $R = r_1, r_2, ... r_n$ where each r_i is a character in the regex.

We use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells in our model, the transition equations for which can be summarized as:

$$\begin{aligned} i_t &= \sigma(U^{(i)} x_t + V^{(i)} h_{t-1} + b^{(i)}), \\ f_t &= \sigma(U^{(f)} x_t + V^{(f)} h_{t-1} + b^{(f)}), \\ o_t &= \sigma(U^{(o)} x_t + V^{(o)} h_{t-1} + b^{(o)}) \\ z_t &= \tanh(U^{(z)} x_t + V^{(z)} h_{t-1} + b^{(z)}) \\ c_t &= i_t \odot z_t + f_t \odot c_{t-1} \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

where σ represents the sigmoid function and $\odot$ is elementwise multiplication. i_t refers to the input gate, f_t is the forget gate, and o_t is the output gate at each time step. The U and V variables are weight matrices for each gate while the b variables are the bias parameters. The input x_t is a word (w_t) for the encoder and the previously generated character r_{t-1} for the decoder.

The attention mechanism is essentially a 'soft' weighting over the encoder's hidden states during decoding:

$$\alpha_t(e) = \frac{\exp(\text{score}(h_t, h_e))}{\sum_{e'} \exp(\text{score}(h_t, h_{e'}))}$$

where h_e is a hidden state in the encoder and *score* is the scoring function. We use the general attention matrix weight (as described in (Luong et al., 2015)) for our scoring function. The outputs of the decoder r_t are generated using a final softmax layer.

Our model is six layers deep, with one word embedding layer, two encoder layers, two decoder layers, and one dense output layer. Our encoder and decoder layers use a stacked LSTM architecture with a width of 512 nodes. We use a global attention mechanism (Bahdanau et al., 2014), which considers all hidden states of the encoder when computing the model's context vector. We perform standard dropout during training (Srivastava et al., 2014) after every LSTM layer with dropout probability equal to 0.25. We train for 20 epochs, utilizing a minibatch size of 32, and a learning-rate of 1.0. The learning rate is decayed by a factor 0.5 if evaluation perplexity does not increase.

4 Creating a Large Corpus of Natural Language / Regular Expression Pairs

Previous work in regular expression generation has used fairly small datasets for training and evaluation.

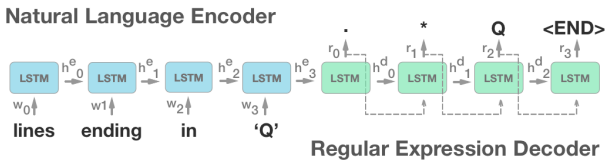


Figure 1: Deep-Regex Encoder-Decoder setup. Blue cells represent the encoder and the green ones represent the decoder.

Non-Terminals		
x & y → x and y	x | y → x or y	∼(x) → not x
.*x.*y → x followed by y	.*x.* → contains x	x{N,} → x, N or more times
x& y& z → x and y and z	x | y | z → x or y or z	x{1,N} → x, at most N times
x.* → starts with x	.*x → ends with x	\b x\b → words with x
(x)+ → x, at least once	(x)* → x, zero or more times	x → only x

Terminals		
[AEIOUaeiou] → a vowel	[0-9] → a number	word → the string 'word'
[A-Z] → an uppercase letter	[a-z] → a lowercase letter	. → a character

Table 1: Regex → Synthetic Grammar for Data Generation

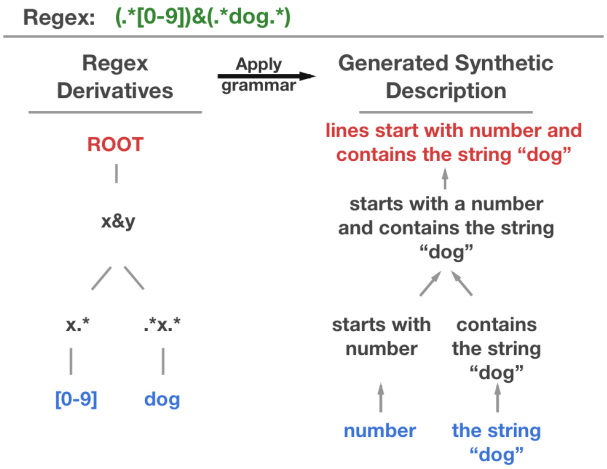


Figure 2: Process for generating Synthetic Descriptions from Regular Expressions. Grammar rules from Table 1 are applied to a node's children and the resulting string is passed to the node's parent.

Synthetic:	lines not words with starting with a capital letter
Paraphrased:	lines that do not contain words that begin with a capital letter
Regex:	∼ (\b([A-Z])(.*)\b)

Table 2: NL-RX Text Descriptions and Regular Expression

In order to fully utilize the power of neural translation models, we create a new large corpus of regular expression, natural language pairs titled **NL-RX**.

The challenge in collecting such corpora is that typical crowdsourcing workers do not possess the specialized knowledge to write regular expressions. To solve this, we employ a two-step generate-and-paraphrase procedure to gather our data. This technique is similar to the methods used by Wang et al. (2015) to create a semantic parsing corpus.

In the generate step, we generate regular expression representations from a small manually-crafted grammar (Table 1). Our grammar includes 15 non-terminal derivations and 6 terminals and of both basic and high-level operations. We identify these via frequency analysis of smaller datasets from previous work (Kushman and Barzilay, 2013). Every grammar rule has associated verbalizations for both regular expressions and language descriptions. We use this grammar to stochastically generate regular expressions and their corresponding synthetic language descriptions. This generation process is shown in Figure 2.

While the automatically generated descriptions are semantically correct, they do not exhibit richness and variability of human-generated descriptions. To obtain natural language (non-synthetic) descriptions, we perform the paraphrase step. In this step, Mechanical Turk (Amazon, 2003) human workers paraphrase the generated synthetic descriptions into the fluent verbalizations.

NL-RX Using the procedure described above, we create a new public dataset (NL-RX) comprising of 10,000 regular expressions and their corresponding natural language descriptions. Table 2 shows an example from our dataset.

Our data collection procedure enables us to create a substantially larger and more varied dataset than previously possible. Employing standard crowdsource workers to paraphrase is more cost-efficient and scalable than employing professional regex programmers, enabling us to create a much larger dataset. Furthermore, our stochastic generation of regular expressions from a grammar results in a more varied dataset because it is not subject to the bias of human workers who, in previous work, wrote many duplicate examples (see Results).

Corpora Statistics Our seed regular expression grammar (Table 1), covers approximately 85% of the original KB13 regular expressions. Additionally, NL-RX contains exact matches with 30.1% of the original KB13 dataset regular expressions. This means that 248 of the 824 regular expressions in the

Verbalization	Frequency
'the word x'	12.6%
'x before y'	9.1%
'x or y'	7.7%
'x, at least once'	6.2%
'a vowel'	5.3%

Table 3: Top Frequent Verbalizations from NL-RX

KB13 dataset were also in our dataset. The average length of regular expressions in NL-RX is 25.9 characters, the average in the KB13 dataset is 19.7 characters. We also computed the grammar breakdown of our NL-RX. The top 5 occurring terminals in our generated regular expressions are those corresponding with the verbalizations shown in Table 3.

Crowdsourcing details We utilize Mechanical Turk for our crowdsource workers. A total of 243 workers completed the 10,000 tasks, with an average task completion time of 101 seconds. The workers proved capable of handling complex and awkward phrasings, such as the example in Table 2, which is one of the most difficult in the set.

We applied several quality assurance measures on the crowdsourced data. Firstly, we ensured that our workers performing the task were of high quality, requiring a record of 97% accuracy over at least 1000 other previous tasks completed on Mechanical Turk. In addition, we ran automatic scripts that filtered out bad submissions (e.g. submissions shorter than 5 characters). In all, we rejected 1.1% of submissions, which were resubmitted for another worker to complete. The combination of these measures ensured a high quality dataset, and we confirmed this by performing a manual check of 100 random examples. This manual check determined that approximately 89% of submissions were a correct interpretation, and 97% were written in fluent English.

5 Experiments

Datasets We split the 10,000 regexp and description pairs in NL-RX into 65% train, 10% dev, and 25% test sets.

In addition, we also evaluate our model on the dataset used by Kushman and Barzilay (2013) (KB13), although it contains far fewer data points (824). We use the 75/25 train/test split used in their work in order directly compare our performance to theirs.

Training We perform a hyper-parameter grid-search (on the dev set), to determine our model hyper-parameters: learning-rate = 1.0, encoder-depth = 2, decoder-depth = 2, batch size = 32, dropout = 0.25. We use a Torch (Collobert et al., 2002) implementation of attention sequence to sequence networks from (Kim, 2016). We train our models on the train set for 20 epochs, and choose the model with the best average loss on the dev set.

Evaluation Metric To accurately evaluate our model, we perform a *functional* equality check called DFA-Equal. We employ functional equality because there are many ways to write equivalent regular expressions. For example, (a|b) is functionally equivalent to (b|a), despite their string representations differing. We report DFA-Equal accuracy as our model's evaluation metric, using Kushman and Barzilay (2013)'s implementation to directly compare our results.

Baselines We compare our model against two baselines:

BoW-NN: BoW-NN is a simple baseline that is a Nearest Neighbor classifier using Bag Of Words representation for each natural language description. For a given test example, it finds the closest cosine-similar neighbor from the training set and uses the regexp from that example for its prediction.

Semantic-Unify: Our second baseline, Semantic-Unify, is the previous state-of-the-art model from (Kushman and Barzilay, 2013), explained above. [2]

6 Results

Our model significantly outperforms the baselines on the NL-RX dataset and achieves comparable performance to *Semantic Unify* on the KB13 dataset (Table 4). Despite the small size of KB13, our model achieves state-of-the-art results on this very resource-constrained dataset (824 examples). Using NL-RX, we investigate the impact of training data size on our model's accuracy. Figure 3 shows how

[2]We trained and evaluated Semantic-Unify in consultation with the original authors.

Models	NL-RX-Synth		NL-RX-Turk		KB13
	Dev	Test	Dev	Test	Test
BoW NN	31.7%	30.6%	18.2%	16.4%	48.5%
Semantic-Unify	41.2%	46.3%	39.5%	38.6%	65.5%
Deep-RegEx	**85.75%**	**88.7%**	**61.2%**	**58.2%**	**65.6%**

Table 4: DFA-Equal accuracy on different datasets. **KB13**: Dataset from Kushman and Barzilay(2013), **NL-RX-Synth**: NL Dataset with original synthetic descriptions, **NL-RX-Turk**: NL Dataset with Mechanical Turk paraphrased descriptions. Best scores are in bold.

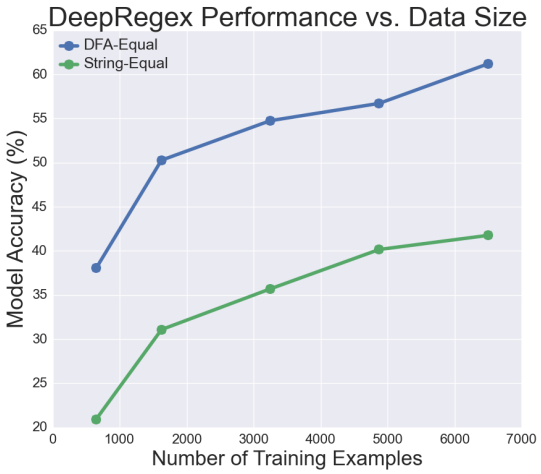


Figure 3: Our model's performance, like many deep learning models, increases significantly with larger datasets. **String-Equal**:Accuracy on direct string match, **DFA-Equal**:Accuracy using the DFA-Equal evaluation.

our model's performance improves as the number of training examples grows.

Differences in Datasets Keeping the previous section in mind, a seemingly unusual finding is that the model's accuracy is higher for the smaller dataset, KB13, than for the larger dataset, NL-RX-Turk. On further analysis, we learned that the KB13 dataset is a much less varied and complex dataset than NL-RX-Turk. KB13 contains many duplicates, with only 45% of its regular expressions being unique. This makes the translation task easier because over half of the correct test predictions will be exact repetitions from the training set. In contrast, NL-RX-Turk does not suffer from this variance problem and contains 97% unique regular expressions. The relative easiness of the KB13 dataset is further illustrated by the high performance of the Nearest-Neighbor baselines on the KB13 dataset.

7 Conclusions

In this paper we demonstrate that generic neural architectures for generating regular expressions outperform customized, heavily engineered models. The results suggest that this technique can be employed to tackle more challenging problems in broader families of formal languages, such as mapping between language description and program scripts. We also have created a large parallel corpus of regular expressions and natural language queries using typical crowd-sourcing workers, which we make available publicly.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback and suggestions.

References

[Amazon2003] Amazon. 2003. Mechanical turk. https://mturk.com.

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Collobert et al.2002] Ronan Collobert, Samy Bengio, and Johnny Marithoz. 2002. Torch: A modular machine learning software library. https://torch.ch.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

[Friedl2002] Jeffrey EF Friedl. 2002. *Mastering regular expressions*. " O'Reilly Media, Inc.".

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kim2016] Yoon Kim. 2016. Seq2seq-attn. https://github.com/harvardnlp/seq2seq-attn.

[Kushman and Barzilay2013] Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. North American Chapter of the Association for Computational Linguistics (NAACL).

[Luong et al.2015] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages

1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

[Mnih et al.2014] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

[Ranta1998] Aarne Ranta. 1998. A multilingual natural-language interface to regular expressions. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 79–90. Association for Computational Linguistics.

[Raza et al.2015] Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. *International Joint Conference on Artificial Intelligence (IJCAI)*.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Wang et al.2015] Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. *Association for Computational Linguistics (ACL)*.

Supervised Keyphrase Extraction as Positive Unlabeled Learning

Lucas Sterckx[*], **Cornelia Caragea**[†], **Thomas Demeester**[*], **Chris Develder**[*]
[*]Ghent University – iMinds, Ghent, Belgium
{lusterck,tdmeeste,cdvelder}@intec.ugent.be
[†]University of North Texas, Texas, USA
cornelia.caragea@unt.edu

Abstract

The problem of noisy and unbalanced training data for supervised keyphrase extraction results from the subjectivity of keyphrase assignment, which we quantify by crowdsourcing keyphrases for news and fashion magazine articles with many annotators per document. We show that annotators exhibit substantial disagreement, meaning that single annotator data could lead to very different training sets for supervised keyphrase extractors. Thus, annotations from single authors or readers lead to noisy training data and poor extraction performance of the resulting supervised extractor. We provide a simple but effective solution to still work with such data by reweighting the importance of unlabeled candidate phrases in a two stage Positive Unlabeled Learning setting. We show that performance of trained keyphrase extractors approximates a classifier trained on articles labeled by multiple annotators, leading to higher average F_1scores and better rankings of keyphrases. We apply this strategy to a variety of test collections from different backgrounds and show improvements over strong baseline models.

1 Introduction

Keyphrase extraction is the task of extracting a selection of phrases from a text document to concisely summarize its contents. Applications of keyphrases range from summarization (D'Avanzo et al., 2004) to contextual advertisement (Yih et al., 2006) or simply as aid for navigation through large text corpora.

Existing work on automatic keyphrase extraction can be divided in supervised and unsupervised approaches. While unsupervised approaches are domain independent and do not require labeled training data, supervised keyphrase extraction allows for more expressive feature design and is reported to outperform unsupervised methods on many occasions (Kim et al., 2012; Caragea et al., 2014). A requirement for supervised keyphrase extractors is the availability of labeled training data. In literature, training collections for supervised keyphrase extraction are generated in different settings. In these collections, keyphrases for text documents are either supplied by the authors or their readers. In the first case, authors of academic papers or news articles assign keyphrases to their content to enable fast indexing or to allow for the discovery of their work in electronic libraries (Frank et al., 1999; Hulth, 2003; Bulgarov and Caragea, 2015). Other collections are created by crowdsourcing (Marujo et al., 2012) or based on explicit deliberation by a small group of readers (Wan and Xiao, 2008). A minority of test collections provide multiple opinions per document, but even then the amount of opinions per document is kept minimal (Nguyen and Kan, 2007).

The traditional procedure for supervised keyphrase extraction is reformulating the task as a binary classification of keyphrase candidates. Supervision for keyphrase extraction faces several shortcomings. Candidate phrases (generated in a separate candidate generation procedure), which are not annotated as keyphrases, are seen as non-keyphrase and are used as negative training data for the supervised classifiers. First, on many occasions these negative phrases outnumber true keyphrases many times, creating an unbalanced

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1924–1929,
Austin, Texas, November 1-5, 2016.

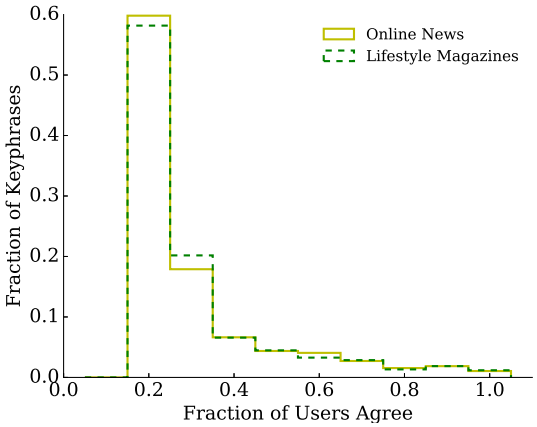


Figure 1: This plot shows the fraction of all keyphrases from the training set agreed upon versus the fraction of all annotators.

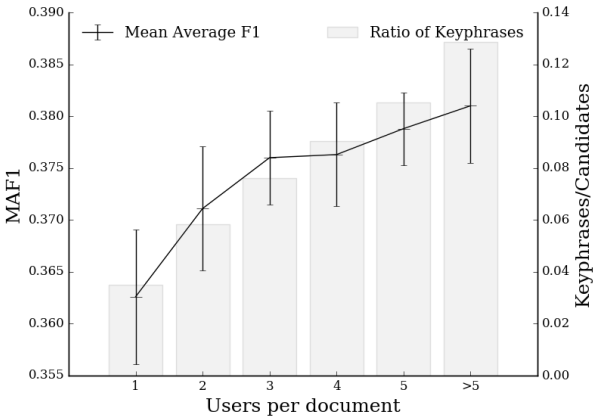


Figure 2: Effect of overlap on extraction performance.

training set (Frank et al., 1999; Kim et al., 2012). Second, as Frank et al. (1999) noted: authors do not always choose keyphrases that best describe the content of their paper, but they may choose phrases to slant their work a certain way, or to maximize its chance of being noticed by searchers. Another problem is that keyphrases are inherently subjective, i.e., keyphrases assigned by one annotator are not the only correct ones (Nguyen and Kan, 2007). These assumptions have consequences for training, developing and evaluating supervised models. Unfortunately, a large collection of annotated documents by reliable annotators with high overlap per document is missing, making it difficult to study disagreement between annotators or the resulting influence on trained extractors, as well as to provide a reliable evaluation setting. In this paper, we address these problems by creating a large test collection of articles with many different opinions per article, evaluate the effect on extraction performance, and present a procedure for supervised keyphrase extraction with noisy labels.

2 Noisy Training Data for Supervised Keyphrase Extraction

A collection of online news articles and lifestyle magazine articles was presented to a panel of 357 annotators of various ages and backgrounds, (selected and managed by iMinds - Living Labs[1]) who were trained to select a *limited number* of *short* phrases that concisely reflect the documents' contents. No prior limits or constraints were set on the amount, length, or form of the keyphrases. Each document was presented multiple times to different users. Each user was assigned with 140 articles, but was not required to finish the full assignment. The constructed training collections have on average six and up to ten different opinions per article.

We visualize the agreement on single keyphrases in Figure 1, which shows the fraction of annotated keyphrases versus agreement by the complete set of readers. Agreement on keyphrases appears low, as a large fraction of all keyphrases assigned to documents (>50%) are only assigned by single annotators. We note that different sets of keyphrases by different annotators are the result of the subjectiveness of the task, of different interpretations by the annotators of the document, but also because of semantically equivalent keyphrases being annotated in different forms, e.g., "Louis Michel" vs. "Prime Minister Louis Michel" or "Traffic Collision" vs. "Car Accident".

The observation in Figure 1 has important consequences for training models on keyphrases annotated by a single annotator, since other annotators may have chosen some among the ones that the sin-

[1] https://www.iminds.be/en/succeed-with-digital-research/proeftuinonderzoek/

gle selected annotator did not indicate (and hence these should not be used as negative training data).

A single annotator assigning keyphrases to 100 documents results on average in a training set with 369 positive training instances and 4,981 negative training instances generated by the candidate extractor. When assigning these 100 documents to 9 other annotators, the amount of positive instances increases to 1,258 keyphrases, which means that labels for 889 keyphrase candidates, or 17% of the original negative candidates when training on annotations by a single annotator, can be considered noise and relabeled. As a result, ratios of positive to negative data also change drastically. We visualize the effect of using training data from multiple annotators per document in Figure 2. Classifiers trained on the aggregated training collection with multiple opinions (using all assigned keyphrases at least once as positive training data) perform better on held-out test collections containing only keyphrases of high agreement (assigned by > 2 annotators).

When using keyphrases from many different annotators per document, the amount of positive candidates increases and as a result, the Macro Average F_1 (MAF_1) of the corresponding classifier. We detail our experimental setup and supervised classifier in Section 4.

3 Reweighting Keyphrase Candidates

Observations described in Section 2 indicate that unlabeled keyphrase candidates are not reliable as negative examples by default. A more suitable assumption is to treat supervised keyphrase extraction as Positive Unlabeled Learning, i.e., an incomplete set of positive examples is available as well as a set of unlabeled examples, of which some are positive and others negative. This topic has received much attention as it knows many applications (Ren et al., 2014; du Plessis et al., 2014), but has not been linked to supervised keyphrase extraction. We base our approach on work by Elkan and Noto (2008) and modify the supervised extractor by assigning individual weights to training examples. Instead of assuming the noise to be random, we assign weights depending on the document and the candidate.

By reweighting importance of training samples, we seek to mimic the case of multiple annotators, to model the uncertainty of negative keyphrase candidates, based only on annotations by a single annotator. In a first stage, we train a classifier on the single annotator data and use predictions on the negative or unlabeled candidates, to reweigh training instances. The reweighted training collection is then used to train a second classifier to predict a final ranking or the binary labels of the keyphrase candidates.

Feature	**Definition**
Head match	$head_{keyphrase} == head_{candidate}$
Extent match	$extent_{keyphrase} == extent_{candidate}$
Substring	$head_{keyphrase}$ substring of $head_{candidate}$
Alias	$acronym(head_{keyphrase}) == head_{candidate}$

Table 1: String relation features for coreference resolution

Positive examples are given unit weight and unlabeled examples are duplicated; one copy of each unlabeled keyphrase candidate x is made positive with weight $w(x) = P(keyphrase|x, s = 0)$ and the other copy is made negative with weight $1 - w(x)$ with s indicating whether x is labeled or not.

Instead of assigning this weight as a constant factor of the predictions by the initial classifier as in Elkan and Noto (2008), we found that two modifications allow improving the weight estimate, $w(x) \leq 1$. We normalize probabilities $P(keyphrase, x, s = 0)$ to candidates not included in the initial set of keyphrases per document. Next to this self-predicted probability, we include a simple measure indicating pairwise coreference between unlabeled candidates and known keyphrases in a function $\text{Coref}(candidate, keyphrase) \in \{0, 1\}$, returning 1 if one of the binary indicator features, presented in (Bengtson and Roth, 2008) and shown in Table 1, is present. In this description, the term *head* means the head noun phrase of a candidate or keyphrase and the *extent* is the largest noun phrase headed by the head noun phrase. The self-predicted probability is summed with the output of the coreference resolver and the final weight becomes:

$$w(x) = min\left(1, \frac{P(keyphrase|x)}{\max_{(x',s=0)\in d} P(keyphrase|x')} \cdot + \max_{\forall keyphrase \in d} \text{Coref}(x, keyphrase)\right) \quad (1)$$

with d being a document from the training collection.

	Test Collections				
Name	Online News	Lifestyle Magazines	WWW	KDD	Inspec
Type	Sports Articles	Fashion, Lifestyle	WWW Paper Abstracts	KDD Paper Abstracts	Paper Abstracts
# Documents	1,259	2,202	1,895	1,011	500
# Keyphrases	19,340	29,970	3,922	1,966	4,913
⊘ Keyphrases/User	5.7	4.7	/	/	/
⊘ Keyphrases/Document	15.4	13.7	2.0	1.8	9.8
⊘ Tokens/Document	332	284	164	195	134
⊘ Candidate Keyphrases/Doc.	52	49	47	54	34
1/2/3/3+ -gram distribution (%)	55/27/9/9	58/25/9/8	63/27/8/2	60/28/9/3	13/53/25/9

Table 2: Description of test collections.

Method	Online News		Lifestyle Magazines		WWW		KDD		Inspec	
	MAF_1	P@5	MAF_1	P@5	MAF_1	P@5	MAF_1	P@5	MAF_1	P@5
Single Annotator	.364	.416	.294	.315	.230	.189	.266	.200	.397	.432
Multiple Annotators	.381	.426	.303	.327	/	/	/	/	/	/
Self Training	.366	.417	.301	.317	.236	.190	.269	.196	.401	**.434**
Reweighting (Elkan and Noto, 2008)	.364	.417	.297	.313	.238	.189	**.275**	**.201**	.401	.429
Reweighting +Norm +Coref	**.374**	**.419**	**.305**	**.322**	**.245**	**.194**	**.275**	.200	**.402**	**.434**

Table 3: Mean average F_1 score per document and precision for five most confident keyphrases on different test collections.

4 Experiments and Results

Hasan and Ng (2010) have shown that techniques for keyphrase extraction are inconsistent and need to be tested across different test collections. Next to our collections with multiple opinions (*Online News* and *Lifestyle Magazines*), we apply the reweighting strategy on test collections with sets of author-assigned keyphrases: two sets from CiteSeer abstracts from the World Wide Web Conference (*WWW*) and Knowledge Discovery and Data Mining (*KDD*), similar to the ones used in (Bulgarov and Caragea, 2015). The *Inspec* dataset is a collection of 2,000 abstracts commonly used in keyphrase extraction literature, where we use the ground truth phrases from controlled vocabulary (Hulth, 2003). Descriptive statistics of these test collections are given in Table 2.

We use a rich feature set consisting of statistical, structural, and semantic properties for each candidate phrase, that have been reported as effective in previous studies on supervised extractors (Frank et al., 1999; Hulth, 2003; Kim and Kan, 2009): (i) term frequency, (ii) number of tokens in the phrase, (iii) length of the longest term in the phrase, (iv) number of capital letters in the phrase, (v) the phrase's POS-tags, (vi) relative position of first occurrence, (vii) span (relative last occurrence minus relative first occurrence), (viii) TF*IDF (IDF's trained on large background collections from the same source) and (ix) Topical Word Importance, a feature measuring the similarity between the word-topic topic-document distributions presented in (Sterckx et al., 2015), with topic models trained on background collections from a corresponding source of content.

As classifier we use gradient boosted decision trees implemented in the XGBoost package (Chen and Guestrin, 2016). During development, this classifier consistently outperformed Naive Bayes and linear classifiers like logistic regression or support vector machines.

We compare the reweighting strategy with uniform reweighting and strategies to counter the imbalance or noise of the training collections, such as subsampling, weighting unlabeled training data as in (Elkan and Noto, 2008), and self-training in which only confident initial predictions are used as positive and negative data. For every method, global thresholds are chosen to optimize the macro averaged F_1 per document (MAF_1). Next to the threshold sensitive F_1, we report on ranking quality using the Precision@5 metric.

Results are shown in Table 3 with five-fold cross-validation. To study the effect of reweighting, we limit training collections during folds to 100 documents for each test collection. Our approach consistently improves on single annotator trained classifiers, on one occasion even outperforming a training collection with multiple opinions. Compensating for

imbalance and noise tends to have less effect when the ratio of keyphrases versus candidates is high (as for *Inspec*) or training collection is very large. When the amount of training documents increases, the ratio of noisy versus true negative labels drops.

5 Conclusion

It has been suggested that keyphrase annotation is highly subjective. We present two data sets where we purposely gathered multiple annotations of the same document, as to quantify the limited overlap between keyphrases selected by different annotators. We suggest to treat non-selected phrases as *unlabeled* rather than *negative* training data. We further show that using multiple annotations leads to more robust automatic keyphrase extractors, and propose reweighting of single annotator labels based on probabilities from a first-stage classifier. This reweighting approach outperforms other single-annotator state-of-the-art automatic keyphrase extractors on different test collections, when we normalize probabilities per document and include co-reference indicators.

Acknowledgments

The authors like to thank the anonymous reviewers for their helpful comments. The research presented in this article relates to STEAMER (`http://www.iminds.be/en/projects/2014/07/12/steamer`), a MiX-ICON project facilitated by iMinds Media and funded by IWT and Innoviris.

References

[Bengtson and Roth2008] Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 294–303.

[Bulgarov and Caragea2015] Florin Adrian Bulgarov and Cornelia Caragea. 2015. A comparison of supervised keyphrase extraction models. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 13–14.

[Caragea et al.2014] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1435–1446, Doha, Qatar, October. Association for Computational Linguistics.

[Chen and Guestrin2016] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

[D'Avanzo et al.2004] Ernesto D'Avanzo, Bernardo Magnini, and Alessandro Vallin. 2004. Keyphrase extraction for summarization purposes: The LAKE system at DUC-2004. In *Proceedings of the 2004 DUC*.

[du Plessis et al.2014] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. 2014. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 703–711.

[Elkan and Noto2008] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 213–220.

[Frank et al.1999] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-manning. 1999. Domain specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on AI*, pages 668–673.

[Hasan and Ng2010] Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of the 23rd COLING*, COLING 2010, pages 365–373, Stroudsburg, PA, USA.

[Hulth2003] Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

[Kim and Kan2009] Su Nam Kim and Min-Yen Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications*, pages 9–16. Association for Computational Linguistics.

[Kim et al.2012] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2012. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*, 47(3):723–742, December.

[Marujo et al.2012] Luis Marujo, Anatole Gershman, Jaime Carbonell, Robert Frederking, and Jo ao P. Neto. 2012. Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization. In *Proceedings of LREC 2012*. ELRA.

[Nguyen and Kan2007] Thuy Dung Nguyen and Min-Yen Kan. 2007. Key phrase extraction in scientific publications. In *Proceeding of International Conference on Asian Digital Libraries*, pages 317–326.

[Ren et al.2014] Yafeng Ren, Donghong Ji, and Hongbin Zhang. 2014. Positive unlabeled learning for deceptive reviews detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 488–498.

[Sterckx et al.2015] Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2015. Topical word importance for fast keyphrase extraction. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 121–122. International World Wide Web Conferences Steering Committee.

[Wan and Xiao2008] Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI 2008, pages 855–860.

[Yih et al.2006] Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 213–222, New York, NY, USA. ACM.

Learning to Answer Questions from Wikipedia Infoboxes

Alvaro Morales
CSAIL MIT
alvarom@mit.edu

Varot Premtoon
CSAIL MIT
varot@mit.edu

Cordelia Avery
CSAIL MIT
cavery@mit.edu

Sue Felshin
CSAIL MIT
sfelshin@mit.edu

Boris Katz
CSAIL MIT
boris@mit.edu

Abstract

A natural language interface to answers on the Web can help us access information more efficiently. We start with an interesting source of information—infoboxes in Wikipedia that summarize factoid knowledge—and develop a comprehensive approach to answering questions with high precision. We first build a system to access data in infoboxes in a structured manner. We use our system to construct a crowdsourced dataset of over 15,000 high-quality, diverse questions. With these questions, we train a convolutional neural network model that outperforms models that achieve top results in similar answer selection tasks.

1 Introduction

The goal of open-domain question answering is to provide high-precision access to information. With many sources of knowledge on the Web, selecting the right answer to a user's question remains challenging. Wikipedia contains over five million articles in its English version. Providing a natural language interface to answers in Wikipedia is an important step towards more effective information access.

Many Wikipedia articles have an *infobox*, a table that summarizes key information in the article in the form of attribute–value pairs like "Narrated by: Fred Astaire". This data source is appealing for question answering because it covers a broad range of facts that are inherently relevant: a human editor manually highlighted this information in the infobox.

Although many infoboxes appear to be similar, they are only semi-structured—few attributes have consistent value types across articles, infobox templates do not mandate which attributes must be included, and editors are allowed to add article-specific attributes. Infobox-like tables are very common on the Web. Since it is infeasible to incorporate every such source into structured knowledge bases like Freebase (Bollacker et al., 2008), we need techniques that do not rely on ontology or value type information.

Q	Who took over after Nelson Mandela?
A	Succeeded by: Thabo Mbeki
Q	Who designed Central Park?
A	Architect: Frederick Law Olmsted
Q	Where did Oscar Wilde earn his degree?
A	Alma mater: Trinity College, Dublin
Q	What does Intel do?
A	Industry: Semiconductors

Table 1: Example questions and answers with little lexical overlap from the INFOBOXQA dataset.

We focus on the answer selection problem, where the goal is to select the best answer out of a given candidate set of attribute–value pairs from infoboxes corresponding to a named entity in the question. Table 1 illustrates how questions from users may have little lexical overlap with the correct attribute–value pair. Answer selection is an important subtask in building an end-to-end question answering system.

Our work has two main contributions: (1) We compiled the INFOBOXQA dataset, a crowdsourced corpus of over 15,000 questions with answers from

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1930–1935,
Austin, Texas, November 1-5, 2016.

infoboxes in 150 articles in Wikipedia. Unlike existing answer selection datasets with answers from knowledge bases or long-form text, INFOBOXQA targets tabular data that is not augmented with value types or linked to an ontology. (2) We built a multichannel convolutional neural network (CNN) model that achieves the best results on INFOBOXQA compared to other neural network models in the answer selection task.

2 The INFOBOXQA dataset

Infoboxes are designed to be human-readable, not machine-interpretable. This allowed us to devise a crowdsourced assignment where we ask participants to generate questions from infoboxes. With little to no training, humans can form coherent questions out of terse, potentially ambiguous attribute–value pairs.

Wikipedia does not provide a way to access specific information segments; its API returns the entire article. We first worked on the data access problem and developed a system called WikipediaBase to robustly extract attribute–value pairs from infoboxes. Inspired by Omnibase (Katz et al., 2002), we organize infoboxes in an *object–attribute–value* data model, where an object (Lake Titicaca) has an attribute ("Surface area") with a value (8,372 km^2). Attributes are grouped by infobox class (for instance, the `film` class contains attributes like "Directed by" and "Cinematography"). The data model allowed us to extend WikipediaBase to information outside of infoboxes. We implemented methods for accessing images, categories, and article sections.

We then created a question-writing assignment where participants see infoboxes constructed using data from WikipediaBase. These infoboxes visually resembled the original ones in Wikipedia but were designed to control for variables. To prevent participants from only generating questions for attributes at the top of the table, the order of attributes was randomly shuffled. To ensure that the task could be completed in a reasonable amount of time, infoboxes were partitioned into assignments with up to ten attributes. A major goal of this data collection was to gather question paraphrases. For each attribute, we asked participants to write two questions. It is likely that at least one of the questions will use words from the attribute, but requiring an additional question encouraged them to think of alternative phrasings.

Total questions	15266
Total attributes	762
Average questions per attribute	20.0
Average answers per question	17.8

Table 2: Statistics of the INFOBOXQA dataset.

Every infobox in the experiment included a picture to help disambiguate the article. For instance, the cover image for "Grand Theft Auto III" (in concert with the values in the infobox) makes it reasonably clear that the assignment is about a video game and not a type of crime. We asked participants to include an explicit referent to the article title in each question (e.g., "Where was Albert Einstein born?" instead of "Where was he born?").

We analyzed the occurrences of infobox attributes in Wikipedia and found that they fit a rapidly-decaying exponential distribution with a long tail of attributes that occur in few articles. This distribution means that with a carefully chosen subset of articles we can achieve a large coverage of frequently appearing attributes. We developed a greedy approximation algorithm that selects a subset of infobox classes, picks a random sample of articles in the class, and chooses three representative articles that contain the largest quantity of attributes. 150 articles from 50 classes were selected, covering roughly half of common attributes found in Wikipedia.

The dataset contains example questions q_i, with an attribute–value pair (a_i, v_i) that answers the question. To generate negative examples for the answer selection task, we picked every other tuple $(a_j, v_j); \forall j \neq i$ from the infobox that contains the correct answer. If we know that a question asks about a specific entity, we must consider every attribute in the entity's infobox as a possible answer. In INFOBOXQA, candidate answers are just attribute–value pairs with no type information. Because of this, every attribute in the infobox is indistinguishable a priori, and is thus in the candidate set. Not having type information makes the task harder but also more realistic. Table 2 shows statistics of INFOBOXQA. The dataset is available online.[1]

[1] http://groups.csail.mit.edu/infolab/infoboxqa/

3 Model description

Deep learning models for answer selection assume that there is a high similarity between question and answer representations (Yu et al., 2014). Instead of comparing them directly, the main intuition in our model is to use the *attribute* as an explicit bridge to facilitate the match between question and answer. Consider the question "Who replaced Dwight D. Eisenhower?", with answer "Succeeded by: John F. Kennedy". Clearly, the attribute "Succeeded by" plays a crucial role in indicating the match between the question and the answer. If the question and attribute have certain semantic similarities, and those similarities match the similarities of the answer and the attribute, then the answer must be a good match for the question.

We propose an architecture with three weight-sharing CNNs, each one processing either the question, the attribute, or the answer. We then use an element-wise product merge layer to compute similarities between the question and attribute, and between the attribute and answer. We refer to this model as Tri-CNN. Tri-CNN has five types of layers: an input layer, a convolution layer, a max-pooling layer, a merge layer, and a final multi-layer perceptron (MLP) scoring layer that solves the answer selection task. We now describe each layer.

Input layer. Let s_q be a matrix $\in \mathbb{R}^{|s_q| \times d}$, where row i is a d-dimensional word embedding of the i-th word in the question. Similarly, let s_{attr} and s_{ans} be word embedding matrices for the attribute and answer, respectively. s_q, s_{attr}, and s_{ans} are zero-padded to have the same length. We use pre-trained GloVe[2] embeddings with $d = 300$ (Pennington et al., 2014), which we keep adaptable during training.

Convolution layer. We use the multi-channel CNN architecture of (Kim, 2014) with three weight-sharing CNNs, one each for s_q, s_{attr}, and s_{ans}. Different lengths of token substrings (e.g., unigrams or bigrams) are used as channels. The CNNs share weights among the three inputs in a Siamese architecture (Bromley et al., 1993). Weight-sharing allows the model to compute the representation of one input influenced by the other inputs; i.e., the representation of the question is influenced by the representations of the attribute and answer.

[2]http://nlp.stanford.edu/projects/glove/

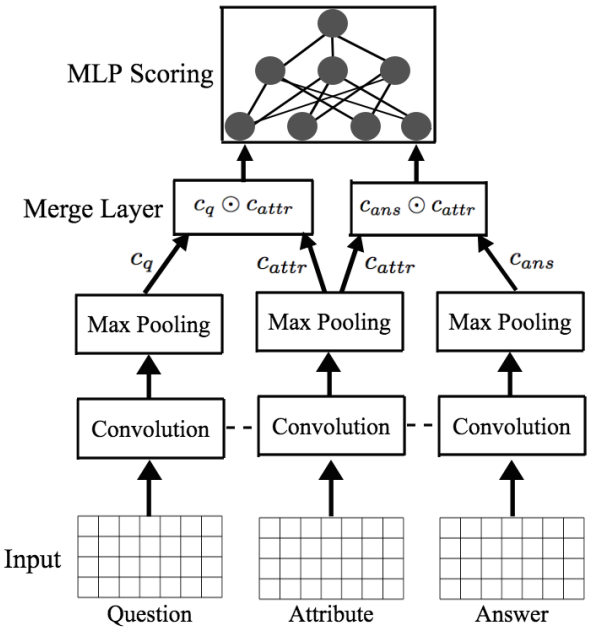


Figure 1: A schematic of the Tri-CNN model.

We describe the convolution layer with respect to the input s, which can stand for s_q, s_{attr}, or s_{ans}. For each channel $h \in [1...M]$, a filter $w \in \mathbb{R}^{h \times d}$ is applied to a sliding window of h rows of s to produce a feature map C. Formally, C is a matrix where:

$$C[i,:] = \tanh(w \cdot s[i...i+h-1,:] + b) \quad (1)$$

and $b \in \mathbb{R}^d$ is the bias. We use *wide convolution* to ensure that terminal and non-terminal words are considered equally when applying the filter w (Blunsom et al., 2014).

Max-pooling layer. Pooling is used to extract meaningful features from the output of convolution (Yin et al., 2015). We apply a max-pooling layer to the output of each channel h. The result is a vector $c_h \in \mathbb{R}^d$ where

$$c_h[i] = \max\{C[:,i]\} \quad (2)$$

Max-pooling is applied to all M channels. The resulting vectors c_h for $h \in [1...M]$ are concatenated into a single vector c.

Merge layer. Our goal is to model the semantic similarities between the question and the attribute, and between the answer and the attribute. We compute the element-wise product of the feature vectors

generated by convolution and max-pooling as follows:

$$d_{q,attr} = c_q \odot c_{attr} \quad (3)$$

$$d_{ans,attr} = c_{ans} \odot c_{attr} \quad (4)$$

where $\odot$ is the element-wise product operator, such that d_{ij} is a vector. Each element in d_{ij} encodes a similarity in a single semantic aspect between two feature representations.

MLP scoring layer. We wish to compute a real-valued similarity between the distance vectors from the merge layer. Instead of directly computing this using, e.g., cosine similarity, we follow (Baudiš and Šedivỳ, 2016) and first project the two distance vectors into a shared embedding space. We compute element-wise sums and products of the embeddings, which are then input to a two-layer perceptron.

4 Experiments

We implemented Tri-CNN in the `dataset-sts`[3] framework for semantic text similarity, built on top of the Keras deep learning library (Chollet, 2015). The framework aims to unify various sentence matching tasks, including answer selection, and provides implementations for variants of sentence-matching models that achieve state-of-the-art results on the TREC answer selection dataset (Wang et al., 2007). We evaluated the performance of various models in `dataset-sts` against INFOBOXQA for the task of answer selection. We report the average and the standard deviation for mean average precision (MAP) and mean reciprocal rank (MRR) from five-fold cross validation. We used 10% of the training set for validation.

In answer selection, a model learns a function to score candidate answers; the set of candidate answers is already given. Entity linking is needed to generate candidate answers and is often treated as a separate module. For INFOBOXQA, we asked humans to generate questions from pre-specified infoboxes. Given this setup, we already know which entity the question refers to; we also know that the question is answerable by the infobox. Entity linking was therefore out of scope in our experiments. By effectively asking humans to identify the named entity, our evaluation results are not affected by noise caused by a faulty entity linking strategy.

[3] https://github.com/brmson/dataset-sts

Model	MAP		MRR	
	Avg	*SD*	*Avg*	*SD*
TF-IDF	0.503	0.004	0.501	0.065
BM-25	0.531	0.007	0.532	0.056
AVG	0.593	0.021	0.609	0.042
RNN	0.685	0.024	0.674	0.028
ATTN1511	0.772	0.016	0.771	0.014
CNN	0.757	0.015	0.754	0.024
Tri-CNN	**0.806**	**0.014**	**0.781**	**0.025**

Table 3: Results of five-fold cross validation. Our Tri-CNN model achieves the best results in MAP and MRR.

4.1 Benchmarks

We compare against TF-IDF and BM25 (Robertson et al., 1995), two models from the information retrieval literature that calculate weighted measures of word co-occurrence between the question and answer. We also experiment with various neural network sentence matching models. AVG is a baseline model that computes averages of unigram word embeddings. CNN is the model most similar to Tri-CNN, with two CNNs in a Siamese architecture, one for the question and one for the answer. Max-pooling is computed on the output of convolution, and then fed to the output layer directly. RNN computes summary embeddings of the question and answer using bidirectional GRUs (Cho et al., 2014). ATTN1511 feeds the outputs of the bi-GRU into the convolution layer. It implements an asymmetric attention mechanism as in (Tan et al., 2015), where the output of convolution and max-pooling of the question is used to re-weight the input to convolution of the answer. The convolution weights are not shared. For these neural architectures, we use the same MLP scoring layer used in Tri-CNN as the output layer and train using bipartite RankNet loss (Burges et al., 2005).

4.2 Results

Table 3 summarizes the results of experiments on INFOBOXQA. The performance of the baselines indicates that unigram bag-of-words models are not sufficiently expressive for matching; Tri-CNN makes use of larger semantic units through its multiple channels. The attention mechanism and the combination of an RNN and CNN in ATTN1511 achieves better results than RNN, but still performs

slightly worse than the CNN model with weight-sharing. The Siamese architecture allows an input's representation to be influenced by the other inputs. The convolution feature maps are thus encoded in a comparable scheme that is more amenable to a matching task. Our Tri-CNN model built on top of this weight-sharing architecture achieves the best performance. Tri-CNN computes the match by comparing the similarities between question–attribute and answer–attribute, which leads to improved results over models that compare the question and answer directly.

5 Related work

Deep learning approaches to answer selection have been successful on the standard TREC dataset and the more recent WIKIQA corpus (Yang et al., 2015). Models like (Feng et al., 2015) and (Wang and Nyberg, 2015) generate feature representations of questions and answers using neural networks, computing the similarity of these representations to select an answer. Recently, attention mechanisms to influence the calculation of the representation (Tan et al., 2015) or to re-weight feature maps before matching (Santos et al., 2016) have achieved good results. Our work differs from past approaches in that we use the attribute as an additional input to the matching task. Other approaches to question answering over structured knowledge bases focus on mapping questions into executable database queries (Berant et al., 2013) or traversing embedded sub-graphs in vector space (Bordes et al., 2014).

6 Conclusion

We presented an approach to answering questions from infoboxes in Wikipedia. We first compiled the INFOBOXQA dataset, a large and varied corpus of interesting questions from infoboxes. We then trained a convolutional neural network model on this dataset that uses the infobox attribute as a bridge in matching the question to the answer. Our Tri-CNN model achieved the best results when compared to recent CNN and RNN-based architectures. We plan to test our model's ability to generalize to other types of infobox-like tables on the Web. We expect our methods to achieve good results for sources such as product descriptions on shopping websites.

Acknowledgments

We thank Andrei Barbu and Yevgeni Berzak for helpful discussions and insightful comments on this paper. We also thank Ayesha Bose, Michael Silver, and the anonymous reviewers for their valuable feedback. Federico Mora, Kevin Ellis, Chris Perivolaropoulos, Cheuk Hang Lee, Michael Silver, and Mengjiao Yang also made contributions to the early iterations and current version of WikipediaBase. The work described in this paper has been supported in part by AFRL contract No. FA8750-15-C-0010 and in part through funding provided by the Center for Brains, Minds, and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

Petr Baudiš and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96. ACM.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *arXiv preprint arXiv:1508.01585*.

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. 2002. Omnibase: Uniform access to heterogeneous data for question answering. In *Natural Language Processing and Information Systems*, pages 230–234. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 13.

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

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. *NIST Special Publication SP*, 109:109.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*, volume 7, pages 22–32.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Citeseer.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*, December.

Timeline extraction using distant supervision and joint inference

Savelie Cornegruta
Department of Biomedical Engineering
King's College London, UK
savelie.cornegruta@kcl.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

In timeline extraction the goal is to order all the events in which a target entity is involved in a timeline. Due to the lack of explicitly annotated data, previous work is primarily rule-based and uses pre-trained temporal linking systems. In this work, we propose a distantly supervised approach by heuristically aligning timelines with documents. The noisy training data created allows us to learn models that anchor events to temporal expressions and entities; during testing, the predictions of these models are combined to produce the timeline. Furthermore, we show how to improve performance using joint inference. In experiments in the SemEval-2015 TimeLine task we show that our distantly supervised approach matches the state-of-the-art performance while joint inference further improves on it by 3.2 F-score points.

1 Introduction

Temporal information extraction focuses on extracting relations and events along with the time when they were true or happened. In this work we focus on timeline extraction, following the recent SemEval TimeLine shared task (Minard et al., 2015). The aim of the task is to extract timelines from multiple documents consisting of events in which a given target entity is the main participant. An example timeline for the entity *Steve Jobs* extracted from 4 documents is given in Fig.1.

The development data provided by the TimeLine shared task does not contain annotations for the various intermediate processing stages needed, only a set of documents with annotated event mentions (input) and the timelines extracted for a few target entities (output). No training data was provided, thus participating systems used rules combined with temporal linking systems trained on related tasks in order to anchor events to temporal expressions and entities to construct the timelines.

We propose a new approach to timeline extraction that uses the development data provided as distant supervision to generate noisy training data (Craven and Kumlien, 1999; Mintz et al., 2009). More specifically, we heuristically align the target entity and the timestamps from the timelines with automatically recognized entities and temporal expressions in the documents. This noisy labeled data set allows us to learn models for the subtasks of anchoring events to temporal expressions and to entities, without requiring training models on additional data. Also, we improve the performance using joint inference for both anchoring subtasks. In our experiments, we show that our distantly supervised approach matches the state-of-the-art performance while joint inference further improves on it by 3.2 F-score points. Our code is publicly available at `http://github.com/savac/timeline`.

2 Timeline extraction

The task of timeline extraction given a target entity and a set of documents can be decomposed as follows. The initial stages are event mention extraction, target entity recognition, and temporal expression identification and resolution. The next stages are anchoring event mentions to target entities and temporal expressions. The final stages are event corefer-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1936–1942,
Austin, Texas, November 1-5, 2016.

Documents:

DocId: 16844, DCT: 2010-06-08, Sentence: 2,3,4,5; *Yesterday*[2010-06-07] *, at this year 's Apple Worldwide Developers Conference (WWDC) , company CEO Steve Jobs*[Steve Jobs] *unveiled iPhone 4*[iPhone 4] *, along with the new iOS 4 operating system for Apple mobile devices . The announcement was long-awaited but not a very big surprise . In April , the technology blog Gizmodo obtained a prototype of the new phone*[iPhone 4] *and published details of it*[iPhone 4] *online . While introducing iPhone 4*[iPhone 4] *, at the annual conference , Jobs*[Steve Jobs] *started by hinting at the incident , saying , " Stop me if you 've already seen this .*

DocId: 17036, DCT: 2010-07-17, Sentence: 6,15; *Rather than recall the devices or offer a hardware fix , Jobs*[Steve Jobs] *said yesterday*[2010-07-16] *that Apple will offer a free case to anyone who has purchased an iPhone 4*[iPhone 4] *. [...] However Jobs*[Steve Jobs] *admitted that the percentage of calls dropped on the iPhone 4*[iPhone 4] *was slightly greater than the percentage of calls dropped on the 3GS*[iPhone 3GS] *.*

DocId: 16900, DCT: 2010-06-16, Sentence: 6; *The newest iPhone*[iPhone 4] *, iPhone 4*[iPhone 4] *was introduced by Apple CEO*[Steve Jobs] *Steve Jobs*[Steve Jobs] *at the company 's 2010 Worldwide Developer 's Conference less than two weeks ago*[2010-06] *.*

DocId 16983, 2010-10-23, Sentence 10; *In his*[Steve Jobs] *keynote address Wednesday*[2010-10-20] *, Jobs*[Steve Jobs] *announced the release of Apple 's iLife '11 software suite , which includes the iPhoto , iMovie , and GarageBand programs .*

Timeline: *Steve Jobs*

1	2010-06-07	16844-2-unveiled	
1	2010-06-07	16844-5-introducing	16900-11-introduced
1	2010-06-07	16844-5-hinting	
1	2010-06-07	16844-5-saying	
2	2010-07-16	17036-6-said	
2	2010-07-16	17036-15-admitted	
3	2010-10-20	16983-10-address	
3	2010-10-20	16983-10-announced	

Figure 1: Example timeline for target entity *Steve Jobs*. The input to the system is the documents annotated with event mentions annotations and their Document Creation Time (DCT). The event mentions appearing in the timeline are identified by their document id-sentence index. The annotations for the target entities and temporal expression mentions need to be done by the system.

ence resolution and ordering of the events in a timeline, which rely largely on their anchoring to temporal expressions. The TimeLine shared task had two tracks, A and B, the only difference being that in Track B the event mentions are provided in the input. We consider this track in this paper and focus on learning the anchoring of events to temporal expressions and entities.

The development data provided in the context of the shared task consisted of documents related to *Apple* and gold timelines for six target entities. Evaluation was performed by extracting timelines from three document sets, each related to *Airbus*, *GM* and *Stock market* respectively. We used the official evaluation which is based on the metric introduced by UzZaman and Allen (2011) which assesses a predicted timeline versus the gold standard one using precision, recall and F-score over binary temporal relations between the events.

3 Distant supervision

In order to generate training data for anchoring event mentions to target entities and temporal expressions via distant supervision, we first need to identify them. For entity recognition we use approximate string matching combined with the Stanford Coreference Resolution System (Lee et al., 2013). For temporal expression identification and resolution to absolute timestamps we use the UWTime temporal parser (Lee et al., 2014).

Next we generate labeled instances as follows. For anchoring events to entities, we consider for each event mention the correct entity mention to be the nearest mention of the target entity in the same sentence, and all others to be incorrect. Similarly, for anchoring events to timestamps, we consider for each event mention the correct temporal expression to be the nearest temporal expression that exactly matches the timestamp according to the timeline (but not necessarily in the same sentence), and all others to be incorrect. The datasets generated will be noisy since correct anchors may be entity mentions and temporal expressions that are not the nearest ones. Further noise is expected due to errors in the entity recognition and temporal expression identification and resolution stages.

Features	type
Measure distance in tokens between event and target entity mentions	local
Syntactic dependencies between event and target entity mentions (extracted from training corpus)	local
Check if subsequent events have the same stem and are attributed to the same target entity	global
Check if subsequent events are in the same sentence and are attributed to the same target entity	global
Check if subsequent events are both communication events and are attributed to the same target entity	global

Table 1: Features to encode dependencies between events and target entities

Features	type
Measure distance in sentences between event mention and temporal expression	local
Measure distance in tokens between event mention and temporal expression	local
Syntactic dependencies between event mention and temporal expression (extracted from training corpus)	local
Check if temporal expression is before of after the event mention	local
Check if timestamp is in the future wrt the DCT	local
Check if timestamp is undefined (i.e. XX-XX-XXXX)	local
Check if timestamp is incomplete	local
Check if subsequent events and are linked to the same temporal expression	global
Check if subsequent events have the same stem and are linked to the same temporal expression	global
Check if subsequent events are in the same sentence and are linked to the same temporal expression	global
Check if subsequent events are communication events and are linked to the same temporal expression	global

Table 2: Features to encode dependencies between events and temporal expressions

4 Event anchoring

After generating training data for anchoring event mentions to target entities and to temporal expressions with distant supervision, we now proceed to developing linear models for each of these tasks.

4.1 Classification

Using distant supervision we obtained examples of correct and incorrect anchoring of event mentions to entities and temporal expressions. Thus we learn for each of the two tasks a binary linear classifier of the form:

$$score(x, y, \mathbf{w}) = \mathbf{w} \cdot \phi(x, y) \quad (1)$$

where x is an event mention, y is the anchor (either the target entity or the temporal expression) and $\mathbf{w}$ are the parameters to be learned. The features extracted by ϕ represent various distance measures and syntactic dependencies between the event mention and the anchor obtained using Stanford CoreNLP (Manning et al., 2014). The temporal expression anchoring model also uses a few feature templates that depend on the timestamp of the temporal expression. The full list of features extracted by ϕ are denoted as local in Tables 1 and 2.

4.2 Alignment

The classification approach described is limited to anchoring each event mention to an entity or a temporal expression in isolation. However it would be preferable to infer the decisions for each task jointly at the document level and take into account the dependencies in anchoring different events, e.g. that consecutive events in text are likely to be anchored to the same entity, as shown in Figure 2, or to the same temporal expression. Capturing such dependencies can be crucial when the correct anchor is not explicitly signalled in the text but can be inferred considering other relations and/or their ordering in text (Derczynski, 2013).

Defining our joint model formally, let $\mathbf{x}$ be a vector containing all event mentions in a document and $\mathbf{y}$ be the vector of all anchors (target entity mentions or temporal expressions) in the same document. The order of the events in $\mathbf{x}$ is as they appear in the document. Let $\mathbf{z}$ be a vector of the same length as $\mathbf{x}$ that defines the alignment between $\mathbf{x}$ and $\mathbf{y}$ by containing pointers to elements in $\mathbf{y}$, thus allowing for multiple events to share the same anchor. The scoring function is defined as

$$score(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}) = \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad (2)$$

where the global feature function Φ, in addition to the features returned by the local scoring function (Eq. 1), also returns features taking into account anchoring predictions across the document. Apart from features encoding subsequences of anchoring

Documents:

DocId: 17036, DCT: 2010-07-17, Sentence: 9,10,11:

Jobs[Steve Jobs] *also said that those who had already purchased a bumper will receive a full refund for the accessory. For consumers still dissatisfied with iPhone 4*[iPhone 4]*, Jobs*[Steve Jobs] *said that the phones can be returned for a refund as well. Jobs*[Steve Jobs] *acknowledged that "a very small percentage of users" were experiencing antenna issues, but dismissed the existence of an "Antennagate," saying that similar problems plague all cellular phones and that the iPhone*[iPhone 4] *issue "has been blown so out of proportion that it is incredible."*

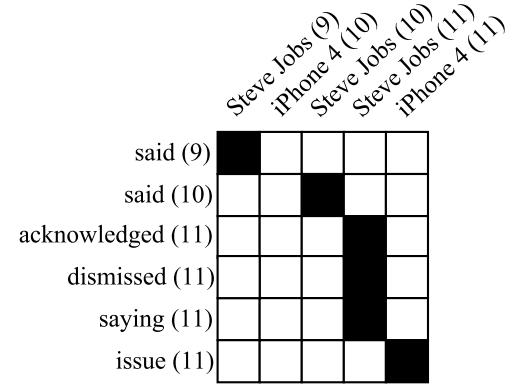


Figure 2: The correct alignment of events and target entity mentions is shown with the numbers in brackets denoting the index of the sentence in which the mention is found. The consecutive events acknowledged, dismissed and saying are anchored to entity Steve Jobs that was only mentioned once in the beginning of the sentence.

predictions, it also makes possible to make them dependent on the events, e.g. a binary indicator encoding whether two consecutive events with the same stem share the same anchor or not. The full list of local and global features extracted by Φ are presented in Tables 1 and 2. Predicting with the scoring function in Eq.2 amounts to finding the anchoring sequence vector $\mathbf{z}$ that maximizes it. To be able to perform exact inference efficiently, we impose a first order Markov assumption and use the Viterbi algorithm (Viterbi, 1967). Similar approaches have been successful in word alignment for machine translation (Blunsom and Cohn, 2006).

4.3 Post-processing

During testing, we need to construct the timeline for each target entity using the events that were predicted to be anchored to it and the timestamps of the temporal expressions each event was anchored to. Thus, we need to perform two additional tasks, event coreference and ordering. For the former we define a simple heuristic by which if two mentions have the same stems and timestamps then they refer to the same event. The only exception is that if two mentions represent communication events (*said*, *announced* etc.), then they are resolved to different events when in the same document. We finally order the events according to their timestamp.

5 Results

We evaluate our system using the setup provided by the TimeLine task ensuring that the training and validation are performed only using the development data i.e. the *Apple* collection. All linear models were trained with the perceptron update rule (Pedregosa et al., 2011). We tuned the number of perceptron iterations by performing cross-validation using the development data by holding out the timeline for one target entity and training on the timelines for the remaining ones.

In Table 3 we compare the binary classification model (Our_System_Binary) against the alignment model (Our_System_Alignment) and show that the latter outperforms the former by a margin of 3.2 points in F-score, achieving a micro F_1-score of 28.58 across the three test corpora, thus confirming the benefits of joint inference. The only corpus in which joint inference did not help was *Stock* which has on average shorter event chains per document (Minard et al., 2015) and thus renders joint anchoring less likely to be useful.

We now compare our approach to the two participants in the TimeLine shared task with two runs each. The best-performing GPLSIUA team (Navarro and Saquete, 2015) used the TIPSem tool developed by Llorens et al. (2010) for temporal relation processing which extracts events and temporal expressions and uses a Conditional Random Field model to anchor them against each other. However, TIPSem only considers anchoring of events to temporal expressions that are in the same sentence. GPLSIUA also used the semantic role labeler from SENNA (Collobert et al., 2011) and OpenNER and anchored entities to events using a rule-based approach. The HeidelToul team (Moulahi et al., 2015) used HeidelTime (Strötgen et al., 2013) to identify and resolve temporal expressions and de-

System	Airbus F_1	GM F_1	Stock F_1	Total P	Total R	Total F_1
GPLSIUA_1	22.35	19.28	33.59	21.73	**30.46**	25.36
GPLSIUA_2	20.47	16.17	29.90	20.08	26.00	22.66
HeidelToul_1	19.62	7.25	20.37	20.11	14.76	17.03
HeidelToul_2	16.50	10.82	25.89	13.58	28.23	18.34
Our_System_Binary	17.99	20.97	**34.95**	25.97	24.79	25.37
Our_System_Alignment	**25.65**	**26.64**	32.35	**29.05**	28.12	**28.58**

Table 3: Results for our system and other participants in the SemEval 2015 Task 4: TimeLine.

veloped a target entity mention identification tool similar to ours using Stanford CoreNLP (Manning et al., 2014). However, they rely on a rule-based approach for event anchoring. Our binary model matches the performance of the best system, and our alignment model exceeds it by 3.2 F_1-score points across, even though we do not use any off-the-shelf components developed for temporal relation extraction. Instead we rely on training data generated with distant supervision, and UWTime for temporal expression identification and resolution, for which the participants also used similar components.

6 Related work

In recent work, Laparra et al. (2015) also considered anchoring at the document-level in the context of the Track A of the TimeLine shared task, however they developed a rule-based approach. The structure features used in our joint inference approach encode similar intuitions, but we are learning model weights using distant supervision so that we can combine them more flexibly. And even though the noise in the trainng data generated with distant supervision is a concern, manual annotation of temporal relations is known to have low inter-annotator agreement rates[1] and thus also likely to be noisy.

Prior to the TimeLine shared task, TempEval (Verhagen et al., 2007) was the original task that focused on categorising the relations between events, temporal expressions and Document Creation Time using the the TimeML annotation language. The task classified only the relations between mentions in the same or consecutive sentences. The two following tasks, TempEval-2 (Verhagen et al., 2010) and TempEval-3 (UzZaman et al., 2013), added tasks for event and temporal expression identification as well as an end-to-end temporal relation processing task that was performed on raw text.

Beyond TempEval, McClosky and Manning (2012) used distant supervision in order to learn how to extract the temporal bounds for events in the context of the TAC temporal knowledge base population task (Ji et al., 2011). However they focus on learning real-world event ordering constraints (e.g. people go to school before university) instead of how events are reported in text.

7 Conclusions

In this paper we proposed a timeline extraction approach in which we generate noisy training data for anchoring events to entities and temporal expressions using distant supervision. By learning a binary classifier we match the state-of-the-art F_1-score for the Track B of the TimeLine shared task. We further improve this result by 3.2 F_1-score points using joint inference.

Acknowledgments

Part of this work was conducted while both authors were at University College London. The authors would like to thank Leon Derczynski for his feedback on an earlier version. Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139).

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 65–72. Association for Computational Linguistics.

[1] http://www.timeml.org/timebank/documentation-1.2.html

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press.

Leon Derczynski. 2013. *Determining the Types of Temporal Relations in Discourse*. Ph.D. thesis, University of Sheffield.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of Text Analysis Conference (TAC)*.

Egoitz Laparra, Itziar Aldabe, and German Rigau. 2015. Document level time-anchoring for timeline extraction. In *ACL*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Comput. Linguist.*, 39(4):885–916, December.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Baltimore, Maryland, June. Association for Computational Linguistics.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

David McClosky and Christopher D Manning. 2012. Learning constraints for consistent timeline extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 873–882. Association for Computational Linguistics.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. Semeval-2015 task 4: Timeline: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado, June. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bilel Moulahi, Jannik Strötgen, Michael Gertz, and Lynda Tamine. 2015. Heideltoul: A baseline approach for cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 825–829, Denver, Colorado, June. Association for Computational Linguistics.

Borja Navarro and Estela Saquete. 2015. Gplsiua: Combining temporal information and topic modeling for cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 820–824, Denver, Colorado, June. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jannik Strötgen, Julian Zell, and Michael Gertz. 2013. Heideltime: Tuning english and developing spanish resources for tempeval-3. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 15–19, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Naushad UzZaman and James F. Allen. 2011. Temporal evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 351–356, Stroudsburg, PA, USA. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia,

USA, June. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April.

Combining Supervised and Unsupervised Ensembles for Knowledge Base Population

Nazneen Fatema Rajani and **Raymond J. Mooney**
Department Of Computer Science
The University of Texas at Austin
nrajani@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

We propose an algorithm that combines supervised and unsupervised methods to ensemble multiple systems for two popular Knowledge Base Population (KBP) tasks, Cold Start Slot Filling (CSSF) and Tri-lingual Entity Discovery and Linking (TEDL). We demonstrate that it outperforms the best system for both tasks in the 2015 competition, several ensembling baselines, as well as a state-of-the-art stacking approach. The success of our technique on two different and challenging problems demonstrates the power and generality of our combined approach to ensembling.

1 Introduction

Ensembling multiple systems is a well known standard approach to improving accuracy in several machine learning applications (Dietterich, 2000). Ensembles have been applied to parsing (Henderson and Brill, 1999), word sense disambiguation (Pedersen, 2000), sentiment analysis (Whitehead and Yaeger, 2010) and information extraction (IE) (Florian et al., 2003; McClosky et al., 2012). Recently, using *stacking* (Wolpert, 1992) to ensemble systems was shown to give state-of-the-art results on slot-filling and entity linking for *Knowledge Base Population* (KBP) (Viswanathan et al., 2015; Rajani and Mooney, 2016). Stacking uses supervised learning to train a meta-classifier to combine multiple system outputs; therefore, it requires historical data on the performance of each system. Rajani and Mooney (2016) use data from the 2014 iteration of the KBP competition for training and then test on the data from the 2015 competition, therefore they can only ensemble the *shared systems* that participated in both years.

However, we would sometimes like to ensemble systems for which we have no historical performance data. For example, due to privacy, some companies may be unwilling to share their performance on arbitrary training sets. Simple methods such as voting permit "unsupervised" ensembling, and several more sophisticated methods have also been developed for this scenario (Wang et al., 2013). However, such methods fail to exploit supervision for those systems for which we *do* have training data. Therefore, we present an approach that utilizes supervised *and* unsupervised ensembling to exploit the advantages of both. We first use unsupervised ensembling to combine systems without training data, and then use stacking to combine this ensembled system with other systems with available training data.

Using this new approach, we demonstrate new state-of-the-art results on two NIST KBP challenge tasks: *Cold Start Slot-Filling* (CSSF)[1] and the *Trilingual Entity Discovery and Linking* (TEDL) (Ji et al., 2015). Our approach outperforms the best system as well as other state-of-the-art ensembling methods on both tasks in the most recent 2015 competition. There is one previous work on ensembling supervised and unsupervised models using graph-based consensus maximization (Gao et al., 2009), however we show that it does not do as well as our stacking method.

[1] http://www.nist.gov/tac/2015/KBP/ColdStart/guidelines.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1943–1948,
Austin, Texas, November 1-5, 2016.

2 Overview of KBP Tasks

2.1 Cold Start Slot Filling (CSSF)

The goal of CSSF is to collect information (fills) about specific attributes (slots) for a set of entities (queries) from a given corpus. The query entities can be a person, organization, or geo-political entity (PER/ORG/GPE). The input is a set of queries along with a text corpus in which to look for information. The output is a set of slot fills for each query. Systems must also provide *provenance* in the form of *docid:startoffset-endoffset*, where *docid* specifies a source document and the offsets demarcate the text in this document supporting the filler. Systems may also provide a confidence score to indicate their certainty in the extracted information.

2.2 Tri-lingual Entity Discovery and Linking (TEDL)

The first step in the TEDL task is to discover all entity mentions in a corpus with English, Spanish and Chinese documents. The entities can be a person, organization or geo-political entity (PER/ORG/GPE) and in 2015 two more entity types were introduced – facility and location (FAC/LOC). The extracted mentions are then linked to an existing English KB (a version of FreeBase) entity via its ID. If there is no KB entry for an entity, systems are expected to cluster all the mentions for that entity using a NIL ID. The output for the task is a set of extracted mentions, each with a string, its provenance in the corpus, and a corresponding KB ID if the system could successfully link the mention, or else a mention cluster with a NIL ID. Systems can also provide a confidence score for each mention.

3 Ensembling Algorithm

Figure 1 illustrates our system which trains a final meta-classifier for combining multiple systems using confidence scores and other auxiliary features depending on the task.

3.1 Supervised Ensembling Approach

For the KBP systems that are common between years, we use the stacking method of Viswanathan et al. (2015) for these shared systems.

The meta-classifier makes a binary decision for each distinct output represented as a *key-value* pair.

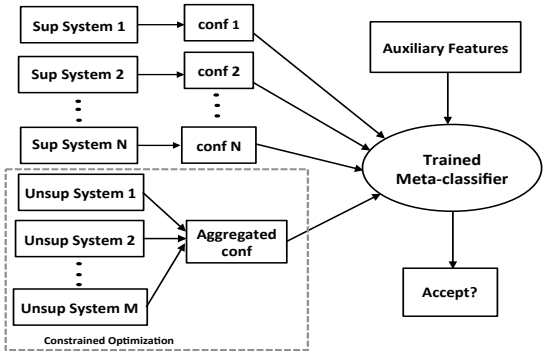


Figure 1: Illustration of our approach to combine supervised and unsupervised ensembles.

The function of the *key* is to provide a handle for aggregating outputs that are common across systems. For the CSSF task, the *key* for ensembling multiple systems is a query along with a slot type, for example, per:age of "Barack Obama" and the *value* is a computed *slot fill*. For TEDL, the key is the *KB (or NIL) ID* and the value is a *mention*, that is a specific reference to an entity in the text. The top half of Figure 1 illustrates ensembling multiple systems with historical training data using a supervised approach.

3.2 Unsupervised Ensembling Approach

Only 38 of the 70 systems that participated in CSSF 2015 also participated in 2014, and only 24 of the 34 systems that participated in TEDL 2015 also participated in 2014 EDL. Therefore, many KBP systems in 2015 were new and did not have past training data. In fact, some of the new systems performed better than the shared systems, for example the *hltcoe* system did not participate in 2014 but was ranked 4^{th} in the 2015 TEDL task (Ji et al., 2015). Thus, for improving *recall* and performance in general, it is crucial to use systems without historical training data, which we call unsupervised systems. To achieve this end, we first ensemble such systems using an unsupervised technique. Frequently, the confidence scores provided by systems are not well-calibrated probabilities. So in order to calibrate the confidence scores across unsupervised systems, we use the constrained optimization approach proposed by Wang et al. (2013). The idea is to aggregate the raw confidence values produced by individual KBP systems,

to arrive at a single aggregated confidence value for each *key-value* pair. The constraints ensure that the aggregated confidence score is close to the raw score as well as proportional to the agreement among systems on a value for a given key. Thus for a given key, if a system's value is also produced by multiple other systems, it would have a higher score than if it were not produced by any other system. The authors use the inverse ranking of the average precision previously achieved by individual systems as the weights in their algorithm. However since we use this approach for systems with no historical performance data, we use uniform weights across all unsupervised systems for both the tasks.

We use the slot type for the CSSF task and entity type for the TEDL task to define the constraints on the values. The output from the constrained optimization approach for both tasks is a set of key-values with aggregated confidence scores across all unsupervised systems which go directly into the stacker as shown in the lower half of Figure 1. Using the aggregation approach as opposed to directly using the raw confidence scores allows the classifier to meaningfully compare confidence scores across multiple systems although they are produced by very diverse systems.

Another unsupervised ensembling method we tried in place of the constrained optimization approach is the Bipartite Graph based Consensus Maximization (BGCM) approach of Gao et al. (2009). BGCM is presented as a way of combining supervised and unsupervised models, so we compare it to our stacking approach to combining supervised and unsupervised systems, as well as an alternative approach to ensembling *just* the unsupervised systems before passing their output to the stacker. BGCM performs an optimization over a bipartite graph of systems and outputs, where the objective function favors the smoothness of the label assignments over the graph, as well as penalizing deviations from the initial labeling provided by supervised models.

3.3 Combining Supervised and Unsupervised

We propose a novel approach to combine the aforementioned supervised and unsupervised methods using a stacked meta-classifier as the final arbiter for accepting a given key-value. The outputs from the supervised and unsupervised systems are fed into the stacker in a consistent format such that there is a unique input *key-value* pair. Most KBP teams submit multiple variations of their system. Before ensembling, we first combine multiple runs of the same team into one. Of the 38 CSSF systems from 10 teams for which we have 2014 data for training and the 32 systems from 13 teams that do not have training data, we combine the runs of each team into one to ensure diversity of the final ensemble. For the slot fills that were common between the runs of a given team, we compute an average confidence value, and then add any additional fills that are not common between runs. Thus, we obtained 10 systems (one for each team) for which we have supervised data for training stacking. Similarly, we combine the 24 TEDL systems from 6 teams that have 2014 training data and 10 systems from 4 teams that did not have training data into one per team. Thus using the notation in Figure 1, for TEDL, $N = 6$ and $M = 4$ while for CSSF, $N = 10$ and $M = 13$.

The unsupervised method produces aggregated, calibrated confidence scores which go directly into our final meta-classifier. We treat this combination as a single system called the *unsupervised ensemble*. We add the unsupervised ensemble as an additional system to the stacker, thus giving us a total of $N+1$, that is 11 CSSF and 7 TEDL systems. Once we have extracted the auxiliary features for each of the N supervised systems and the unsupervised ensemble for both years, we train the stacker on 2014 systems, and test on the 2015 systems. The unsupervised ensemble for each year is composed of different systems, but hopefully the stacker learns to combine a generic unsupervised ensemble with the supervised systems that are shared across years. This allows the stacker to arbitrate the final correctness of a key-value pair, combining systems for which we have no historical data with systems for which training data *is* available. To learn the meta-classifier, we use an L1-regularized SVM with a linear kernel (Fan et al., 2008) (other classifiers gave similar results).

3.4 Post-processing

Once we obtain the decisions on each key-value pair from the stacker, we perform some final post-processing. For CSSF, each list-valued slot fill that is classified as correct is included in the final output. For single-valued slot fills, if they are multiple fills

Methodology	Precision	Recall	F1
Combined stacking and constrained optimization with auxiliary features	0.4679	**0.4314**	**0.4489**
Top ranked SFV system in 2015 (Rodriguez et al., 2015)	0.4930	0.3910	0.4361
Stacking using BGCM instead of constrained optimization	**0.5901**	0.3021	0.3996
BGCM for combining supervised and unsupervised systems	0.4902	0.3363	0.3989
Stacking with auxiliary features described in (Rajani and Mooney, 2016)	0.4656	0.3312	0.3871
Ensembling approach described in (Viswanathan et al., 2015)	0.5084	0.2855	0.3657
Top ranked CSSF system in 2015 (Angeli et al., 2015)	0.3989	0.3058	0.3462
Oracle Voting baseline (3 or more systems must agree)	0.4384	0.2720	0.3357
Constrained optimization approach described in (Wang et al., 2013)	0.1712	0.3998	0.2397

Table 1: Results on 2015 Cold Start Slot Filling (CSSF) task using the official NIST scorer

Methodology	Precision	Recall	F1
Combined stacking and constrained optimization	0.686	**0.624**	**0.653**
Stacking using BGCM instead of constrained optimization	0.803	0.525	0.635
BGCM for combining supervised and unsupervised outputs	0.810	0.517	0.631
Stacking with auxiliary features described in (Rajani and Mooney, 2016)	0.813	0.515	0.630
Ensembling approach described in (Viswanathan et al., 2015)	**0.814**	0.508	0.625
Top ranked TEDL system in 2015 (Sil et al., 2015)	0.693	0.547	0.611
Oracle Voting baseline (4 or more systems must agree)	0.514	0.601	0.554
Constrained optimization approach	0.445	0.176	0.252

Table 2: Results on 2015 Tri-lingual Entity Discovery and Linking (TEDL) using official NIST scorer and CEAF metric

that were classified as correct for the same query and slot type, we include the fill with the highest meta-classifier confidence.

For TEDL, for each entity mention link that is classified as correct, if the link is a KB cluster ID then we include it in the final output, but if the link is a NIL cluster ID then we keep it aside until all mention links are processed. Thereafter, we resolve the NIL IDs across systems since NIL ID's for each system are unique. We merge NIL clusters across systems into one if there is at least one common entity mention among them.

4 Experimental Results

All results were obtained using the official NIST scorers after the competitions ended.[2] We compare to the purely supervised approach of Viswanathan et al. (2015) using shared systems between 2014 and 2015, and the constrained optimization approach of Wang et al. (2013) using all 2015 systems. We also compare to BGCM (Gao et al., 2009) in two ways. First, we use BGCM in place of the constrained optimization approach to ensemble unsupervised systems while keeping the rest of our pipeline the same. Secondly, we also compare to combining both supervised and unsupervised systems using BGCM instead of stacking. We also include an *"oracle"* voting ensembling baseline, which varies the threshold on the number of systems that must agree to identify an "oracle" threshold that results in the highest F1 score for 2015. We find that for CSSF a threshold of 3, and for TEDL a threshold of 4, gives us the best F1 score.

Tables 1 and 2 show CSSF and TEDL results. Our full system, which combines supervised and unsupervised ensembling performed the best on both tasks. TAC-KBP also includes the Slot Filler Validation (SFV) task[3] where the goal is to ensemble/filter outputs from multiple slot filling systems. The top ranked system in 2015 (Rodriguez et al., 2015) does substantially better than many of the other ensembling approaches, but it does not do as well as our best performing system. The purely

[2] http://www.nist.gov/tac/2015/KBP/ColdStart/tools.html, https://github.com/wikilinks/neleval

[3] http://www.nist.gov/tac/2015/KBP/SFValidation/index.html

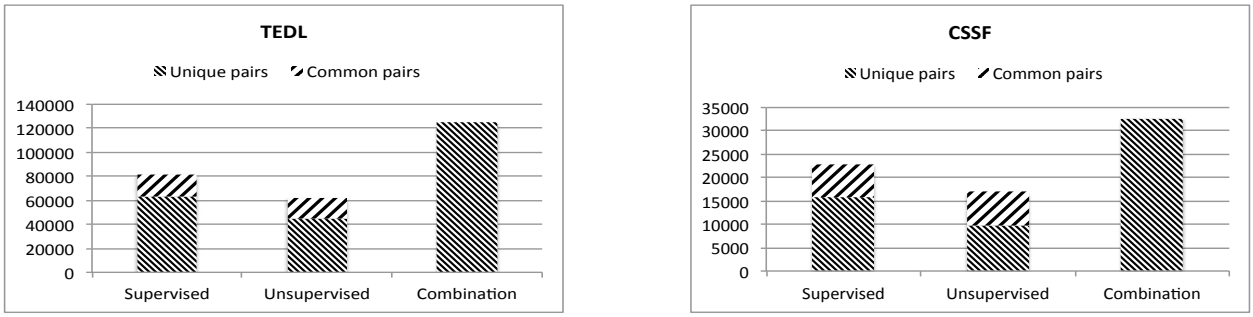


Figure 2: Total number of unique and common input pairs contributed by the supervised and unsupervised systems to the combination for the TEDL and CSSF tasks respectively.

supervised approach of Viswanathan et al. (2015) and the auxiliary features approach of Rajani and Mooney (2016) performs substantially worse, although still outperforming the top-ranked individual system in the 2015 competition. These approaches only use the common systems from 2014, thus ignoring approximately half of the systems. The approach of Wang et al. (2013) performs very poorly by itself; but when combined with stacking gives a boost to recall and thus the overall $F1$. Note that all our combined methods have a substantially higher recall. The oracle voting baseline also performs very poorly indicating that naive ensembling is not advantageous.

TEDL provides three different approaches to measuring accuracy: entity discovery, entity linking, and mention CEAF (Ji et al., 2015). CEAF finds the optimal alignment between system and gold standard clusters, then evaluates precision and recall micro-averaged. We obtained similar results on all three metrics and only include CEAF. The purely supervised stacking approach over shared systems does not do as well as any of our combined approaches even though it beats the best performing system (i.e. IBM) in the 2015 competition (Sil et al., 2015). The relative ranking of the approaches is similar to those obtained for CSSF, proving that our approach is very general and improves performance on two quite different and challenging problems.

Even though it is obvious that the boost in our recall was because of adding the unsupervised systems, it isn't clear how many new *key-value* pairs were generated by these systems. We thus evaluated the contribution of the systems ensembled using the supervised approach and those ensembled using the unsupervised approach, to the final combination for both the tasks. Figure 2 shows the number of unique as well as common *key-value* pairs that were contributed by each of the approaches. The unique pairs are those that were produced by one approach but not the other and the common pairs are those that were produced by both approaches. The number of unique pairs in the combination is the union of unique pairs in the supervised and unsupervised approaches. We found that approximately one third of the input pairs in the combination came from the unique pairs produced just by the unsupervised systems for both the TEDL and CSSF tasks. Only about 15% and 22% of the total input pairs were common between the two approaches for the TEDL and CSSF tasks respectively. Our findings highlight the importance of utilizing systems that do not have historical training data.

5 Conclusion

We presented results on two diverse KBP tasks, showing that a novel stacking-based approach to ensembling both supervised and unsupervised systems is very promising. The approach outperforms the top ranked systems from both 2015 competitions as well as several other ensembling methods, achieving a new state-of-the-art for both of these important, challenging tasks. We found that adding the unsupervised ensemble along with the shared systems specifically increased recall substantially.

Acknowledgment

This research was supported by the DARPA DEFT program under AFRL grant FA8750-13-2-0026.

References

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D. Manning. 2015. Bootstrapped Self Training for Knowledge Base Population. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

T. Dietterich. 2000. Ensemble Methods in Machine Learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. 2009. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Advances in Neural Information Processing Systems (NIPS2009)*, pages 585–593.

John C. Henderson and Eric Brill. 1999. Exploiting Diversity in Natural Language Processing: Combining Parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP99)*, pages 187–194, College Park, MD.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

David McClosky, Sebastian Riedel, Mihai Surdeanu, Andrew McCallum, and Christopher D Manning. 2012. Combining Joint Models for Biomedical Event Extraction. *BMC Bioinformatics*.

Ted Pedersen. 2000. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. In *North American Chapter of the Association for Computational Linguistics (NAACL2000)*, pages 63–69.

Nazneen Fatema Rajani and Raymond J. Mooney. 2016. Stacking With Auxiliary Features. *ArXiv e-prints*.

Miguel Rodriguez, Sean Goldberg, and Daisy Zhe Wang. 2015. University of Florida DSR lab system for KBP slot filler validation 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Avirup Sil, Georgiana Dinu, and Radu Florian. 2015. The IBM systems for trilingual entity discovery and linking at TAC 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond J. Mooney. 2015. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL2015)*, pages 177–187, Beijing, China, July.

I-Jeng Wang, Edwina Liu, Cash Costello, and Christine Piatko. 2013. JHUAPL TAC-KBP2013 Slot Filler Validation System. In *Proceedings of the Sixth Text Analysis Conference (TAC2013)*.

Matthew Whitehead and Larry Yaeger. 2010. Sentiment mining using ensemble classification models. In Tarek Sobh, editor, *Innovations and Advances in Computer Sciences and Engineering*. SPRINGER, Berlin.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Character Sequence Models for Colorful Words

Kazuya Kawakami [♠], **Chris Dyer** [♠♣] **Bryan R. Routledge** [♢] **Noah A. Smith** [♡]
[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[♣]Google DeepMind, London, UK
[♢]Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
{kkawakam,cdyer}@cs.cmu.edu, routledge@cmu.edu, nasmith@cs.washington.edu

Abstract

We present a neural network architecture to predict a point in color space from the sequence of characters in the color's name. Using large scale color–name pairs obtained from an online color design forum, we evaluate our model on a "color Turing test" and find that, given a name, the colors predicted by our model are preferred by annotators to color names created by humans. Our datasets and demo system are available online at http://colorlab.us.

1 Introduction

Color is a valuable vehicle for studying the association between words and their nonlinguistic referents. Perception of color has long been studied in psychology, and quantitative models linking physical stimuli and psychological perception have been in place since the 1920s (Broadbent, 2004). Although perceptually faithful color representations require only a few dimensions (§2), linguistic expressions of color often rely on association and figurative language. There are, for example, 34,000 examples of "blue" in our data. The varieties of blue range can be emotional, descriptive, metaphoric, literal, and whimsical. Consider these examples (best viewed in color): murkey blue, blueberry muffin, greeny blue, and jazzy blue.

This rich variety of descriptive names of colors provides an ideal way to study linguistic creativity, its variation, and an important aspect of visual understanding. This paper uses predictive modeling to explore the relationship between colors (represented in three dimensions) and casual, voluntary linguistic descriptions of them by users of a crafting and design website (§3).[1]

In this dataset's creative vocabulary, word-level representations are so sparse as to be useless, so we turn to models that build name representations out of *characters* (§4). We evaluate our model on a "color Turing test" and find that, given a name, it tends to generate a color that humans find matches the name better than the color that actually inspired the name. We also investigate the reverse mapping, from colors to names (§5). We compare a conditional LSTM language model used in caption generation (Karpathy and Fei-Fei, 2014) to a new latent-variable model, achieving a 10% perplexity reduction.

We expect such modeling to find purchase in computational creativity applications (Veale and Al-Najjar, 2015), design and marketing aids (Deng et al., 2010), and new methods for studying the interface between the human visual and linguistic systems (Marcus, 1991).

2 Color Spaces

In electronic displays and other products, colors are commonly represented in RGB space where each color is embedded in $\{0, \ldots, 255\}^3$, with coordinates corresponding to red, green, and blue levels. While convenient for digital processing, distances in this space are perceptually non-uniform. We instead use a different three-dimensional representation, *Lab*, which was originally designed so that Euclidean distances correlate with human-perceived differences (Hunter, 1958). *Lab* is also continu-

[1] http://www.colourlovers.com

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1949–1954,
Austin, Texas, November 1-5, 2016.

	Number of pairs	Unique names
Train	670,032	476,713
Dev.	53,166	52,753
Test	53,166	52,760
ggplot2	66	66
Paint	956	956

Table 1: Datasets used in this paper. The train/dev./test split of the COLOURlovers data was random. For ggplot2 and Paint, we show the number of test instances which are not in Train set.

Model	Test	ggplot2	Paint
Unigram	1018.35	814.58	351.54
Bigram	977.46	723.61	364.41
RNN	750.26	431.90	305.05
1-layer LSTM	664.11	355.56	303.03
2-layer LSTM	652.49	343.97	274.83

Table 2: MSE in *Lab* space on held-out datasets.

ous, making it more suitable for the gradient-based learning used in this paper. The transformation from RGB to *Lab* is nonlinear.

3 Task and Dataset

We consider the task of predicting a color in *Lab* space given its name. Our dataset is a collection of user-named colors downloaded from COLOURlovers,[1] a creative community where people create and share colors, palettes, and patterns. Our dataset contains 776,364 pairs with 581,483 unique names. Examples of the color/name pairs from COLOURlovers are the following: Sugar Hearts You, Vitamin C, Haunted milk.

We considered two held-out datasets from other sources; these do not overlap with the training data.
ggplot2: the 141 officially-named colors used in ggplot2, a common plotting package for the R programming language (e.g., MidnightBlue. MediumSeaGreen),[2]
Paint: The paint manufacturer Sherwin Williams has 7,750 named colors (e.g., Pompeii Red, Butter Up).[3]

4 Names to Colors

Our word-to-color model is used to predict a color in *Lab* space given the sequence of characters in a color's name, $\boldsymbol{c} = \langle c_1, c_2, \ldots, c_{|\boldsymbol{c}|} \rangle$, where each c_i is a character in a finite alphabet. Each character c_i is represented by learned vector embedding in $\mathbb{R}^{300}$. To build a color out of the sequence, we use an LSTM (Hochreiter and Schmidhuber, 1997) with 300 hidden units. The final hidden state is used as a vector representation $\mathbf{h} \in \mathbb{R}^{300}$ of the sequence. The associated color value in *Lab* space is then defined to be $\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{h} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{3\times 300}$ and $\mathbf{b} \in \mathbb{R}^3$ transform $\mathbf{h}$.

This model instantiates the one proposed by Ling et al. (2015) for learning word embeddings built from representations of characters.

To learn the parameters of the model (i.e., the parameters of the LSTMs, the character embeddings, and $\mathbf{W}$ and $\mathbf{b}$), we use reference color labels $\mathbf{y}$ from our training set and minimize squared error, $||\mathbf{y} - \hat{\mathbf{y}}||^2$, averaged across the training set. Learning is accomplished using backpropagation and the Adam update rule (Kingma and Ba, 2014).

4.1 Evaluation

We evaluated our model in two ways. First, we computed mean-squared error on held-out data using several variants of our model. The baseline models are linear regression models, which predict a color from a bag of character unigrams and bigrams. We compare an RNN and LSTMs with one and two layers. Table 2 shows that the two-layer LSTM achieves lower error than the unigram and bigram baselines and an RNN. We see the same pattern of results on the out-of-domain test sets.

The Color Turing Test. Our second evaluation attempts to assess whether our model's associations are human-like. For this evaluation, we asked human judges to choose the color better described by a name from one of our test sets: our model's predicted color or the color in the data. For each dataset, we randomly selected 20 examples. 111 judges considered each instance.[4] Judges were presented instances in random order and forced to make a choice between the two and explicitly directed to

[2] http://sape.inf.usi.ch/quick-reference/ggplot2/colour

[3] http://bit.ly/PaintColorNames

[4] We excluded results from an additional 19 annotators who made more than one mistake in a color blindness test (Oliver, 1888).

Preference	Test	ggplot2	Paint
Actual color	43.2%	32.6%	31.0%
Predicted color	56.7%	67.3%	69.0%

Table 3: Summary of color Turing test results.

make an arbitrary choice if neither was better.[5] The test is shown at `http://colorlab.us/turk`.

Results are shown in Table 3; on the ggplot2 and Paint datasets, our prediction is preferred to the actual names in a majority of cases. The Test dataset from COLOURlovers is a little bit challenging, with more noisy and creative names; still, in the majority of cases, our prediction is preferred.

4.2 Visualization and Exploration

To better understand our model, we provide illustrations of its predictions on several kinds of inputs.

Character by character prediction. We consider how our model reads color names character by character. Fig. 1 shows some examples, such as *blue*, variously modified. The word *deep* starts dark brown, but eventually modifies *blue* to a dark blue. Our model also performs sensibly on colors named after things (*mint*, *cream*, *sand*).

D	A	M	G	R
De	Aq	Mi	Go	Ro
Dee	Aqu	Min	Gol	Ros
Deep	Aqua	Mint	Gold	Rosy
Deep	Aqua	MintC	Gold	Rosy
Deep B	Aqua B	MintCr	Gold S	Rosy P
Deep Bl	Aqua Bl	MintCre	Gold Sa	Rosy Pi
Deep Blu	Aqua Blu	MintCrea	Gold San	Rosy Pin
Deep Blue	Aqua Blue	MintCream	Gold Sand	Rosy Pink

Figure 1: Visualization of character-by-character prediction.

Genre and color. We can use our model to investigate how colors are evoked in text by predicting the colors of each word in a text. Fig. 3 shows a colored recipe. Noting that many words are rendered in neutral grays and tans, we investigated how our model colors words in three corpora: 3,300 English poems (1800–present), 256 recipes from the CURD dataset (Tasse and Smith, 2008),[6] and 6,000 beer reviews.[7] For each corpus, we examine the distribution of Euclidean distances of $\hat{\mathbf{y}}$ from the *Lab* representation of the "middle" color RGB (128, 128, 128). The Euclidean distances from the mean are measuring the variance of the color of words in a document. Fig. 2 shows these distributions; recipes and beer reviews are more "colorful" than poems, under our model's learned definition of color.

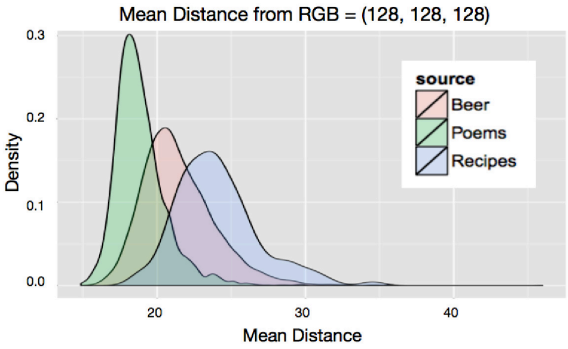
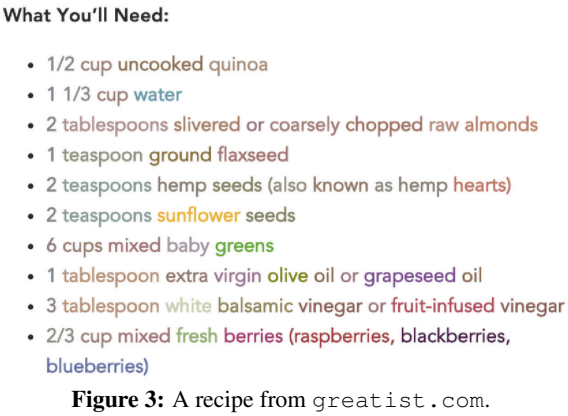


Figure 2: Distribution of Euclidean distances in *Lab* from estimated colors of words in each corpus to RGB (128, 128, 128).


What You'll Need:

- 1/2 cup uncooked quinoa
- 1 1/3 cup water
- 2 tablespoons slivered or coarsely chopped raw almonds
- 1 teaspoon ground flaxseed
- 2 teaspoons hemp seeds (also known as hemp hearts)
- 2 teaspoons sunflower seeds
- 6 cups mixed baby greens
- 1 tablespoon extra virgin olive oil or grapeseed oil
- 3 tablespoon white balsamic vinegar or fruit-infused vinegar
- 2/3 cup mixed fresh berries (raspberries, blackberries, blueberries)

Figure 3: A recipe from `greatist.com`.

5 Generating Names from Colors

The first of our two color naming models generates character sequences conditioned on *Lab* color representations, following other sequence-to-sequence approaches (Sutskever et al., 2014; Karpathy and Fei-Fei, 2014). The transformation is as follows: First, a linear transformation maps the color vector into 300 dimensions, together comprising the initial

[5] A preliminary study that allowed a judge to say that there was no difference led to a similar result.

[6] `http://www.cs.cmu.edu/~ark/CURD/`

[7] `http://beeradvocate.com`

hidden and memory vectors. Next a character LSTM is iteratively applied to the hidden, memory, and next-character vectors, and the next character produced by applying affine and then softmax functions to the hidden vector. The model is trained to maximize conditional likelihood of each character given its history. We used 300 dimensions for character embeddings and recurrence weights. The output vocabulary size was 98 without lowercasing.

We also propose a model to capture variations in color description with latent variables by extending the variational autoencoder (Kingma and Welling, 2013) to a conditional model. We want to model the conditional probability of word $\mathbf{y}$ and latent variables $\mathbf{z}$ given color $\mathbf{x}$. The latent variable gives the model capacity to account for the complexity of the color–word mapping. Since $p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = p(\mathbf{z})p(\mathbf{y} \mid \mathbf{x}, \mathbf{z})$, the variational objective is:

$$\begin{aligned}
&\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z} \mid \mathbf{x}) + \log p_\theta(\mathbf{y}, \mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z} \mid \mathbf{x}) + \log p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z})p(\mathbf{z})] \\
&\simeq -D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}^l)
\end{aligned}$$

The first term regularizes the shape of posterior, $q(\mathbf{z} \mid \mathbf{x})$, to be close to prior $p(\mathbf{z})$ where it is a Gaussian distribution, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The second term is the log likelihood of the character sequence conditioned on color values. To optimize θ and ϕ, we reparameterize the model, we write $\mathbf{z}$ in terms of a mean and variance and samples from a standard normal distribution, i.e., $\mathbf{z} = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We predict mean and log variance of the model with a multi-layer perceptron and initialize the decoder-LSTM with $\mathbf{h}_0 = \tanh(\mathbf{W}\mathbf{z} + \mathbf{b})$. We trained the model with mini-batch size 128 and Adam optimizer. The sample size L was set to 1.

Evaluation. We evaluated our models by estimating perplexity on the Test set (Table 1). Our baseline is a character-level unconditional LSTM language model. Conditioning on color improved per-character perplexity by 7% and the latent variable gave a further 3%; see Table 4.

A second dataset we evaluate on is the Munroe Color Corpus[8] which contains 2,176,417 color description for 829 words (i.e., single words have multiple color descriptions). Monroe et al. (2016) have developed word-based (rather character-based) recurrent neural network model.

[8] https://blog.xkcd.com/2010/05/03/color-survey-results/

Model	Perplexity
LSTM-LM	5.9
VAE	5.9
color-conditioned LSTM-LM	5.5
color-conditioned VAE	**5.3**

Table 4: Comparison of language models.

Our character-based model with 1024 hidden units achieved 12.48 per-description perplexity, marginally better than 12.58 obtained with a word-based neural network model reported in that work. Thus, we see that modeling color names as sequences of characters is wholly feasible. However, since the corpus only contains color description for 829 words, the model trained on the Munroe Color Corpus does not provide suitable supervision for evaluation on our more lexically diverse dataset.

6 Related Work and Discussion

Color is one of the lowest-level visual signals playing an important role in cognition (Wurm et al., 1993) and behavior (Maier et al., 2008; Lichtenfeld et al., 2009). It plays a role in human object recognition: to name an object, we first need to encode visual information such as shape and surface information including color and texture. Given a visual encoding, we search our memory for a structural, semantic and phonological description (Humphreys et al., 1999). Adding color information to shape significantly improves naming accuracy and speeds correct response times (Rossion et al., 2004).

Colors and their names have some association in our cognition. The Stroop (1935) effect is a well-known example showing interference of colors and color terms: when we see a color term printed in a different color—blue—it takes us longer to name the word, and we are more prone to naming errors than when the ink matches—blue (De Houwer, 2003).

Recent evidence suggests that colors and words are associated in the brain. The brain uses different regions to perceive various modalities, but processing a color word activates the same brain region as the color it denotes (del Prado Martín et al., 2006; Simmons et al., 2007).

Closer to NLP, the relationship between visual

stimuli and their linguistic descriptions by humans has been explored extensively through automatic text generation from images (Kiros et al., 2014; Karpathy and Fei-Fei, 2014; Xu et al., 2015). Color association with word semantics has also been investigated in several previous papers (Mohammad, 2011; Heer and Stone, 2012; Andreas and Klein, 2014; McMahan and Stone, 2015).

7 Conclusion

In this paper, we introduced a computational model to predict a point in color space from the sequence of characters in the color's name. Using a large set of color–name pairs obtained from a color design forum, we evaluate our model on a "color Turing test" and find that, given a name, the colors predicted by our model are preferred by annotators to color names created by humans. We also investigate the reverse mapping, from colors to names. We compare a conditional LSTM language model to a new latent-variable model, achieving a 10% perplexity reduction.

Acknowledgments

We thank Lucas Beyer for very helpful comments and discussions, and we also appreciate all the participants of our color Turing test.

References

Jacob Andreas and Dan Klein. 2014. Grounding language with points and paths in continuous spaces. In *CoNLL*, pages 58–67.

Arthur D. Broadbent. 2004. A critical review of the development of the CIE1931 RGB color-matching functions. *Color Research & Application*, 29(4):267–272.

Jan De Houwer. 2003. On the role of stimulus-response and stimulus-stimulus compatibility in the Stroop effect. *Memory & Cognition*, 31(3):353–359.

Fermín Moscoso del Prado Martín, Olaf Hauk, and Friedemann Pulvermüller. 2006. Category specificity in the processing of color-related and form-related words: An erp study. *Neuroimage*, 29(1):29–37.

Xiaoyan Deng, Sam K. Hui, and J. Wesley Hutchinson. 2010. Consumer preferences for color combinations: An empirical analysis of similarity-based color relationships. *Journal of Consumer Psychology*, 20(4):476–484.

Jeffrey Heer and Maureen Stone. 2012. Color naming models for color selection, image editing and palette design. In *Proc. CHI*.

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

Glyn W. Humphreys, Cathy J. Price, and M. Jane Riddoch. 1999. From objects to names: A cognitive neuroscience approach. *Psychological Research*, 62(2-3):118–130.

Richard S. Hunter. 1958. Photoelectric color difference meter. *Josa*, 48(12):985–993.

Andrej Karpathy and Li Fei-Fei. 2014. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603.

Stephanie Lichtenfeld, Markus A. Maier, Andrew J. Elliot, and Reinhard Pekrun. 2009. The semantic red effect: Processing the word red undermines intellectual performance. *Journal of Experimental Social Psychology*, 45(6):1273–1276.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *CoRR*, abs/1511.04586.

Markus A. Maier, Andrew J. Elliot, and Stephanie Lichtenfeld. 2008. Mediation of the negative effect of red on intellectual performance. *Personality and Social Psychology Bulletin*.

Aaron Marcus. 1991. *Graphic design for electronic documents and user interfaces*. ACM.

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Saif Mohammad. 2011. Colourful language: Measuring word-colour associations. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 97–106. Association for Computational Linguistics.

Will Monroe, Noah D. Goodman, and Christopher Potts. 2016. Learning to generate compositional color descriptions. In *Proc. EMNLP*.

Charles A Oliver. 1888. Tests for color-blindness. *Transactions of the American Ophthalmological Society*, 5:86.

Bruno Rossion, Gilles Pourtois, et al. 2004. Revisiting snodgrass and vanderwart's object pictorial set: The role of surface detail in basic-level object recognition. *PERCEPTION-LONDON-*, 33(2):217–236.

W. Kyle Simmons, Vimal Ramjee, Michael S. Beauchamp, Ken McRae, Alex Martin, and Lawrence W. Barsalou. 2007. A common neural substrate for perceiving and knowing about color. *Neuropsychologia*, 45(12):2802–2810.

J. Ridley Stroop. 1935. Studies of interference in serial verbal reactions. *Journal of experimental psychology*, 18(6):643.

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

Dan Tasse and Noah A Smith. 2008. Sour cream: Toward semantic processing of recipes. Technical report, Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.

Tony Veale and Khalid Al-Najjar. 2015. Unweaving the lexical rainbow: Grounding linguistic creativity in perceptual semantics.

Lee H. Wurm, Gordon E. Legge, Lisa M. Isenberg, and Andrew Luebker. 1993. Color improves object recognition in normal and low vision. *Journal of Experimental Psychology: Human perception and performance*, 19(4):899.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Analyzing the Behavior of Visual Question Answering Models

Aishwarya Agrawal[*], **Dhruv Batra**[†,*], **Devi Parikh**[†,*]
[*]Virginia Tech [†]Georgia Institute of Technology
{aish, dbatra, parikh}@vt.edu

Abstract

Recently, a number of deep-learning based models have been proposed for the task of Visual Question Answering (VQA). The performance of most models is clustered around 60-70%. In this paper we propose systematic methods to analyze the behavior of these models as a first step towards recognizing their strengths and weaknesses, and identifying the most fruitful directions for progress. We analyze two models, one each from two major classes of VQA models – with-attention and without-attention and show the similarities and differences in the behavior of these models. We also analyze the winning entry of the VQA Challenge 2016.

Our behavior analysis reveals that despite recent progress, today's VQA models are "myopic" (tend to fail on sufficiently novel instances), often "jump to conclusions" (converge on a predicted answer after 'listening' to just half the question), and are "stubborn" (do not change their answers across images).

1 Introduction

Visual Question Answering (VQA) is a recently-introduced (Antol et al., 2015; Geman et al., 2014; Malinowski and Fritz, 2014) problem where given an image and a natural language question (e.g., "What kind of store is this?", "How many people are waiting in the queue?"), the task is to automatically produce an accurate natural language answer ("bakery", "5"). A flurry of recent deep-learning based models have been proposed for VQA (Antol et al., 2015; Chen et al., 2015; Yang et al., 2016; Xu and Saenko, 2016; Jiang et al., 2015; Andreas et al., 2016a; Wang et al., 2015; Kafle and Kanan, 2016; Lu et al., 2016; Andreas et al., 2016b; Shih et al., 2016; Kim et al., 2016; Fukui et al., 2016; Noh and Han, 2016; Ilievski et al., 2016; Wu et al., 2016; Xiong et al., 2016; Zhou et al., 2015; Saito et al., 2016). Curiously, the performance of most methods is clustered around 60-70% (compared to human performance of 83% on open-ended task and 91% on multiple-choice task) with a mere 5% gap between the top-9 entries on the VQA Challenge 2016.[1] It seems clear that as a first step to understand these models, to meaningfully compare strengths and weaknesses of different models, to develop insights into their failure modes, and to identify the most fruitful directions for progress, it is crucial to develop techniques to understand the behavior of VQA models.

In this paper, we develop novel techniques to characterize the behavior of VQA models. As concrete instantiations, we analyze two VQA models (Lu et al., 2015; Lu et al., 2016), one each from two major classes of VQA models – with-attention and without-attention. We also analyze the winning entry (Fukui et al., 2016) of the VQA Challenge 2016.

2 Related Work

Our work is inspired by previous works that diagnose the failure modes of models for different tasks. (Karpathy et al., 2016) constructed a series of oracles to measure the performance of a character level

[1] http://www.visualqa.org/challenge.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1955–1960,
Austin, Texas, November 1-5, 2016.

language model. (Hoiem et al., 2012) provided analysis tools to facilitate detailed and meaningful investigation of object detector performance. This paper aims to perform behavior analyses as a first step towards diagnosing errors for VQA.

(Yang et al., 2016) categorize the errors made by their VQA model into four categories – model focuses attention on incorrect regions, model focuses attention on appropriate regions but predicts incorrect answers, predicted answers are different from labels but might be acceptable, labels are wrong. While these are coarse but useful failure modes, we are interested in understanding the behavior of VQA models along specific dimensions – whether they generalize to novel instances, whether they listen to the entire question, whether they look at the image.

3 Behavior Analyses

We analyze the behavior of VQA models along the following three dimensions –

Generalization to novel instances: We investigate whether the test instances that are incorrectly answered are the ones that are "novel" i.e., not similar to training instances. The novelty of the test instances may be in two ways – 1) the test question-image (QI) pair is "novel", i.e., too different from training QI pairs; and 2) the test QI pair is "familiar", but the answer required at test time is "novel", i.e., answers seen during training are different from what needs to be produced for the test QI pairs.

Complete question understanding: To investigate whether a VQA model is understanding the input question or not, we analyze if the model 'listens' to only first few words of the question or the entire question, if it 'listens' to only question (wh) words and nouns or all the words in the question.

Complete image understanding: The absence of a large gap between performance of language-alone and language + vision VQA models (Antol et al., 2015) provides evidence that current VQA models seem to be heavily reliant on the language model, perhaps not really understanding the image. In order to analyze this behavior, we investigate whether the predictions of the model change across images for a given question.

We present our behavioral analyses on the VQA dataset (Antol et al., 2015). VQA is a large-scale free-form natural-language dataset containing $\sim$0.25M images, $\sim$0.76M questions, and $\sim$10M answers, with open-ended and multiple-choice modalities for answering the visual questions. All the experimental results are reported on the VQA validation set using the following models trained on the VQA training set for the open-ended task –

CNN + LSTM based model without-attention (CNN+LSTM): We use the best performing model of (Antol et al., 2015) (code provided by (Lu et al., 2015)), which achieves an accuracy of 54.13% on the VQA validation set. It is a two channel model – one channel processes the image (using Convolutional Neural Network (CNN) to extract image features) and the other channel processes the question (using Long Short-Term Memory (LSTM) recurrent neural network to obtain question embedding). The image and question features obtained from the two channels are combined and passed through a fully connected (FC) layer to obtain a softmax distribution over the space of answers.

CNN + LSTM based model with-attention (ATT): We use the top-entry on the VQA challenge leaderboard (as of June 03, 2016) (Lu et al., 2016), which achieves an accuracy of 57.02% on the VQA validation set.[2] This model jointly reasons about image and question attention, in a hierarchical fashion. The attended image and question features obtained from different levels of the hierarchy are combined and passed through a FC layer to obtain a softmax distribution over the space of answers.

VQA Challenge 2016 winning entry (MCB): This is the multimodal compact bilinear (mcb) pooling model from (Fukui et al., 2016) which won the real image track of the VQA Challenge 2016. This model achieves an accuracy of 60.36% on the VQA validation set.[3] In this model, multimodal compact bilinear pooling is used to predict attention over image features and also to combine the attended image features with the question features. These combined features are passed through a FC layer to obtain a softmax distribution over the space of answers.

[2]Code available at https://github.com/jiasenlu/HieCoAttenVQA

[3]Code available at https://github.com/akirafukui/vqa-mcb

3.1 Generalization to novel instances

Do VQA models make mistakes because test instances are too different from training ones? To analyze the first type of novelty (the test QI pair is novel), we measure the correlation between test accuracy and distance of test QI pairs from its k nearest neighbor (k-NN) training QI pairs. For each test QI pair we find its k-NNs in the training set and compute the average distance between the test QI pair and its k-NNs. The k-NNs are computed in the space of combined image + question embedding (just before passing through FC layer) for all the three models (using euclidean distance metric for the CNN+LSTM model and cosine distance metric for the ATT and MCB models).

The correlation between accuracy and average distance is significant (-0.41 at k=50[4] for the CNN+LSTM model and -0.42 at k=15[5] for the ATT model). A high negative correlation value tells that the model is less likely to predict correct answers for test QI pairs which are not very similar to training QI pairs, suggesting that the model is not very good at generalizing to novel test QI pairs. The correlation between accuracy and average distance is not significant for the MCB model (-0.14 at k=1[6]) suggesting that MCB is better at generalizing to novel test QI pairs.

We also found that 67.5% of mistakes made by the CNN+LSTM model *can be successfully predicted* by checking distance of test QI pair from its k-NN training QI pairs (66.7% for the ATT model, 55.08% for the MCB model). Thus, this analysis not only exposes a reason for mistakes made by VQA models, but also allows us to build human-like models that can predict their own oncoming failures, and potentially refuse to answer questions that are 'too different' from ones seen in past.

To analyze the second type of novelty (the answer required at test time is not familiar), we compute the correlation between test accuracy and the average distance of the test ground truth (GT) answer with GT answers of its k-NN training QI pairs. The distance between answers is computed in the space of average Word2Vec (Mikolov et al., 2013) vectors of answers. This correlation turns out to be quite high (-0.62) for both CNN+LSTM and ATT models and significant (-0.47) for the MCB model. A high negative correlation value tells that the model tends to regurgitate answers seen during training.

Figure 1: Examples from test set where the CNN+LSTM model makes mistakes and their corresponding nearest neighbor training instances. See supplementary for more examples.

These distance features are also good at predicting failures – 74.19% of failures can be predicted by checking distance of test GT answer with GT answers of its k-NN training QI pairs for CNN+LSTM model (75.41% for the ATT model, 70.17% for the MCB model). Note that unlike the previous analysis, this analysis only explains failures but cannot be used to predict failures (since it uses GT labels). See Fig. 1 for qualitative examples.

From Fig. 1 (row1) we can see that the test QI pair is semantically quite different from its k-NN training QI pairs ({1st, 2nd, 3rd}-NN distances are {15.05, 15.13, 15.17}, which are higher than the corresponding distances averaged across all success cases: {8.74, 9.23, 9.50.}), explaining the mistake. Row2 shows an example where the model has seen the same question in the training set (test QI pair is semantically similar to training QI pairs) but, since it has not seen "green cone" for training instances (answers seen during training are different from what needs to be produced for the test QI pair), it is unable to answer the test QI pair correctly. This shows that current models lack compositionality: the ability to combine the concepts of "cone" and "green" (both of which have been seen in training set) to answer "green cone" for the test QI pair. This compositionality is desirable and central to intelligence.

[4] k=50 leads to highest correlation
[5] k=15 leads to highest correlation
[6] k=1 leads to highest correlation

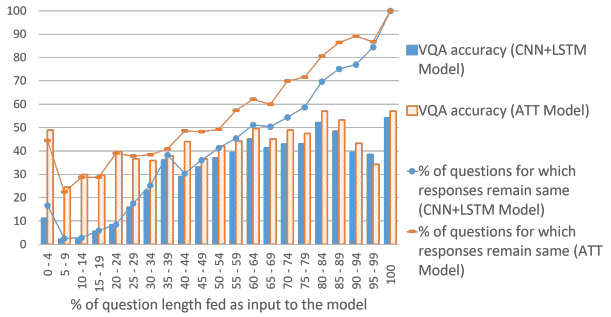


Figure 2: X-axis shows length of partial question (in %) fed as input. Y-axis shows percentage of questions for which responses of these partial questions are the same as full questions and VQA accuracy of partial questions.

3.2 Complete question understanding

We feed partial questions of increasing lengths (from 0-100% of question from left to right). We then compute what percentage of responses do not change when more and more words are fed.

Fig. 2 shows the test accuracy and percentage of questions for which responses remain same (compared to entire question) as a function of partial question length. We can see that for 40% of the questions, the CNN+LSTM model seems to have converged on a predicted answer after 'listening' to just half the question. This shows that the model is listening to first few words of the question more than the words towards the end. Also, the model has 68% of the final accuracy (54%) when making predictions based on half the original question. When making predictions just based on the image, the accuracy of the model is 24%. The ATT model seems to have converged on a predicted answer after listening to just half the question more often (49% of the time), achieving 74% of the final accuracy (57%). The MCB model converges on a predicted answer after listening to just half the question 45% of the time, achieving 67% of the final accuracy (60%). See Fig. 3 for qualitative examples.

We also analyze the change in responses of the model's predictions (see Fig. 4), when words of a particular part-of-the-speech (POS) tag are dropped from the question. The experimental results indicate that wh-words effect the model's decisions the most (most of the responses get changed on dropping these words from the question), and that pronouns effect the model's decisions the least.

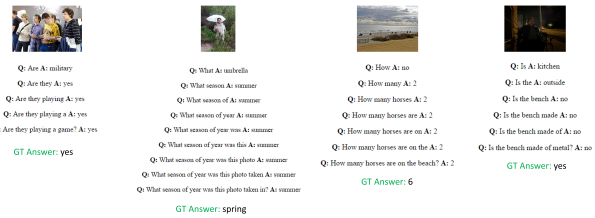


Figure 3: Examples where the CNN+LSTM model does not change its answer after first few question words. On doing so, it is correct for some cases (the extreme left example) and incorrect for other cases (the remaining three examples). See supplementary for more examples.

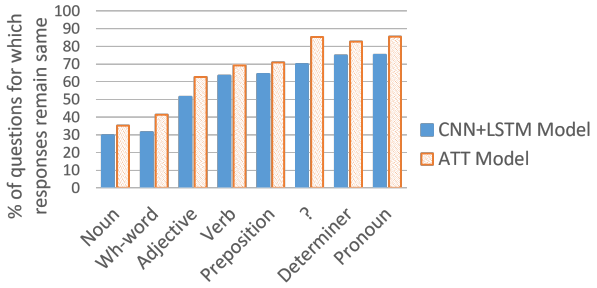


Figure 4: Percentage of questions for which responses remain same (compared to entire question) as a function of POS tags dropped from the question.

3.3 Complete image understanding

Does a VQA model really 'look' at the image? To analyze this, we compute the percentage of the time (say X) the response does not change across images (e.g.,, answer for all images is "2") for a given question (e.g., "How many zebras?") and plot histogram of X across questions (see Fig. 5). We do this analysis for questions occurring for atleast 25 images in the VQA validation set, resulting in total 263 questions. The cumulative plot indicates that for 56% questions, the CNN+LSTM model outputs the same answer for at least half the images. This is fairly high, suggesting that the model is picking the same answer no matter what the image is. Promisingly, the ATT and MCB models (that do not work with a holistic entire-image representation and purportedly pay attention to specific spatial regions in an image) produce the same response for at least half the images for fewer questions (42% for the ATT model, 40% for the MCB model).

Interestingly, the average accuracy (see the VQA accuracy plots in Fig. 5) for questions for which the models produce same response for $>$50% and $<$55% of the images is 56% for the CNN+LSTM

Figure 5: Histogram of percentage of images for which model produces same answer for a given question and its comparison with test accuracy. The cumulative plot shows the % of questions for which model produces same answer for *atleast* x % of images.

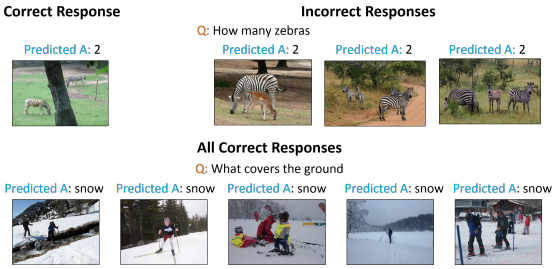


Figure 6: Examples where the predicted answers do not change across images for a given question. See supplementary for more examples.

model (60% for the ATT model, 73% for the MCB model) which is more than the respective average accuracy on the entire VQA validation set (54.13% for the CNN+LSTM model, 57.02% for the ATT model, 60.36% for the MCB model). Thus, producing the same response across images seems to be statistically favorable. Fig. 6 shows examples where the CNN+LSTM model predicts the same response across images for a given question. The first row shows examples where the model makes errors on several images by predicting the same answer for all images. The second row shows examples where the model is always correct even if it predicts the same answer across images. This is so because questions such as *"What covers the ground?"* are asked for an image in the VQA dataset only when ground is covered with snow (because subjects were looking at the image while asking questions about it). Thus, this analysis exposes label biases in the dataset. Label biases (in particular, for "yes/no" questions) have also been reported in (Zhang et al., 2016).

4 Conclusion

We develop novel techniques to characterize the behavior of VQA models, as a first step towards understanding these models, meaningfully comparing the strengths and weaknesses of different models, developing insights into their failure modes, and identifying the most fruitful directions for progress. Our behavior analysis reveals that despite recent progress, today's VQA models are "myopic" (tend to fail on sufficiently novel instances), often "jump to conclusions" (converge on a predicted answer after 'listening' to just half the question), and are "stubborn" (do not change their answers across images), with attention based models being less "stubborn" than non-attention based models.

As a final thought, we note that the somewhat pathological behaviors exposed in the paper are in some sense "correct" given the model architectures and the dataset being trained on. Ignoring optimization error, the maximum-likelihood training objective is clearly intended to capture statistics of the dataset. Our motive is simply to better understand current generation models via their behaviors, and use these observations to guide future choices – do we need novel model classes? or dataset with different biases? etc. Finally, it should be clear that our use of anthropomorphic adjectives such as "stubborn", "myopic" etc. is purely for pedagogical reasons – to easily communicate our observations to our readers. No claims are being made about today's VQA models being human-like.

Acknowledgements

We would like to thank the EMNLP reviewers for valuable feedback and Yash Goyal for sharing his code. This work was supported in part by: NSF CAREER awards, ARO YIP awards, ICTAS Junior Faculty awards, Google Faculty Research awards, awarded to both DB and DP, ONR grant N00014-14-1-0679, AWS in Education Research grant, NVIDIA GPU donation, awarded to DB, Paul G. Allen Family Foundation Allen Distinguished Investigator award, ONR YIP and Alfred P. Sloan Fellowship, awarded to DP. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government or any sponsor.

References

[Andreas et al.2016a] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Deep compositional question answering with neural module networks. In *CVPR*. 1

[Andreas et al.2016b] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Learning to compose neural networks for question answering. In *NAACL*. 1

[Antol et al.2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*. 1, 2

[Chen et al.2015] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. ABC-CNN: an attention based convolutional neural network for visual question answering. *CoRR*, abs/1511.05960. 1

[Fukui et al.2016] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*. 1, 2

[Geman et al.2014] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. 2014. A Visual Turing Test for Computer Vision Systems. In *PNAS*. 1

[Hoiem et al.2012] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. 2012. Diagnosing error in object detectors. In *ECCV*. 2

[Ilievski et al.2016] Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. 2016. A focused dynamic attention model for visual question answering. *CoRR*, abs/1604.01485. 1

[Jiang et al.2015] Aiwen Jiang, Fang Wang, Fatih Porikli, and Yi Li. 2015. Compositional memory for visual question answering. *CoRR*, abs/1511.05676. 1

[Kafle and Kanan2016] Kushal Kafle and Christopher Kanan. 2016. Answer-type prediction for visual question answering. In *CVPR*. 1

[Karpathy et al.2016] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *ICLR Workshop*. 1

[Kim et al.2016] Jin-Hwa Kim, Sang-Woo Lee, Dong-Hyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2016. Multimodal residual learning for visual QA. In *NIPS*. 1

[Lu et al.2015] Jiasen Lu, Xiao Lin, Dhruv Batra, and Devi Parikh. 2015. Deeper lstm and normalized cnn visual question answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN`. 1, 2

[Lu et al.2016] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*. 1, 2

[Malinowski and Fritz2014] Mateusz Malinowski and Mario Fritz. 2014. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. In *NIPS*. 1

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*. 3

[Noh and Han2016] Hyeonwoo Noh and Bohyung Han. 2016. Training recurrent answering units with joint loss minimization for vqa. *CoRR*, abs/1606.03647. 1

[Saito et al.2016] Kuniaki Saito, Andrew Shin, Yoshitaka Ushiku, and Tatsuya Harada. 2016. Dualnet: Domain-invariant network for visual question answering. *CoRR*, abs/1606.06108. 1

[Shih et al.2016] Kevin J. Shih, Saurabh Singh, and Derek Hoiem. 2016. Where to look: Focus regions for visual question answering. In *CVPR*. 1

[Wang et al.2015] Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. 2015. Explicit knowledge-based reasoning for visual question answering. *CoRR*, abs/1511.02570. 1

[Wu et al.2016] Qi Wu, Peng Wang, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. 2016. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*. 1

[Xiong et al.2016] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. 1

[Xu and Saenko2016] Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*. 1

[Yang et al.2016] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2016. Stacked attention networks for image question answering. In *CVPR*. 1, 2

[Zhang et al.2016] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and Yang: Balancing and answering binary visual questions. In *CVPR*. 5

[Zhou et al.2015] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *CoRR*, abs/1512.02167. 1

Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text

Subhashini Venugopalan UT Austin vsub@cs.utexas.edu
Lisa Anne Hendricks UC Berkeley lisa_anne@berkeley.edu
Raymond Mooney UT Austin mooney@cs.utexas.edu
Kate Saenko Boston University saenko@bu.edu

Abstract

This paper investigates how linguistic knowledge mined from large text corpora can aid the generation of natural language descriptions of videos. Specifically, we integrate both a neural language model and distributional semantics trained on large text corpora into a recent LSTM-based architecture for video description. We evaluate our approach on a collection of Youtube videos as well as two large movie description datasets showing significant improvements in grammaticality while modestly improving descriptive quality.

1 Introduction

The ability to automatically describe videos in natural language (NL) enables many important applications including content-based video retrieval and video description for the visually impaired. The most effective recent methods (Venugopalan et al., 2015a; Yao et al., 2015) use recurrent neural networks (RNN) and treat the problem as machine translation (MT) from video to natural language. Deep learning methods such as RNNs need large training corpora; however, there is a lack of high-quality paired video-sentence data. In contrast, raw text corpora are widely available and exhibit rich linguistic structure that can aid video description. Most work in statistical MT utilizes both a language model trained on a large corpus of monolingual target language data as well as a translation model trained on more limited parallel bilingual data. This paper explores methods to incorporate knowledge from language corpora to capture general linguistic regularities to aid video description.

This paper integrates linguistic information into a video-captioning model based on Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) RNNs which have shown state-of-the-art performance on the task. Further, LSTMs are also effective as language models (LMs) (Sundermeyer et al., 2010). Our first approach (early fusion) is to pre-train the network on plain text before training on parallel video-text corpora. Our next two approaches, inspired by recent MT work (Gulcehre et al., 2015), integrate an LSTM LM with the existing video-to-text model. Furthermore, we also explore replacing the standard one-hot word encoding with distributional vectors trained on external corpora.

We present detailed comparisons between the approaches, evaluating them on a standard Youtube corpus and two recent large movie description datasets. The results demonstrate significant improvements in grammaticality of the descriptions (as determined by crowdsourced human evaluations) and more modest improvements in descriptive quality (as determined by both crowdsourced human judgements and standard automated comparison to human-generated descriptions). Our main contributions are 1) multiple ways to incorporate knowledge from external text into an existing captioning model, 2) extensive experiments comparing the methods on three large video-caption datasets, and 3) human judgements to show that external linguistic knowledge has a significant impact on grammar.

2 LSTM-based Video Description

We use the successful S2VT video description framework from Venugopalan et al. (2015a) as our

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1961–1966,
Austin, Texas, November 1-5, 2016.

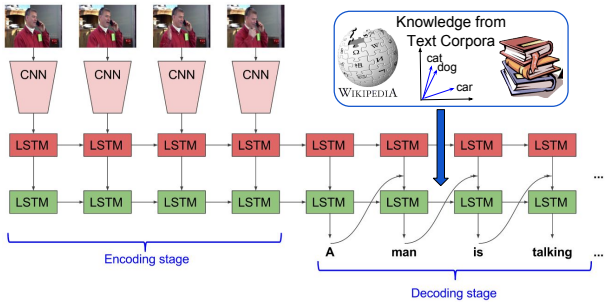


Figure 1: The S2VT architecture encodes a sequence of frames and decodes them to a sentence. We propose to add knowledge from text corpora to enhance the quality of video description.

underlying model and describe it briefly here. S2VT uses a sequence to sequence approach (Sutskever et al., 2014; Cho et al., 2014) that maps an input $\vec{x} = (x_1, ..., x_T)$ video frame feature sequence to a fixed dimensional vector and then decodes this into a sequence of output words $\vec{y} = (y_1, ..., y_N)$.

As shown in Fig. 1, it employs a stack of two LSTM layers. The input $\vec{x}$ to the first LSTM layer is a sequence of frame features obtained from the penultimate layer (fc_7) of a Convolutional Neural Network (CNN) after the ReLu operation. This LSTM layer encodes the video sequence. At each time step, the hidden control state h_t is provided as input to a second LSTM layer. After viewing all the frames, the second LSTM layer learns to decode this state into a sequence of words. This can be viewed as using one LSTM layer to model the visual features, and a second LSTM layer to model language conditioned on the visual representation. We modify this architecture to incorporate linguistic knowledge at different stages of the training and generation process. Although our methods use S2VT, they are sufficiently general and could be incorporated into other CNN-RNN based captioning models.

3 Approach

Existing visual captioning models (Vinyals et al., 2015; Donahue et al., 2015) are trained solely on text from the caption datasets and tend to exhibit some linguistic irregularities associated with a restricted language model and a small vocabulary. Here, we investigate several techniques to integrate prior linguistic knowledge into a CNN/LSTM-based network for video to text (S2VT) and evaluate their effectiveness at improving the overall description.

Early Fusion. Our first approach (*early fusion*), is to pre-train portions of the network modeling language on large corpora of raw NL text and then continue "fine-tuning" the parameters on the paired video-text corpus. An LSTM model learns to estimate the probability of an output sequence given an input sequence. To learn a language model, we train the LSTM layer to predict the next word given the previous words. Following the S2VT architecture, we embed one-hot encoded words in lower dimensional vectors. The network is trained on web-scale text corpora and the parameters are learned through backpropagation using stochastic gradient descent.[1] The weights from this network are then used to *initialize* the embedding and weights of the LSTM layers of S2VT, which is then trained on video-text data. This trained LM is also used as the LSTM LM in the late and deep fusion models.

Late Fusion. Our late fusion approach is similar to how neural machine translation models incorporate a trained language model during decoding. At each step of sentence generation, the video caption model proposes a distribution over the vocabulary. We then use the language model to re-score the final output by considering the weighted average of the sum of scores proposed by the LM as well as the S2VT video-description model (VM). More specifically, if y_t denotes the output at time step t, and if p_{VM} and p_{LM} denote the proposal distributions of the video captioning model, and the language models respectively, then for all words $y' \in V$ in the vocabulary we can recompute the score of each new word, $p(y_t = y')$ as:

$$\alpha \cdot p_{VM}(y_t = y') + (1 - \alpha) \cdot p_{LM}(y_t = y') \quad (1)$$

Hyper-parameter α is tuned on the validation set.

Deep Fusion. In the deep fusion approach (Fig. 2), we integrate the LM a step deeper in the generation process by concatenating the hidden state of the language model LSTM (h_t^{LM}) with the hidden state of the S2VT video description model (h_t^{VM}) and use the combined latent vector to predict the output word. This is similar to the technique proposed by Gulcehre et al. (2015) for incorporating language models trained on monolingual corpora for machine translation. However, our approach differs in two

[1] The LM was trained to achieve a perplexity of 120

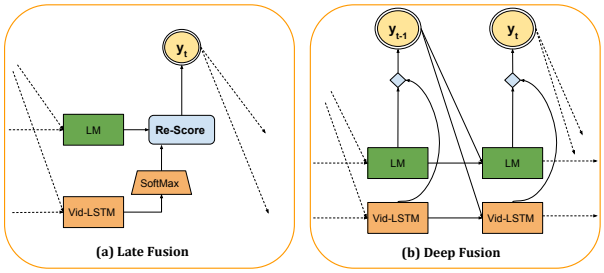


Figure 2: Illustration of our late and deep fusion approaches to integrate an independently trained LM to aid video captioning. The deep fusion model learns jointly from the hidden representations of the LM and S2VT video-to-text model (Vid-LSTM), whereas the late fusion re-scores the softmax output of the video-to-text model.

key ways: (1) we only concatenate the hidden states of the S2VT LSTM and language LSTM and do not use any additional context information, (2) we fix the weights of the LSTM language model but train the full video captioning network. In this case, the probability of the predicted word at time step t is:

$$p(y_t|\vec{y}_{<t}, \vec{x}) \propto \exp(\mathrm{Wf}(h_t^{VM}, h_t^{LM}) + b) \quad (2)$$

where $\vec{x}$ is the visual feature input, W is the weight matrix, and b the biases. We avoid tuning the LSTM LM to prevent overwriting already learned weights of a strong language model. But we train the full video caption model to incorporate the LM outputs while training on the caption domain.

Distributional Word Representations. The S2VT network, like most image and video captioning models, represents words using a 1-of-N (one hot) encoding. During training, the model learns to embed "one-hot" words into a lower 500d space by applying a linear transformation. However, the embedding is learned only from the limited and possibly noisy text in the caption data. There are many approaches (Mikolov et al., 2013; Pennington et al., 2014) that use large text corpora to learn vector-space representations of words that capture fine-grained semantic and syntactic regularities. We propose to take advantage of these to aid video description. Specifically, we replace the embedding matrix from one-hot vectors and instead use 300-dimensional GloVe vectors (Pennington et al., 2014) pre-trained on 6B tokens from Gigaword and Wikipedia 2014. In addition to using the distributional vectors for the input, we also explore variations where the model predicts both the one-hot word (trained on the $\mathrm{softmax}$ loss), as well as predicting the distributional vector from the LSTM hidden state using Euclidean loss as the objective. Here the output vector (y_t) is computed as $y_t = (W_g h_t + b_g)$, and the loss is given by:

$$\mathbb{L}(y_t, w_{glove}) = \|(W_g h_t + b_g) - w_{glove}\|^2 \quad (3)$$

where h_t is the LSTM output, w_{glove} is the word's GloVe embedding and W, b are weights and biases. The network then essentially becomes a multi-task model with two loss functions. However, we use this loss only to influence the weights learned by the network, the predicted word embedding is not used.

Ensembling. The overall loss function of the video-caption network is non-convex, and difficult to optimize. In practice, using an ensemble of networks trained slightly differently can improve performance (Hansen and Salamon, 1990). In our work we also present results of an ensemble by averaging the predictions of the best performing models.

4 Experiments

Datasets. Our language model was trained on sentences from Gigaword, BNC, UkWaC, and Wikipedia. The vocabulary consisted of 72,700 most frequent tokens also containing GloVe embeddings. Following the evaluation in Venugopalan et al. (2015a), we compare our models on the Youtube dataset (Chen and Dolan, 2011), as well as two large movie description corpora: MPII-MD (Rohrbach et al., 2015) and M-VAD (Torabi et al., 2015).

Evaluation Metrics. We evaluate performance using machine translation (MT) metrics METEOR (Denkowski and Lavie, 2014) and BLEU (Papineni et al., 2002) to compare the machine-generated descriptions to human ones. For the movie corpora which have just a single description we use only METEOR which is more robust.

Human Evaluation. We also obtain human judgements using Amazon Turk on a random subset of 200 video clips for each dataset. Each sentence was rated by 3 workers on a Likert scale of 1 to 5 (higher is better) for relevance and grammar. No video was provided during grammar evaluation. For movies, due to copyright, we only evaluate on grammar.

Model	METEOR	B-4	Relevance	Grammar
S2VT	29.2	37.0	2.06	3.76
Early Fusion	29.6	37.6	-	-
Late Fusion	29.4	37.2	-	-
Deep Fusion	29.6	39.3	-	-
Glove	30.0	37.0	-	-
Glove+Deep				
- Web Corpus	30.3	38.1	2.12	4.05*
- In-Domain	30.3	38.8	2.21*	4.17*
Ensemble	**31.4**	**42.1**	**2.24***	**4.20***

Table 1: Youtube dataset: METEOR and BLEU@4 in %, and human ratings (1-5) on relevance and grammar. Best results in bold, * indicates significant over S2VT.

4.1 Youtube Video Dataset Results

Comparison of the proposed techniques in Table 1 shows that Deep Fusion performs well on both METEOR and BLEU; incorporating Glove embeddings substantially increases METEOR, and combining them both does best. Our final model is an ensemble (weighted average) of the Glove, and the two Glove+Deep Fusion models trained on the external and in-domain COCO (Lin et al., 2014) sentences. We note here that the state-of-the-art on this dataset is achieved by HRNE (Pan et al., 2015) (METEOR 33.1) which proposes a superior visual processing pipeline using attention to encode the video.

Human ratings also correlate well with the METEOR scores, confirming that our methods give a modest improvement in descriptive quality. However, incorporating linguistic knowledge significantly[2] improves the grammaticality of the results, making them more comprehensible to human users.

Embedding Influence. We experimented multiple ways to incorporate word embeddings: *(1) GloVe input:* Replacing one-hot vectors with GloVe on the LSTM input performed best. *(2) Fine-tuning:* Initializing with GloVe and subsequently fine-tuning the embedding matrix reduced validation results by 0.4 METEOR. *(3) Input and Predict.* Training the LSTM to accept and predict GloVe vectors, as described in Section 3, performed similar to (1). All scores reported in Tables 1 and 2 correspond to the setting in (1) with GloVe embeddings only as input.

[2]Using the Wilcoxon Signed-Rank test, results were significant with $p < 0.02$ on relevance and $p < 0.001$ on grammar.

Model	MPII-MD		M-VAD	
	METEOR	Grammar	METEOR	Grammar
S2VT†	6.5	2.6	6.6	2.2
Early Fusion	6.7	-	6.8	-
Late Fusion	6.5	-	6.7	-
Deep Fusion	6.8	-	6.8	-
Glove	6.7	3.9*	6.7	3.1*
Glove+Deep	6.8	**4.1***	6.7	**3.3***

Table 2: Movie Corpora: METEOR (%) and human grammar ratings (1-5, higher is better). Best results in bold, * indicates significant over S2VT.

S2VT: Someone sits in the bed.
Glove: Someone sits on the couch and watches her phone.
Glove+Deep: Someone sits on the couch, watching her, her feet on her lap.
GT: Someone drops the flowers and kisses someone.

Figure 3: Two frames from a clip. Models generate visually relevant sentences but differ from groundtruth (GT).

4.2 Movie Description Results

Results on the movie corpora are presented in Table 2. Both MPII-MD and M-VAD have only a single ground truth description for each video, which makes both learning and evaluation very challenging (E.g. Fig.3). METEOR scores are fairly low on both datasets since generated sentences are compared to a single reference translation. S2VT† is a re-implementation of the base S2VT model with the new vocabulary and architecture (embedding dimension). We observe that the ability of external linguistic knowledge to improve METEOR scores on these challenging datasets is small but consistent. Again, human evaluations show significant (with $p < 0.0001$) improvement in grammatical quality.

5 Related Work

Following the success of LSTM-based models on Machine Translation (Sutskever et al., 2014; Bahdanau et al., 2015), and image captioning (Vinyals et al., 2015; Donahue et al., 2015), recent video description works (Venugopalan et al., 2015b; Venugopalan et al., 2015a; Yao et al., 2015) propose CNN-RNN based models that generate a vector representation for the video and "decode" it using an LSTM sequence model to generate a description. Venugopalan et al. (2015b) also incorporate external data such as images with captions to improve

video description, however in this work, our focus is on integrating external linguistic knowledge for video captioning. We specifically investigate the use of distributional semantic embeddings and LSTM-based language models trained on external text corpora to aid existing CNN-RNN based video description models.

LSTMs have proven to be very effective language models (Sundermeyer et al., 2010). Gulcehre et al. (2015) developed an LSTM model for machine translation that incorporates a monolingual language model for the target language showing improved results. We utilize similar approaches (late fusion, deep fusion) to train an LSTM for translating video to text that exploits large monolingual-English corpora (Wikipedia, BNC, UkWac) to improve RNN based video description networks. However, unlike Gulcehre et al. (2015) where the monolingual LM is used only to tune specific parameters of the translation network, the key advantage of our approach is that the output of the monolingual language model is used (as an input) when training the full underlying video description network.

Contemporaneous to us, Yu et al. (2015), Pan et al. (2015) and Ballas et al. (2016) propose video description models focusing primarily on improving the video representation itself using a hierarchical visual pipeline, and attention. Without the attention mechanism their models achieve METEOR scores of 31.1, 32.1 and 31.6 respectively on the Youtube dataset. The interesting aspect, as demonstrated in our experiments (Table 1), is that the contribution of language alone is considerable and only slightly less than the visual contribution on this dataset. Hence, it is important to focus on both aspects to generate better descriptions.

6 Conclusion

This paper investigates multiple techniques to incorporate linguistic knowledge from text corpora to aid video captioning. We empirically evaluate our approaches on Youtube clips as well as two movie description corpora. Our results show significant improvements on human evaluations of grammar while modestly improving the overall descriptive quality of sentences on all datasets. While the proposed techniques are evaluated on a specific video-caption network, they are generic and can be applied to many captioning models. The code and models are shared on `http://vsubhashini.github.io/language_fusion.html`.

Figure 4: Representative frames from clips in the movie description corpora. S2VT is the baseline model, Glove indicates the model trained with input Glove vectors, and Glove+Deep uses input Glove vectors with the Deep Fusion approach. GT indicates groundtruth sentence.

Acknowledgements

This work was supported by NSF awards IIS-1427425 and IIS-1212798, and ONR ATL Grant N00014-11-1-010, and DARPA under AFRL grant FA8750-13-2-0026. Raymond Mooney and Kate Saenko also acknowledge support from a Google grant. Lisa Anne Hendricks is supported by the National Defense Science and Engineering Graduate (NDSEG) Fellowship.

References

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

[Ballas et al.2016] Nicolas Ballas, Li Yao, Chris Pal, and Aaron C. Courville. 2016. Delving deeper into convolutional networks for learning video representations. *ICLR*.

[Chen and Dolan2011] David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL*.

[Cho et al.2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

[Denkowski and Lavie2014] Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL*.

[Donahue et al.2015] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

[Gulcehre et al.2015] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.C. Lin, F. Bougares, H. Schwenk, and Y. Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

[Hansen and Salamon1990] L. K. Hansen and P. Salamon. 1990. Neural network ensembles. *IEEE TPAMI*, 12(10):993–1001, Oct.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

[Lin et al.2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *NIPS*.

[Pan et al.2015] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2015. Hierarchical recurrent neural encoder for video representation with application to captioning. *CVPR*.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

[Rohrbach et al.2015] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A dataset for movie description. In *CVPR*.

[Sundermeyer et al.2010] M. Sundermeyer, R. Schluter, and H. Ney. 2010. Lstm neural networks for language modeling. In *INTERSPEECH*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[Torabi et al.2015] Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Using descriptive video services to create a large data source for video annotation research. *arXiv:1503.01070v1*.

[Venugopalan et al.2015a] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. 2015a. Sequence to sequence - video to text. *ICCV*.

[Venugopalan et al.2015b] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL*.

[Vinyals et al.2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. *CVPR*.

[Yao et al.2015] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. *ICCV*.

[Yu et al.2015] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2015. Video paragraph captioning using hierarchical recurrent neural networks. *CVPR*.

Representing Verbs with Rich Contexts: an Evaluation on Verb Similarity

Emmanuele Chersoni
Aix-Marseille University
emmanuelechersoni@gmail.com

Enrico Santus
The Hong Kong Polytechnic University
esantus@gmail.com

Alessandro Lenci
University of Pisa
alessandro.lenci@unipi.it

Philippe Blache
Aix-Marseille University
philippe.blache@univ-amu.fr

Chu-Ren Huang
The Hong Kong Polytechnic University
churen.huang@polyu.edu.hk

Abstract

Several studies on sentence processing suggest that the mental lexicon keeps track of the mutual expectations between words. Current DSMs, however, represent context words as separate features, thereby loosing important information for word expectations, such as word interrelations. In this paper, we present a DSM that addresses this issue by defining verb contexts as joint syntactic dependencies. We test our representation in a verb similarity task on two datasets, showing that joint contexts achieve performances comparable to single dependencies or even better. Moreover, they are able to overcome the data sparsity problem of joint feature spaces, in spite of the limited size of our training corpus.

1 Introduction

Distributional Semantic Models (DSMs) rely on the Distributional Hypothesis (Harris, 1954; Sahlgren, 2008), stating that words occurring in similar contexts have similar meanings. On such theoretical grounds, word co-occurrences extracted from corpora are used to build semantic representations in the form of vectors, which have become very popular in the NLP community. Proximity between word vectors is taken as an index of meaning similarity, and vector cosine is generally adopted to measure such proximity, even though other measures have been proposed (Weeds et al., 2004; Santus et al., 2016).

Most of DSMs adopt a bag-of-words approach, that is they turn a text span (i.e., a word window or a parsed sentence) into a set of words and they register separately the co-occurrence of each word with a given target. The problem with this approach is that valuable information concerning word interrelations in a context gets lost, because words co-occurring with a target are treated as independent features. This is why works like Ruiz-Casado et al. (2005), Agirre et al. (2009) and Melamud et al. (2014) proposed to introduce richer contexts in distributional spaces, by using entire word windows as features. These richer contexts proved to be helpful to semantically represent verbs, which are characterized by highly context-sensitive meanings, and complex argument structures. In fact, two verbs may share independent words as features despite being very dissimilar from the semantic point of view. For instance *kill* and *heal* share the same object nouns in *The doctor healed the patient* and the *The poison killed the patient*, but are highly different if we consider their joint dependencies as a single context. Nonetheless, richer contexts like these suffer from data sparsity, therefore requiring either larger corpora or complex smoothing processes.

In this paper, we propose a syntactically savvy notion of **joint contexts**. To test our representation, we implement several DSMs and we evaluate them in a verb similarity task on two datasets. The results show that, even using a relatively small corpus, our syntactic joint contexts are robust with respect to

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1967–1972,
Austin, Texas, November 1-5, 2016.

data sparseness and perform similarly or better than single dependencies in a wider range of parameter settings.

The paper is organized as follows. In Section 2, we provide psycholinguistic and computational background for this research, describing recent models based on word windows. In Section 3, we describe our reinterpretation of joint contexts with syntactic dependencies. Evaluation settings and results are presented in Section 4.

2 Related Work

A number of studies in sentence processing suggests that verbs activate expectations on their typical argument nouns and vice versa (McRae et al., 1998; McRae et al., 2005) and nouns do the same with other nouns occurring as co-arguments in the same events (Hare et al., 2009; Bicknell et al., 2010). Experimental subjects seem to exploit a rich event knowledge to activate or inhibit dynamically the representations of the potential arguments. This phenomenon, generally referred to as *thematic fit* (McRae et al., 1998), supports the idea of a mental lexicon arranged as a web of mutual expectations.

Some past works in computational linguistics (Baroni and Lenci, 2010; Lenci, 2011; Sayeed and Demberg, 2014; Greenberg et al., 2015) modeled thematic fit estimations by means of dependency-based or of thematic roles-based DSMs. However, these semantic spaces are built similarly to traditional DSMs as they split verb arguments into separate vector dimensions. By using syntactic-semantic links, they encode the relation between an event and each of its participants, but they do not encode directly the relation between participants co-occurring in the same event.

Another trend of studies in the NLP community aimed at the introduction of richer contextual features in DSMs, mostly based on word windows. The first example was the composite-feature model by Ruiz-Casado et al. (2005), who extracted word windows through a Web Search engine. A composite feature for the target word *watches* is *Alicia always ____ romantic movies*, extracted from the sentence *I heard that Alicia always watches romantic movies with Antony* (the placeholder represents the target position). Thanks to this approach, Ruiz-Casado and colleagues achieved 82.50 in the TOEFL synonym detection test, outperforming Latent Semantic Analysis (LSA; see Landauer et al. (1998)) and several other methods.

Agirre et al. (2009) adopted an analogous approach, relying on a huge learning corpus (1.6 Teraword) to build composite-feature vectors. Their model outperformed a traditional DSM on the similarity subset of the WordSim-353 test set (Finkelstein et al., 2001).

Melamud et al. (2014) introduced a probabilistic similarity scheme for modeling the so-called joint context. By making use of the Kneser-Ney language model (Kneser and Ney, 1995) and of a probabilistic distributional measure, they were able to overcome data sparsity, outperforming a wide variety of DSMs on two similarity tasks, evaluated on VerbSim (Yang and Powers, 2006) and on a set of 1,000 verbs extracted from WordNet (Fellbaum, 1998). On the basis of their results, the authors claimed that composite-feature models are particularly advantageous for measuring verb similarity.

3 Syntactic joint contexts

A joint context, as defined in Melamud et al. (2014), is a word window of order n around a target word. The target is replaced by a placeholder, and the value of the feature for a word w is the probability of w to fill the placeholder position. Assuming n=3, a word like *love* would be represented by a collection of contexts such as *the new students ____ the school campus*. Such representation introduces data sparseness, which has been addressed by previous studies either by adopting huge corpora or by relying on n-gram language models to approximate the probabilities of long sequences of words.

However, features based on word windows do not guarantee to include all the most salient event participants. Moreover, they could include unrelated words, also differentiating contexts describing the same event (e.g. consider *Luis ____ the red ball* and *Luis ____ the blue ball*).

For these reasons, we introduce the notion of *syntactic joint contexts*, further abstracting from linear word windows by using dependencies. Each feature of the word vector, in our view, should correspond to a typical verb-argument combination, as an approx-

imation to our knowledge about typical event participants. In the present study, we are focusing on verbs because verb meaning is highly context sensitive and include information about complex argument configurations. Therefore, verb representation should benefit more from the introduction of joint features (Melamud et al., 2014).

The procedure for defining of our representations is the following:

- we extract a list of verb-argument dependencies from a parsed corpus, and for each target verb we extract all the direct dependencies from the sentence of occurrence. For instance, in *Finally, the dictator acknowledged his failure*, we will have: target = 'acknowledge-v'; subject = 'dictator-n'; and object = 'failure-n'.

- for each sentence, we generate a joint context feature by joining all the dependencies for the grammatical relations of interest. From the example above, we would generate the feature *dictator-n.subj+____+failure-n.obj*.

For our experiments, the grammatical relations that we used are *subject*, *object* and *complement*, where *complement* is a generic relation grouping together all dependencies introduced by a preposition. Our distributional representation for a target word is a vector of syntatic joint contexts. For instance, the word vector for the verb *to begin* would include features like {*jury-n.subj+____+deliberation-n.obj*, *operation-n.subj+____+on-i_thursday-n.comp*, *recruit-n.subj+____+training-n.obj+on-i_street-n.comp* ...}. The value of each joint feature will be the frequency of occurrence of the target verb with the corresponding argument combination, possibly weighted by some statistical association measure.

4 Evaluation

4.1 Corpus and DSMs

We trained our DSMs on the RCV1 corpus, which contains approximately 150 million words (Lewis et al., 2004). The corpus was tagged with the tagger described in Dell'Orletta (2009) and dependency-parsed with DeSR (Attardi et al., 2009). RCV1 was chosen for two reasons: i) to show that our joint context-based representation can deal with data sparseness even with a training corpus of limited size; ii) to allow a comparison with the results reported by Melamud et al. (2014).

All DSMs adopt Positive Pointwise Mutual Information (PPMI; Church and Hanks (1990)) as a context weighting scheme and vary according to three main parameters: i) type of contexts; ii) number of dimensions; iii) application of Singular Value Decomposition (SVD; see Landauer et al. (1998)).

For what concerns the first parameter, we developed three types of DSMs: a) traditional bag-of-words DSMs, where the features are content words co-occurring with the target in a window of width 2; b) dependency-based DSMs, where the features are words in a direct dependency relation with the target; c) joint context-based DSMs, using the joint features described in the previous section. The second parameter refers instead to the number of contexts that have been used as vector dimensions. Several values were explored (i.e. 10K, 50K and 100K), selecting the contexts according to their frequency. Finally, the third parameter concerns the application of SVD to reduce the matrix. We report only the results for a number k of latent dimensions ranging from 200 to 400, since the performance drops significantly out of this interval.

4.2 Similarity Measures

As a similarity measure, we used vector cosine, which is by far the most popular in the existing literature (Turney et al., 2010). Melamud et al. (2014) have proposed the Probabilistic Distributional Similarity (PDS), based on the intuition that two words, w_1 and w_2, are similar if they are likely to occur in each other's contexts. PDS assigns a high similarity score when both $p(w_1|$ contexts of $w_2)$ and $p(w_2|$ contexts of $w_1)$ are high. We tried to test variations of this measure with our representation, but we were not able to achieve satisfying results. Therefore, we report here only the scores with the cosine.

4.3 Datasets

The DSMs are evaluated on two test sets: VerbSim (Yang and Powers, 2006) and the verb subset of SimLex-999 (Hill et al., 2015). The former includes 130 verb pairs, while the latter includes 222 verb pairs.

Both datasets are annotated with similarity judgements, so we measured the Spearman correlation between them and the scores assigned by the model. The VerbSim dataset allows for comparison with Melamud et al. (2014), since they also evaluated their model on this test set, achieving a Spearman correlation score of 0.616 and outperforming all the baseline methods.

The verb subset of SimLex-999, at the best of our knowledge, has never been used as a benchmark dataset for verb similarity. The SimLex dataset is known for being quite challenging: as reported by Hill et al. (2015), the average performances of similarity models on this dataset are much lower than on alternative benchmarks like WordSim (Finkelstein et al., 2001) and MEN (Bruni et al., 2014).

We exclude from the evaluation datasets all the target words occurring less than 100 times in our corpus. Consequently, we cover 107 pairs in the VerbSim dataset (82.3, the same of Melamud et al. (2014)) and 214 pairs in the SimLex verbs dataset (96.3).

4.4 Results

Table 1 reports the Spearman correlation scores for the vector cosine on our DSMs. At a glance, we can notice the discrepancy between the results obtained in the two datasets, as SimLex verbs confirms to be very difficult to model. We can also recognize a trend related to the number of contexts, as the performance tends to improve when more contexts are taken into account (with some exceptions). Single dependencies and joint contexts perform very similarly, and no one has a clear edge on the other. Both of them outperform the bag-of-words model on the VerbSim dataset by a nice margin, whereas the scores of all the model types are pretty much the same on SimLex verbs. Finally, it is noteworthy that the score obtained on VerbSim by the joint context model with 100K dimensions goes very close to the result reported by Melamud et al. (2014) (0.616).

Table 2 and Table 3 report the results of the models with SVD reduction. Independently of the number of dimensions k, the joint contexts almost always outperform the other model types. Overall, the performance of the joint contexts seems to be more stable across several parameter configurations, whereas bag-of-words and single dependencies are subject to bigger drops. Exceptions can be noticed only for the VerbSim dataset, and only with a low number of dimensions. Finally, the correlation coefficients for the two datasets seem to follow different trends, as the models with a higher number of contexts perform better on SimLex verbs, while the opposite is true for the VerbSim dataset.

On the VerbSim dataset, both single dependencies and joint contexts have again a clear advantage over bag-of-words representations Although they achieve a similar performance with 10K contexts, the correlation scores of the former decrease more quickly as the number of contexts increases, while the latter are more stable. Moreover, joint contexts are able to outperform single dependencies.
On SimLex verbs, all the models are very close and – differently from the previous dataset – the higher-dimensional DSMs are the better performing ones. Though differences are not statistically significant, joint context are able to achieve top scores over the other models.[1]

More in general, the best results are obtained with SVD reduction and k=200. The joint context-based DSM with 10K dimensions and k = 200 achieves 0.65, which is above the result of Melamud et al. (2014), although the difference between the two correlation scores is not significant. As for SimLex verbs, the best result (0.283) is obtained by the joint context DSM with 100K dimensions and k = 200.

Model	VerbSim	SimLex verbs
Bag-of-Words-10K	0.385	0.085
Single - 10k	0.561	0.090
Joint - 10k	0.568	0.105
Bag-of-Words-50K	0.478	0.095
Single - 50k	0.592	0.115
Joint - 50k	0.592	0.105
Bag-of-Words-100K	0.488	0.114
Single - 100k	0.587	**0.132**
Joint - 100k	**0.607**	0.114

Table 1: Spearman correlation scores for VerbSim and for the verb subset of SimLex-999. Each model is identified by the type and by the number of features of the semantic space.

[1] p-values computed with Fisher's r-to-z transformation comparing correlation coefficients between the joint context-DSMs and the other models on the same parameter settings.

Model	k = 200	k = 300	k = 400
Bag-of-Words-10K	0.457	0.445	0.483
Single - 10k	0.623	0.647	0.641
Joint - 10k	**0.650**	0.636	0.635
Bag-of-Words-50K	0.44	0.453	0.407
Single - 50k	0.492	0.486	0.534
Joint - 50k	0.571	0.591	0.613
Bag-of-Words-100K	0.335	0.324	0.322
Single - 100k	0.431	0.413	0.456
Joint - 100k	0.495	0.518	0.507

Table 2: Spearman correlation scores for VerbSim, after the application of SVD with different values of *k*.

Model	k = 200	k = 300	k = 400
Bag-of-Words-10K	0.127	0.113	0.111
Single - 10k	0.168	0.172	0.165
Joint - 10k	0.190	0.177	0.181
Bag-of-Words-50K	0.196	0.191	0.21
Single - 50k	0.218	0.228	0.222
Joint - 50k	0.256	0.250	0.227
Bag-of-Words-100K	0.222	0.18	0.16
Single - 100k	0.225	0.218	0.199
Joint - 100k	**0.283**	0.256	0.222

Table 3: Spearman correlation scores for the verb subset of SimLex-999, after the application of SVD with different values of *k*.

4.5 Conclusions

In this paper, we have presented our proposal for a new type of vector representation based on joint features, which should emulate more closely the general knowledge about event participants that seems to be the organizing principle of our mental lexicon. A core issue of previous studies was the data sparseness challenge, and we coped with it by means of a more abstract, syntactic notion of joint context.

The models using joint dependencies were able at least to perform comparably to traditional, dependency-based DSMs. In our experiments, they even achieved the best correlation scores across several parameter settings, especially after the application of SVD. We want to emphasize that previous works such as Agirre et al. (2009) already showed that large word windows can have a higher discriminative power than indipendent features, but they did it by using a huge training corpus. In our study, joint context-based representations derived from a small corpus such as RCV1 are already showing competitive performances. This result strengthens our belief that dependencies are a possible solution for the data sparsity problem of joint feature spaces.

We also believe that verb similarity might not be the best task to show the usefulness of joint contexts for semantic representation. The main goal of the present paper was to show that joint contexts are a viable option to exploit the full potential of distributional information. Our successful tests on verb similarity prove that syntactic joint contexts do not suffer of data sparsity and are also able to beat other types of representations based on independent word features. Moreover, syntactic joint contexts are much simpler and more competitive with respect to window-based ones.

The good performance in the verb similarity task motivates us to further test syntactic joint contexts on a larger range of tasks, such as word sense disambiguation, textual entailment and classification of semantic relations, so that they can unleash their full potential. Moreover, our proposal opens interesting perspectives for computational psycholinguistics, especially for modeling those semantic phenomena that are inherently related to the activation of event knowledge (e.g. thematic fit).

Acknowledgments

This paper is partially supported by HK PhD Fellowship Scheme, under PF12-13656. Emmanuele Chersoni's research is funded by a grant of the University Foundation A*MIDEX.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of the 2009 conference of the NAACL-HLT*, pages 19–27. Association for Computational Linguistics.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. In *Proceedings of EVALITA*, 9.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Klinton Bicknell, Jeffrey L Elman, Mary Hare, Ken McRae, and Marta Kutas. 2010. Effects of event

knowledge in processing verbal arguments. *Journal of Memory and Language*, 63(4):489–505.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47).

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Felice Dell'Orletta. 2009. Ensemble system for part-of-speech tagging. In *Proceedings of EVALITA*, 9.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Clayton Greenberg, Asad Sayeed, and Vera Demberg. 2015. Improving unsupervised vector-space thematic fit evaluation via role-filler prototype clustering. In *Proceedings of the 2015 conference of the NAACL-HLT, Denver, USA*.

Mary Hare, Michael Jones, Caroline Thomson, Sarah Kelly, and Ken McRae. 2009. Activating event knowledge. *Cognition*, 111(2):151–167.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Alessandro Lenci. 2011. Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 58–66. Association for Computational Linguistics.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

Ken McRae, Michael J Spivey-Knowlton, and Michael K Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38(3):283–312.

Ken McRae, Mary Hare, Jeffrey L Elman, and Todd Ferretti. 2005. A basis for generating expectancies for verbs from nouns. *Memory & Cognition*, 33(7):1174–1184.

Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. 2014. Probabilistic modeling of joint-context in distributional similarity. In *CoNLL*, pages 181–190.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of RANLP*, pages 1–7.

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54.

Enrico Santus, Emmanuele Chersoni, Alessandro Lenci, Chu-Ren Huang, and Philippe Blache. 2016. Testing APSyn against Vector Cosine on Similarity Estimation. In *Proceedings of the Pacific Asia Conference on Language, Information and Computing (PACLIC 30)*.

Asad Sayeed and Vera Demberg. 2014. Combining unsupervised syntactic and semantic models of thematic fit. In *Proceedings of the first Italian Conference on Computational Linguistics (CLiC-it 2014)*.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1015. Association for Computational Linguistics.

Dongqiang Yang and David MW Powers. 2006. *Verb similarity on the taxonomy of WordNet*. Masaryk University.

Speed-Accuracy Tradeoffs in Tagging with Variable-Order CRFs and Structured Sparsity

Tim Vieira* and **Ryan Cotterell*** and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
{timv,ryan.cotterell,jason}@cs.jhu.edu

Abstract

We propose a method for learning the structure of variable-order CRFs, a more flexible variant of higher-order linear-chain CRFs. Variable-order CRFs achieve faster inference by including features for only some of the tag n-grams. Our learning method discovers the useful higher-order features at the same time as it trains their weights, by maximizing an objective that combines log-likelihood with a structured-sparsity regularizer. An active-set outer loop allows the feature set to grow as far as needed. On part-of-speech tagging in 5 randomly chosen languages from the Universal Dependencies dataset, our method of shrinking the model achieved a 2–6x speedup over a baseline, with no significant drop in accuracy.

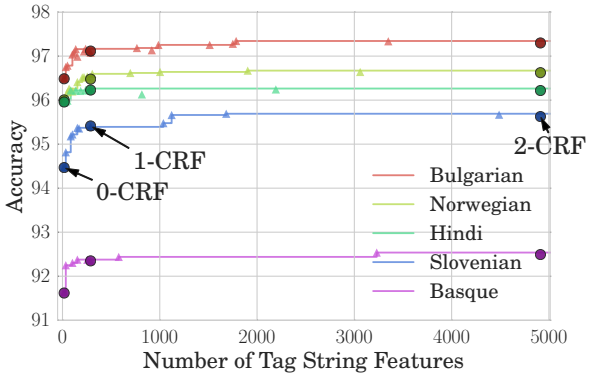


Figure 1: Speed-accuracy tradeoff curves on test data for the 5 languages. Large dark circles represent the k-CRFs of ascending orders along x-axis (marked on for Slovenian). Smaller triangles each represent a VoCRF discovered by sweeping the speed parameters γ. We find faster models at similar accuracy to the best k-CRFs (§5).

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are a convenient formalism for sequence labeling tasks common in NLP. A CRF defines a feature-rich conditional distribution over tag sequences (output) given an observed word sequence (input).

The key advantage of the CRF framework is the flexibility to consider arbitrary features of the input, as well as enough features over the output structure to encourage it to be well-formed and consistent. However, inference in CRFs is fast only if the features over the output structure are limited. For example, an order-k CRF (or "k-CRF" for short, with $k > 1$ being "higher-order") allows expressive features over a window of $k+1$ adjacent tags (as well as the input), and then inference takes time $\mathcal{O}(n \cdot |Y|^{k+1})$, where Y is the set of tags and n is the length of the input.

How large does k need to be? Typically $k = 2$ works well, with big gains from $0 \rightarrow 1$ and modest gains from $1 \rightarrow 2$ (Fig. 1). Small k may be sufficient when there is enough training data to allow the model to attend to many fine-grained features of the input (Toutanova et al., 2003; Liang et al., 2008). For example, when predicting POS tags in morphologically-rich languages, certain words are easily tagged based on their spelling without considering the context ($k = 0$). In fact, such languages tend to have a more free word order, making tag context less useful.

We investigate a hybrid approach that gives the accuracy of higher-order models while reducing runtime. We build on variable-order CRFs (Ye et al., 2009) (VoCRF), which support features on tag subsequences of mixed orders. Since only modest gains are obtained from moving to higher-order models, we posit that only a small fraction of the higher-order features are necessary. We introduce a hyperparameter γ that *discourages* the model from using many higher-order features (= faster inference) and a hyperparameter λ that *encourages* generalization. Thus, sweeping a range of values for γ and λ gives rise to a

*Equal contribution

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1973–1978,
Austin, Texas, November 1-5, 2016.

number of operating points along the speed-accuracy curve (triangle points in Fig. 1).

We present three contributions: (1) A simplified exposition of VoCRFs, including an algorithm for computing gradients that is asymptotically more efficient than prior art (Cuong et al., 2014). (2) We develop a structure learning algorithm for *discovering* the essential set of higher-order dependencies so that inference is fast *and* accurate. (3) We investigate the effectiveness of our approach on POS tagging in five diverse languages. We find that the amount of required context for accurate prediction is highly language-dependent. In all languages, however, our approach meets the accuracy of fixed-order models at a fraction of the runtime.

2 Variable-Order CRFs

An order-k CRF (k-CRF, for short) is a conditional probability distribution of the form

$$p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})} \exp\Big(\sum_{t=1}^{n+1} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, t, y_{t-k} \dots y_t)\Big)$$

where n is the length of the input $\boldsymbol{x}$, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the model parameter, and $\boldsymbol{f}$ is an arbitrary user-defined function that computes a vector in $\mathbb{R}^d$ of features of the tag substring $\boldsymbol{s} = y_{t-k} \dots y_t$ when it appears at position t of input $\boldsymbol{x}$. We define y_i to be a distinguished boundary tag # when $i \notin [1, n]$.

A variable-order CRF or VoCRF is a refinement of the k-CRF, in which $\boldsymbol{f}$ may not always depend on all $k+1$ of the tags that it has access to. The features of a particular tag substring $\boldsymbol{s}$ may sometimes be determined by a shorter suffix of $\boldsymbol{s}$.

To be precise, a VoCRF specifies a finite set $\mathcal{W} \subset Y^*$ that is sufficient for feature computation (where Y^* denotes the set of all tag sequences).[1] The VoCRF's featurization function $\boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{s})$ is then defined as $\boldsymbol{f}'(\boldsymbol{x}, t, \boldsymbol{w}(\boldsymbol{s}))$ where $\boldsymbol{f}'$ can be any function and $\boldsymbol{w}(\boldsymbol{s}) \in Y^*$ is the longest suffix of $\boldsymbol{s}$ that appears in $\mathcal{W}$ (or ε if none exists). The full power of a k-CRF can be obtained by specifying $\mathcal{W} = Y^{k+1}$, but smaller $\mathcal{W}$ will in general allow speedups.

To support our algorithms, we define $\overline{\mathcal{W}}$ to be the closure of $\mathcal{W}$ under prefixes and last-character substitution. Formally, $\overline{\mathcal{W}}$ is the smallest nonempty superset of $\mathcal{W}$ such that if $\boldsymbol{h}y \in \overline{\mathcal{W}}$ for some $\boldsymbol{h} \in Y^*$ and $y \in Y$, then $\boldsymbol{h} \in \overline{\mathcal{W}}$ and also $\boldsymbol{h}y' \in \overline{\mathcal{W}}$ for all $y' \in Y$. This implies that we can factor $\overline{\mathcal{W}}$ as $\mathcal{H} \times Y$, where $\mathcal{H} \subset Y^*$ is called the set of *histories*.

We now define $\textsc{Next}(h, y)$ to return the longest suffix of hy that is in $\mathcal{H}$ (which may be hy itself, or even ε). We may regard NEXT as the transition function of a deterministic finite-state automaton (DFA) with state set $\mathcal{H}$ and alphabet Y. If this DFA is used to read any tag sequence $\boldsymbol{y} \in Y^*$, then the arc that reads y_t comes from a state $\boldsymbol{h}$ such that $\boldsymbol{h}y_t$ is the longest suffix of $\boldsymbol{s} = y_{t-k} \dots y_t$ that appears in $\overline{\mathcal{W}}$—and thus $\boldsymbol{w}(\boldsymbol{h}y_t) = \boldsymbol{w}(\boldsymbol{s}) \in \mathcal{W}$ and provides sufficient information to compute $\boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{s})$.[2]

For a given $\boldsymbol{x}$ of length n and given parameters $\boldsymbol{\theta}$, the log-normalizer $\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})$—which will be needed to compute the log-probability in eq. (1) below—can be found in time $\mathcal{O}(|\overline{\mathcal{W}}|\, n)$ by dynamic programming. Concise pseudocode is in Alg. 1. In effect, this

Algorithm 1 FORWARD: Compute $\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})$.

$\alpha(\cdot, \cdot) = 0$; $\alpha(0, \#) = 1$ ▷ *initialization*
for $t = 1$ **to** $n+1$:
  if $t = n+1$ **then** $Y_t = \{\#\}$ **else** $y_t = Y \backslash \{\#\}$
  for $\boldsymbol{h} \in \mathcal{H}, y_t \in Y_t$:
    $\boldsymbol{h}' = \textsc{Next}(\boldsymbol{h}, y_t)$
    $z = \exp\big(\boldsymbol{\theta}^\top \boldsymbol{f}'(\boldsymbol{x}, t, \boldsymbol{w}(\boldsymbol{h}y_t))\big)$
    $\alpha(t, \boldsymbol{h}')$ += $\alpha(t-1, \boldsymbol{h}) \cdot z$
$Z = \sum_{\boldsymbol{h} \in \mathcal{H}} \alpha(n+1, \boldsymbol{h})$ ▷ *sum over final states*
return $\log Z$

Algorithm 2 GRADIENT: Compute $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\boldsymbol{x})$.

$\beta(\cdot, \cdot) = 0$; $\boldsymbol{\Delta} = \mathbf{0}$
$\beta(n+1, \boldsymbol{h}) = 1$ for all $\boldsymbol{h} \in \mathcal{H}$ ▷ *initialization*
for $t = n+1$ **downto** 1 :
  for $\boldsymbol{h} \in \mathcal{H}, y_t \in Y_t$:
    $\boldsymbol{h}' = \textsc{Next}(\boldsymbol{h}, y_t)$
    $z = \exp\big(\boldsymbol{\theta}^\top \boldsymbol{f}'(\boldsymbol{x}, t, \boldsymbol{w}(\boldsymbol{h}y_t))\big)$
    $\boldsymbol{\Delta}$ += $\boldsymbol{f}'(\boldsymbol{x}, t, \boldsymbol{w}(\boldsymbol{h}y_t)) \cdot \alpha(t-1, \boldsymbol{h}) \cdot z \cdot \beta(t, \boldsymbol{h}')$
    $\beta(t-1, \boldsymbol{h})$ += $z \cdot \beta(t, \boldsymbol{h}')$
return $\boldsymbol{\Delta}/Z$

[1] The constructions given in this section assume that $\mathcal{W}$ does not contain ε nor any sequence having ## as a proper prefix.

[2] Our DFA construction is essentially that of Cotterell and Eisner (2015, Appendix B.5). However, Appendix B of that paper also gives a construction that obtains an even smaller DFA by using failure arcs (Allauzen et al., 2003), which remove the requirement that $\overline{\mathcal{W}}$ be closed under last-character substitution. This would yield a further speedup to our Alg. 1 (replacing it with the efficient backward algorithm in footnote 16 of that paper) and similarly to our Alg. 2 (by differentiating the new Alg. 1).

runs the forward algorithm on the lattice of taggings given by length-n paths through the DFA.

For finding the parameters $\boldsymbol{\theta}$ that minimize eq. (1) below, we want the gradient $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\boldsymbol{x})$. By applying algorithmic differentiation to Alg. 1, we obtain Alg. 2, which uses back-propagation to compute the gradient (asymptotically) as fast as Alg. 1 and $|\mathcal{H}|$ times faster than Cuong et al. (2014)'s algorithm—a significant speedup since $|\mathcal{H}|$ is often quite large (up to 300 in our experiments). Algs. 1–2 together effectively run the forward-backward algorithm on the lattice of taggings.[3]

It is straightforward to modify Alg. 1 to obtain a Viterbi decoder that finds the most-likely tag sequence under $p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})$. It is also straightforward to modify Alg. 2 to compute the marginal probabilities of tag substrings occurring at particular positions.

3 Structured Sparsity and Active Sets

We begin with a k-CRF model whose feature vector $\boldsymbol{f}(\boldsymbol{x}, t, y_{t-k} \dots y_t)$ is partitioned into non-stationary local features $\boldsymbol{f}^{(1)}(\boldsymbol{x}, t, y_t)$ and stationary higher-order features $\boldsymbol{f}^{(2)}(y_{t-k} \dots y_t)$. Specifically, $\boldsymbol{f}^{(2)}$ includes an indicator feature for each tag string $\boldsymbol{w} \in Y^*$ with $1 \leq |\boldsymbol{w}| \leq k+1$, where $f^{(2)}_{\boldsymbol{w}}(y_{t-k} \dots y_t)$ is 1 if $\boldsymbol{w}$ is a suffix of $y_{t-k} \dots y_t$ and is 0 otherwise.[4]

To obtain the advantages of a VoCRF, we merely have to choose a *sparse* weight vector $\boldsymbol{\theta}$. The set $\mathcal{W}$ can then be defined to be the set of strings in Y^* whose features have nonzero weight. Prior work (Cuong et al., 2014) has left the construction of $\mathcal{W}$ to domain experts or "one size fits all" strategies (e.g., k-CRF). Our goal is to choose $\boldsymbol{\theta}$—and thus $\mathcal{W}$—so that inference is accurate *and* fast.

Our approach is to modify the usual L_2-regularized log-likelihood training criterion with a carefully defined runtime penalty scaled by a parameter γ to balance competing objectives: likelihood on the data $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^m$ vs. efficiency (small $\mathcal{W}$).

$$-\sum_{i=1}^{m} \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)})}_{\text{loss}} + \underbrace{\lambda ||\boldsymbol{\theta}||_2^2}_{\text{generalization}} + \underbrace{\gamma \mathcal{R}(\boldsymbol{\theta})}_{\text{runtime}} \quad (1)$$

Recall that the runtime of inference on a given sentence is proportional to the size of $\overline{\mathcal{W}}$, the *closure* of $\mathcal{W}$ under prefixes and last-character replacement. (Any tag strings in $\overline{\mathcal{W}} \backslash \mathcal{W}$ can get nonzero weight without increasing runtime.) Thus, $\mathcal{R}(\boldsymbol{\theta})$ would ideally measure $|\overline{\mathcal{W}}|$, or proportionately, $|\mathcal{H}|$. Experimentally, we find that $|\overline{\mathcal{W}}|$ has $> 99\%$ Pearson correlation with wallclock time, making it an excellent proxy for wallclock time while being more replicable.

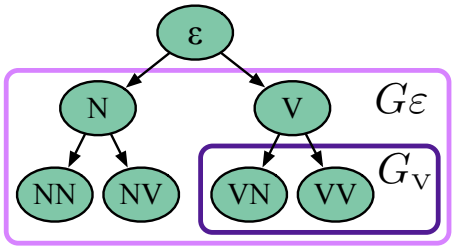


Figure 2: A visual depiction of the tree-structured group lasso penalty. Each node represents a tag string feature. The group indexed by a node's tag string is defined as the *set* of features that are *proper descendants* of the node. For example, the lavender box indicates the largest group G_ε and the aubergine box indicates a smaller group G_{V}. To avoid clutter, not all groups are marked.

We relax this regularizer to a convex function—a tree-structured *group lasso* objective (Yuan and Lin, 2006; Nelakanti et al., 2013). For each string $\boldsymbol{h} \in Y^*$, we have a group $G_{\boldsymbol{h}}$ consisting of the indicator features (in $\boldsymbol{f}^{(2)}$) for all strings $w \in \overline{\mathcal{W}}$ that have $\boldsymbol{h}$ as a proper prefix. Fig. 2 gives a visual depiction. We now define $\mathcal{R}(\boldsymbol{\theta}) = \sum_{\boldsymbol{h} \in Y^*} ||\boldsymbol{\theta}_{G_{\boldsymbol{h}}}||_2$. This penalty encourages each group of weights to remain all at zero (thereby conserving runtime, in our setting, because it means that $\boldsymbol{h}$ does not need to be added to $\mathcal{H}$). Once a single weight in a group becomes nonzero, the "initial inertia" induced by the group lasso penalty is overcome, and other features in the group can be more cheaply adjusted away from zero.

Although eq. (1) is now convex, directly optimizing it would be expensive for large k, since $\boldsymbol{\theta}$ then contains very many parameters. We thus use a heuristic optimization algorithm, the *active set* method (Schmidt, 2010), which starts with a low-dimensional $\boldsymbol{\theta}$ and *incrementally* adds features to the model. This also frees us from needing to specify a limit k. Rather, $\mathcal{W}$ grows until further extensions are unhelpful, and then implicitly $k = \max_{\boldsymbol{w} \in \mathcal{W}} |\boldsymbol{w}| - 1$.

The method defines $\boldsymbol{f}^{(2)}$ to include indicator features for all tag sequences $\boldsymbol{w}$ in an *active set* $\mathcal{W}_{\text{active}}$. Thus, $\boldsymbol{\theta}^{(2)}$ is always a vector of $|\mathcal{W}_{\text{active}}|$ real numbers. Initially, we take $\mathcal{W}_{\text{active}} = Y$ and $\boldsymbol{\theta} = \mathbf{0}$. At each

[3] Eisner (2016) explains the connection between algorithmic differentiation and the forward-backward algorithm.

[4] Extensions to richer sets of higher-order features are possible, such as conjunctions with properties of the words at position t.

active set iteration, we fully optimize eq. (1) to obtain a sparse $\boldsymbol{\theta}$ and a set $\mathcal{W} = \{\boldsymbol{w} \in \mathcal{W}_{\text{active}} \mid \boldsymbol{\theta}^{(2)}_{\boldsymbol{w}} \neq 0\}$ of features that are known to be "useful."[5] We then update $\mathcal{W}_{\text{active}}$ to $\{\boldsymbol{w}y \mid \boldsymbol{w} \in \mathcal{W}, y \in Y\}$, so that it includes single-tag extensions of these useful features; this expands $\boldsymbol{\theta}$ to consider additional features that plausibly might prove useful. Finally, we complete the iteration by updating $\mathcal{W}_{\text{active}}$ to its closure $\overline{\mathcal{W}_{\text{active}}}$, simply because this further expansion of the feature set will not slow down our algorithms. When eq. (1) is re-optimized at the next iteration, some of these newly added features in $\mathcal{W}_{\text{active}}$ may acquire nonzero weights and thus enter $\mathcal{W}$, allowing further extensions. We can halt once $\mathcal{W}$ no longer changes.

As a final step, we follow common practice by running "debiasing" (Martins et al., 2011a), where we fix our $\boldsymbol{f}^{(2)}$ feature set to be given by the final $\overline{\mathcal{W}}$, and retrain $\boldsymbol{\theta}$ without the group lasso penalty term.

In practice, we optimized eq. (1) using the online proximal gradient algorithm SPOM (Martins et al., 2011b) and Adagrad (Duchi et al., 2011) with $\eta = 0.01$ and 15 inner epochs. We limited to 3 active set iterations, and as a result, our final $\overline{\mathcal{W}}$ contained at most tag trigrams.

4 Related Work

Our paper can be seen as transferring methods of Cotterell and Eisner (2015) to the CRF setting. They too used tree-structured group lasso and active set to select variable-order n-gram features $\mathcal{W}$ for globally-normalized sequence models (in their case, to rapidly and accurately approximate beliefs during message-passing inference). Similarly, Nelakanti et al. (2013) used tree-structured group lasso to regularize a variable-order language model (though their focus was *training* speed). Here we apply these techniques to *conditional* models for tagging.

Our work directly builds on the variable-order CRF of Cuong et al. (2014), with a speedup in Alg. 2, but our approach also learns the VoCRF structure. Our method is also related to the *generative* variable-order tagger of Schütze and Singer (1994).

Our static feature selection chooses a single model that permits fast exact marginal inference, similar to learning a low-treewidth graphical model (Bach and Jordan, 2001; Elidan and Gould, 2008). This contrasts with recent papers that learn to do approximate 1-best inference using a sequence of models, whether by dynamic feature selection within a greedy inference algorithm (Strubell et al., 2015), or by gradually increasing the feature set of a 1-best global inference algorithm and pruning its hypothesis space after each increase (Weiss and Taskar, 2010; He et al., 2013).

Schmidt (2010) explores the use of group lasso penalties and the active set method for learning the structure of a graphical model, but does not consider learning repeated structures (in our setting, $\overline{\mathcal{W}}$ defines a structure that is reused at each position). Steinhardt and Liang (2015) jointly modeled the amount of context to use in a variable-order model that dynamically determines how much context to use in a beam search decoder.

5 Experiments[6]

Data: We conduct experiments on multilingual POS tagging. The task is to label each word in a sentence with one of $|Y| = 17$ labels. We train on five typologically-diverse languages from the Universal Dependencies (UD) corpora (Petrov et al., 2012): Basque, Bulgarian, Hindi, Norwegian and Slovenian. For each language, we start with the original train / dev / test split in the UD dataset, then move random sentences from train into dev until the dev set has 3000 sentences. This ensures more stable hyperparameter tuning. We use these new splits below.

Eval: We train models with $(\lambda, \gamma) \in \{10^{-4} \cdot m, 10^{-3} \cdot m, 10^{-2} \cdot m\} \times \{0, 0.1 \cdot m, 0.2 \cdot m, \ldots, m\}$, where m is the number of training sentences. To tag a dev or test sentence, we choose its most probable tag sequence. For each of several model sizes, Table 1 selects the model of that size that achieved the highest per-token tagging accuracy on the dev set, and reports that model's accuracy on the test set.

Features: Recall from §3 that our features include non-stationary zeroth-order features $\boldsymbol{f}^{(1)}$ as well as the stationary features based on $\mathcal{W}$. For $\boldsymbol{f}^{(1)}(\boldsymbol{x}, t, y_t)$ we consider the following language-agnostic properties of $(\boldsymbol{x}, t)$:

- The identities of the tokens $x_{t-3}, ..., x_{t+3}$, and the token bigrams (x_{t+1}, x_t), (x_t, x_{t-1}),

[5] Each gradient computation in this inner optimization takes time $\mathcal{O}(|\overline{\mathcal{W}_{\text{active}}}|\, n)$, which is especially fast at early iterations.

[6] Code and data are available at the following URLs:
`http://github.com/timvieira/vocrf`
`http://universaldependencies.org`

	k-CRF ($\lvert\overline{\mathcal{W}}\rvert = 17^{k+1}$)			VoCRF at different model sizes $\lvert\overline{\mathcal{W}}\rvert$ (which is proportional to runtime)									
	0 (17)	1 (289)	2 (4913)	≤ 34	≤ 85	≤ 170	≤ 340	≤ 850	≤ 1700	≤ 2550	≤ 3400	≤ 4250	≤ 5100
Ba	$91.61^{1,2}$	$\underline{92.35}^{0}$	92.49^{0}	$92.25^{0,2}$	$92.25^{0,2}$	$\underline{92.38}^{0}$	92.34^{0}	92.44^{0}	92.44^{0}	92.44^{0}	$\mathbf{92.54}^{0}$	92.54^{0}	92.54^{0}
Bu	$96.48^{1,2}$	$97.11^{0,2}$	$\underline{97.29}^{0,1}$	$96.75^{0,1,2}$	$96.78^{0,1,2}$	$96.99^{0,1,2}$	$97.08^{0,2}$	$\underline{97.18}^{0,1}$	$97.25^{0,1}$	$\mathbf{97.34}^{0,1}$	$97.34^{0,1}$	$97.34^{0,1}$	$97.34^{0,1}$
Hi	$95.96^{1,2}$	$\underline{96.22}^{0}$	96.21^{0}	$95.97^{1,2}$	$\underline{96.22}^{0}$	96.22^{0}	96.26^{0}	96.13^{0}	96.13^{0}	$\mathbf{96.24}^{0}$	96.24^{0}	96.24^{0}	96.24^{0}
No	$96.00^{1,2}$	$\underline{96.64}^{0}$	96.66^{0}	$96.07^{1,2}$	$96.26^{0,1,2}$	$\underline{96.41}^{0}$	96.60^{0}	96.62^{0}	96.64^{0}	$\mathbf{96.67}^{0}$	96.64^{0}	96.64^{0}	96.64^{0}
Sl	$94.46^{1,2}$	$95.41^{0,2}$	$\underline{95.62}^{0,1}$	$94.82^{1,2}$	$95.18^{0,2}$	$95.36^{0,2}$	$95.39^{0,2}$	$95.39^{0,2}$	$\mathbf{\underline{95.69}}^{0,1}$	$95.69^{0,1}$	$95.69^{0,1}$	$95.69^{0,1}$	$95.67^{0,1}$

Table 1: Part-of-speech tagging with Universal Tags: This table shows test results on 5 languages at different target runtimes. Each row's best results are in **boldface**, where ties in accuracy are broken in favor of faster models. Superscript k indicates that the accuracy is significantly different from the k-CRF (paired permutation test, $p < 0.05$) and this superscript is in blue/red if the accuracy is higher/lower than the k-CRF. In all cases, we find a VoCRF (underlined) that is about as accurate as the 2-CRF (i.e., not significantly less accurate) and far faster, since the 2-CRF has $\lvert\overline{\mathcal{W}}\rvert = 4913$. Fig. 1 plots the Pareto frontiers.

(x_{t-1}, x_{t+1}). We use special boundary symbols for tokens at positions beyond the start or end of the sentence.

- Prefixes and suffixes of x_t, up to 4 characters long, that occur ≥ 5 times in the training data.
- Indicators for whether x_t is all caps, is lowercase, or has a digit.
- Word shape of x_t, which maps the token string into the following character classes (uppercase, lowercase, number) with punctuation unmodified (e.g., VoCRF-like $\Rightarrow$ AaAAA-aaaa, $5,432.10 $\Rightarrow$ $8,888.88).

For efficiency, we hash these properties into 2^{22} bins. The $\boldsymbol{f}^{(1)}$ features are obtained by conjoining these bins with y_t (Weinberger et al., 2009): e.g., there is a feature that returns 0 unless $y_t = \textsf{NOUN}$, in which case it counts the number of bin 1234567's properties that $(\boldsymbol{x}, t)$ has. (The $\boldsymbol{f}^{(2)}$ features are not hashed.)

Results: Our results are presented in Fig. 1 and Table 1. We highlight two key points: (i) Across *all languages* we learned a tagger about as accurate as a 2-CRF, but much faster. (ii) The size of the set $\overline{\mathcal{W}}$ required is highly language-dependent. For many languages, learning a full k-CRF is wasteful; our method resolves this problem.

In each language, the fastest "good" VoCRF is rather faster than the fastest "good" k-CRF (where "good" means statistically indistinguishable from the 2-CRF). These two systems are underlined; the underlined VoCRF systems are smaller than the underlined k-CRF systems (for the 5 languages respectively) by factors of 1.9, 6.4, 3.4, 1.9, and 2.9. In every language, we learn a VoCRF with $\lvert\overline{\mathcal{W}}\rvert \leq 850$ that is not significantly worse than a 2-CRF with $\lvert\overline{\mathcal{W}}\rvert = 17^3 = 4913$.

We also notice an interesting language-dependent effect, whereby certain languages require a small number of tag strings in order to perform well. For example, Hindi has a competitive model that ignores the previous tag y_{t-1} unless it is in $\{\textsf{NOUN}, \textsf{VERB}, \textsf{ADP}, \textsf{PROPN}\}$: thus the stationary features are 17 unigrams plus 4×17 bigrams, for a total of $\lvert\overline{\mathcal{W}}\rvert = 85$. At the other extreme, the Slavic languages Slovenian and Bulgarian seem to require more expressive models over the tag space, remembering as many as 98 useful left-context histories (unigrams and bigrams) for the current tag. An interesting direction for future research would be to determine which morpho-syntactic properties of a language tend to increase the complexity of tagging.

6 Conclusion

We presented a structured sparsity approach for structure learning in VoCRFs, which achieves the accuracy of higher-order CRFs at a fraction of the runtime. Additionally, we derive an asymptotically faster algorithm for the gradients necessary to train a VoCRF than prior work. Our method provides an effective speed-accuracy tradeoff for POS tagging across five languages—confirming that significant speed-ups are possible with little-to-no loss in accuracy.

Acknowledgments: This material is based in part on research sponsored by DARPA under agreement number FA8750-13-2-0017 (DEFT program) and the National Science Foundation under Grant No. 1423276. The second author was funded by a DAAD Long-term research grant and an NDSEG fellowship.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 40–47.

F. R. Bach and M. I. Jordan. 2001. Thin junction trees. In *NIPS*, pages 569–576.

Ryan Cotterell and Jason Eisner. 2015. Penalized expectation propagation for graphical models over strings. In *NAACL-HLT*, pages 932–942.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *JMLR*, 15(1):981–1009.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the EMNLP 16 Workshop on Structured Prediction for NLP*, Austin, TX, November.

G. Elidan and S. Gould. 2008. Learning bounded treewidth Bayesian networks. In *NIPS*, pages 417–424.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *EMNLP*, pages 1455–1464.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *ICML*, pages 592–599.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011a. Structured sparsity in structured prediction. In *EMNLP*, pages 1500–1511.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A.T. Figueiredo. 2011b. Online learning of structured predictors with multiple kernels. In *AISTATS*, pages 507–515.

Anil Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach, and Guillaume Bouchard. 2013. Structured penalties for log-linear language models. In *EMNLP*, pages 233–243.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2012. A universal part-of-speech tagset. In *LREC*, pages 2089–2096.

Mark Schmidt. 2010. *Graphical Model Structure Learning with ℓ_1-Regularization*. Ph.D. thesis, University of British Columbias.

Hinrich Schütze and Yoram Singer. 1994. Part-of-speech tagging using a variable memory Markov model. In *ACL*, pages 181–187.

Jacob Steinhardt and Percy Liang. 2015. Reified context models. In *ICML*, pages 1043–1052.

Emma Strubell, Luke Vilnis, Kate Silverstein, and Andrew McCallum. 2015. Learning dynamic feature selection for fast sequential prediction. In *ACL*, pages 146–155.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *ACL*, pages 173–180.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning.

David J. Weiss and Benjamin Taskar. 2010. Structured prediction cascades. In *AISTATS*, pages 916–923.

Nan Ye, Wee S. Lee, Hai L. Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In *NIPS*, pages 2196–2204.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Learning Robust Representations of Text

Yitong Li **Trevor Cohn** **Timothy Baldwin**
Department of Computing and Information Systems
The University of Melbourne, Australia
yitongl4@student.unimelb.edu.au, {tcohn,tbaldwin}@unimelb.edu.au

Abstract

Deep neural networks have achieved remarkable results across many language processing tasks, however these methods are highly sensitive to noise and adversarial attacks. We present a regularization based method for limiting network sensitivity to its inputs, inspired by ideas from computer vision, thus learning models that are more robust. Empirical evaluation over a range of sentiment datasets with a convolutional neural network shows that, compared to a baseline model and the dropout method, our method achieves superior performance over noisy inputs and out-of-domain data.[1]

1 Introduction

Deep learning has achieved state-of-the-art results across a range of computer vision (Krizhevsky et al., 2012), speech recognition (Graves et al., 2013) and natural language processing tasks (Bahdanau et al., 2015; Kalchbrenner et al., 2014; Yih et al., 2014; Bitvai and Cohn, 2015). However, deep models are often overconfident for noisy test instances, making them susceptible to adversarial attacks (Nguyen et al., 2015; Tabacof and Valle, 2016). Goodfellow et al. (2014) argued that the primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature, due to neural models being intentionally designed to behave in a mostly linear manner to facilitate optimization. Fawzi et al. (2015) provided a theoretical framework for analyzing the robustness of classifiers to adversarial perturbations, and also showed linear models are usually not robust to adversarial noise.

In this work, we present a regularization method which makes deep learning models more robust to noise, inspired by Rifai et al. (2011). The intuition behind the approach is to stabilize predictions by minimizing the ability of features to perturb predictions, based on high-order derivatives. Rifai et al. (2011) introduced contractive auto-encoders based on similar ideas, using the Frobenius norm of the Jacobian matrix as a penalty term to extract robust features. Further, Gu and Rigazio (2014) introduced deep contractive networks, generalizing this idea to a feed-forward neural network. Also related, Martens (2010) investigated a second-order optimization method based on Hessian-free approach for training deep auto-encoders. Where our proposed approach differs is that we train models using first-order derivatives of the training loss as part of a regularization term, necessitating second-order derivatives for computing the gradient. We empirically demonstrate the effectiveness of the model over text corpora with increasing amounts of artificial masking noise, using a range of sentiment analysis datasets (Pang and Lee, 2008) with a convolutional neural network model (Kim, 2014). In this, we show that our method is superior to dropout (Srivastava et al., 2014) and a baseline method using MAP training.

2 Training for Robustness

Our method introduces a regularization term during training to ensure model robustness. We develop our approach based on a general class of parametric mod-

[1] Implementation available at https://github.com/lrank/Robust-Representation.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1979–1985,
Austin, Texas, November 1-5, 2016.

els, with the following structure. Let $\mathbf{x}$ be the input, which is a sequence of (discrete) words, represented by a fixed-size vector of continuous values, $\mathbf{h}$. A transfer function takes $\mathbf{h}$ as input and produces an output distribution, $\mathbf{y}_{\text{pred}}$. Training proceeds using stochastic gradient descent to minimize a loss function L, measuring the difference between $\mathbf{y}_{\text{pred}}$ and the truth $\mathbf{y}_{\text{true}}$.

The purpose of our work is to learn neural models which are more robust to strange or invalid inputs. When small perturbations are applied on $\mathbf{x}$, we want the prediction $\mathbf{y}_{\text{pred}}$ to remain stable. Text can be highly variable, allowing for the same information to be conveyed with different word choice, different syntactic structures, typographical errors, stylistic changes, etc. This is a particular problem in transfer learning scenarios such as domain adaptation, where the inputs in distinct domains are drawn from related, but different, distributions. A good model should be robust to these kinds of small changes to the input, and produce reliable and stable predictions.

Next we discuss methods for learning models which are robust to variations in the input, before providing details of the neural network model used in our experimental evaluation.

2.1 Conventional Regularization and Dropout

Conventional methods for learning robust models include l_1 and l_2 regularization (Ng, 2004), and dropout (Srivastava et al., 2014). In fact, Wager et al. (2013) showed that the dropout regularizer is first-order equivalent to an l_2 regularizer applied after scaling the features. Dropout is also equivalent to "Follow the Perturbed Leader" (FPL) which perturbs exponential numbers of experts by noise and then predicts with the expert of minimum perturbed loss for online learning robustness (van Erven et al., 2014). Given its popularity in deep learning, we take dropout to be a strong baseline in our evaluation.

The key idea behind dropout is to randomly zero out units, along with their connections, from the network during training, thus limiting the extent of co-adaptation between units. We apply dropout on the representation vector $\mathbf{h}$, denoted $\hat{\mathbf{h}} = \text{dropout}_\beta(\mathbf{h})$, where β is the dropout rate. Similarly to our proposed method, training with dropout requires gradient based search for the minimizer of the loss L.

We also use dropout to generate noise in the test data as part of our experimental simulations, as we will discuss later.

2.2 Robust Regularization

Our method is inspired by the work on adversarial training in computer vision (Goodfellow et al., 2014). In image recognition tasks, small distortions that are indiscernible to humans can significantly distort the predictions of neural networks (Szegedy et al., 2014). An intuitive explanation of our regularization method is, when noise is applied to the data, the variation of the output is kept lower than the noise. We adapt this idea from Rifai et al. (2011) and develop the Jacobian regularization method.

The proposed regularization method works as follows. Conventional training seeks to minimise the difference between $\mathbf{y}_{\text{true}}$ and $\mathbf{y}_{\text{pred}}$. However, in order to make our model robust against noise, we also want to minimize the variation of the output when noise is applied to the input. This is to say, when perturbations are applied to the input, there should be as little perturbation in the output as possible. Formally, the perturbations of output can be written as $\mathbf{p}_{\text{y}} = \text{M}(\mathbf{x}+\mathbf{p}_{\text{x}}) - \text{M}(\mathbf{x})$, where $\mathbf{x}$ is the input, $\mathbf{p}_{\text{x}}$ is the vector of perturbations applied to $\mathbf{x}$, M expresses the trained model, $\mathbf{p}_{\text{y}}$ is the vector of perturbations generated by the model, and the output distribution $\mathbf{y} = \text{M}(\mathbf{x})$. Therefore

$$\lim_{\mathbf{p_x}\to\mathbf{0}} \mathbf{p_y} = \lim_{\mathbf{p_x}\to\mathbf{0}} \big(\text{M}(\mathbf{x}+\mathbf{p}_{\text{x}}) - \text{M}(\mathbf{x})\big) = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \cdot \mathbf{p_x},$$

$$\text{and} \quad \text{distance}\left(\lim_{\mathbf{p_x}\to\mathbf{0}} \mathbf{p}_{\text{y}}/\mathbf{p}_{\text{x}}, \mathbf{0}\right) = \left\| \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right\|_F .$$

In other words, minimising local noise sensitivity is equivalent to minimising the Frobenius norm of the Jacobean matrix of partial derivatives of the model outputs *wrt* its inputs.

To minimize the effect of perturbation noise, our method involves an additional term in the loss function, in the form of the derivative of loss L with respect to hidden layer $\mathbf{h}$. Note that while in principle we could consider robustness to perturbations in the input $\mathbf{x}$, the discrete nature of $\mathbf{x}$ adds additional mathematical complications, and thus we defer this setting for future work. Combining the elements, the new loss function can be expressed as

$$\mathscr{L} = L + \lambda \cdot \left\| \frac{\partial L}{\partial \mathbf{h}} \right\|_2 , \tag{1}$$

where λ is a weight term, and $\mathrm{distance}$ takes the form of the l_2 norm. The training objective in Equation (1) supports gradient optimization, but note that it requires the calculation of second-order derivatives of L during back propagation, arising from the $\partial L/\partial \mathbf{h}$ term. Henceforth we refer to this method as **robust regularization**.

2.3 Convolutional Network

For the purposes of this paper, we focus exclusively on convolutional neural networks (CNNs), but stress that the method is compatible with other neural architectures and other types of parametric models (not just deep neural networks). The CNN used in this research is based on the model proposed by Kim (2014), and is outlined below.

Let S be the sentence, consisting of n words $\{w_1, w_2, \cdots, w_n\}$. A look-up table is applied to S, made up of word vectors $\mathbf{e}_{\mathrm{i}} \in \mathbb{R}^m$ corresponding to each word w_i, where m is the word vector dimensionality. Thus, sentence S can be represented as a matrix $\mathbf{E}_{\mathrm{S}} \in \mathbb{R}^{m \times n}$ by concatenating the word vectors $\mathbf{E}_{\mathrm{S}} = \bigoplus_{i=1}^{n} \mathbf{e}_{\mathrm{w_i}}$.

A convolutional layer combined with a number of wide convolutional filters is applied to $\mathbf{E}_{\mathrm{S}}$. Specifically, the k-th convolutional filter operator filter_k involves a weight vector $\mathbf{w}_{\mathrm{k}} \in \mathbb{R}^{m \times t}$, which works on every t_k-sized window of $\mathbf{E}_{\mathrm{S}}$, and is accompanied by a bias term $b \in \mathbb{R}$. The filter operator is followed by the non-linear function $\mathscr{F}$, a rectified linear unit, ReLU, followed by a max-pooling operation, to generate a hidden activation $h_k = \mathrm{MaxPooling}(\mathscr{F}(\mathrm{filter}_k(\mathbf{E}_{\mathrm{S}}; \mathbf{w}_{\mathrm{k}}, b))$. Multiple filters with different window sizes are used to learn different local properties of the sentence. We concatenate all the hidden activations h_k to form a hidden layer $\mathbf{h}$, with size equal to the number of filters. Details of parameter settings can be found in Section 3.2.

The feature vector $\mathbf{h}$ is fed into a final $\mathrm{softmax}$ layer with a linear transform to generate a probability distribution over labels

$$\mathbf{y}_{\mathrm{pred}} = \mathrm{softmax}(\mathbf{w} \cdot \mathbf{h} + \mathbf{b}),$$

where $\mathbf{w}$ and $\mathbf{b}$ are parameters. Finally, the model minimizes the loss of the cross-entropy between the ground-truth and the model prediction, $L = \mathrm{CrossEntropy}(\mathbf{y}_{\mathrm{true}}, \mathbf{y}_{\mathrm{pred}})$, for which we use stochastic gradient descent.

3 Datasets and Experimental Setups

We experiment on the following datasets,[2] following Kim (2014):

- MR: Sentence polarity dataset (Pang and Lee, 2008)[3]
- Subj: Subjectivity dataset (Pang and Lee, 2005)[3]
- CR: Customer review dataset (Hu and Liu, 2004)[4]
- SST: Stanford Sentiment Treebank, using the 3-class configuration (Socher et al., 2013)[5]

In each case, we evaluate using classification accuracy.

3.1 Noisifying the Data

Different to conventional evaluation, we corrupt the test data with noise in order to evaluate the robustness of our model. We assume that when dealing with short text such as Twitter posts, it is common to see unknown words due to typos, abbreviations and sociolinguistic marking of different types (Han and Baldwin, 2011; Eisenstein, 2013). To simulate this, we apply word-level dropout noise to each document, by randomly replacing words by a unique sentinel symbol.[6] This is applied to each word with probability $\alpha \in \{0, 0.1, 0.2, 0.3\}$.

We also experimented with adding different levels of Gaussian noise to the sentence embeddings $\mathbf{E}_{\mathrm{S}}$, but found the results to be largely consistent with those for word dropout noise, and therefore we have omitted these results from the paper.

To directly test the robustness under a more realistic setting, we additionally perform cross-domain evaluation, where we train a model on one dataset

[2] For datasets where there is no pre-defined training/test split, we evaluate using 10-fold cross validation. Refer to Kim (2014) for more details on the datasets.

[3] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[4] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[5] http://nlp.stanford.edu/sentiment/

[6] This was to avoid creating new n-grams which would occur when symbols are deleted from the input. Masking tokens instead results in partially masked n-grams as input to the convolutional filters.

Dataset		MR				Subj			
Word dropout rate (α)		0	0.1	0.2	0.3	0	0.1	0.2	0.3
Baseline		80.5	79.4	77.9	76.5	**93.1**	92.0	90.9	89.8
Dropout (β)	0.3	80.3	79.5	78.1	76.7	92.7	92.0	90.9	89.5
	0.5	80.3	79.0	78.0	76.5	93.0	92.0	91.1	89.9
	0.7	80.3	79.3	78.3	76.8	92.8	91.9	90.9	89.8
Robust Regularization (λ)	10^{-3}	80.5	79.4	78.3	76.7	93.0	**92.2**	91.1	89.8
	10^{-2}	**80.8**	79.3	78.4	77.0	93.0	92.2	91.0	90.0
	10^{-1}	80.4	78.8	77.8	77.0	92.7	91.9	91.0	89.8
	1	79.3	77.1	76.1	75.5	91.7	91.1	90.1	89.3
Dropout + Robust	$\beta = 0.5, \lambda = 10^{-2}$	80.6	**79.9**	**78.6**	**77.3**	93.0	**92.2**	**91.2**	**90.1**

Dataset		CR				SST			
Word dropout rate (α)		0	0.1	0.2	0.3	0	0.1	0.2	0.3
Baseline		83.2	82.3	80.4	77.9	84.1	82.3	80.3	77.8
Dropout (β)	0.3	83.3	82.1	80.3	78.9	84.2	82.3	80.2	78.0
	0.5	83.2	82.4	81.0	79.3	84.2	82.4	80.5	78.2
	0.7	83.2	82.2	80.7	78.8	83.9	82.5	80.9	78.2
Robust Regularization (λ)	10^{-3}	83.3	82.6	81.4	79.5	**84.5**	**82.8**	**81.4**	78.8
	10^{-2}	**83.4**	82.5	81.6	79.3	84.2	82.4	80.7	78.6
	10^{-1}	83.3	**82.7**	**82.0**	79.6	82.5	81.5	79.7	77.6
	1	82.9	81.4	79.8	79.0	82.2	80.9	79.1	77.3
Dropout + Robust	$\beta = 0.5, \lambda = 10^{-2}$	83.3	82.5	81.5	**79.7**	84.3	82.6	80.8	**79.1**

Table 1: Accuracy (%) with increasing word-level dropout across the four datasets. For each dataset, we apply four levels of noise $\alpha = \{0, 0.1, 0.2, 0.3\}$; the best result for each combination of α and dataset is indicated in **bold**. The Baseline model is a simple CNN model without regularization. The last model combines dropout and our method with fixed parameters β and λ as indicated.

and apply it to another. For this, we use the pairing of MR and CR, where the first dataset is based on movie reviews and the second on product reviews, but both use the same label set. Note that there is a significant domain shift between these corpora, due to the very nature of the items reviewed.

3.2 Word Vectors and Hyper-parameters

To set the hyper-parameters of the CNN, we follow the guidelines of Zhang and Wallace (2015), setting word embeddings to $m = 300$ dimensions and initialising based on word2vec pre-training (Mikolov et al., 2013). Words not in the pre-trained vector table were initialized randomly by the uniform distribution $U([-0.25, 0.25)^m)$. The window sizes of filters (t) are set to $3, 4, 5$, with 128 filters for each size, resulting in a hidden layer dimensionality of $384 = 128 \times 3$. We use the Adam optimizer (Kingma and Ba, 2015) for training.

4 Results and Discussions

The results for word-level dropout noise are presented in Table 1. In general, increasing the word-level dropout noise leads to a drop in accuracy for all four datasets, however the relative dropoff in accuracy for Robust Regularization is less than for Word Dropout, and in 15 out of 16 cases (four noise levels across the four datasets), our method achieves the best result. Note that this includes the case of $\alpha = 0$, where the test data is left in its original form, which shows that Robust Regularization is also an effective means of preventing overfitting in the model.

For each dataset, we also evaluated based on the combination of Word Dropout and Robust Regularization using the fixed parameters $\beta = 0.5$ and $\lambda = 10^{-2}$, which are overall the best individual settings. The combined approach performs better than either individual method for the highest noise levels tested across all datasets. This indicates that Robust

Train/Test		MR/CR	CR/MR
Baseline		67.5	61.0
Dropout (β)	0.3	71.6	62.2
	0.5	71.0	62.1
	0.7	70.9	62.0
Robust Regularization (λ)	10^{-3}	70.8	61.6
	10^{-2}	71.1	**62.5**
	10^{-1}	**72.0**	62.2
	1	71.8	62.3
Dropout + Robust	$\beta = 0.5, \lambda = 10^{-2}$	**72.0**	62.4

Table 2: Accuracy under cross-domain evaluation; the best result for each dataset is indicated in **bold**.

Regularization acts in a complementary way to Word Dropout.

Table 2 presents the results of the cross-domain experiment, whereby we train a model on MR and test on CR, and vice versa, to measure the robustness of the different regularization methods in a more real-world setting. Once again, we see that our regularization method is superior to word-level dropout and the baseline CNN, and the techniques combined do very well, consistent with our findings for synthetic noise.

4.1 Running Time

Our method requires second-order derivatives, and thus is a little slower at training time. Figure 1 is a plot of the training and test accuracy at varying points during training over SST.

We can see that the runtime till convergence is only slightly slower for Robust Regularization than standard training, at roughly 30 minutes on a two-core CPU (one fold) with standard training vs. 35–40 minutes with Robust Regularization. The convergence time for Robust Regularization is comparable to that for Word Dropout.

5 Conclusions

In this paper, we present a robust regularization method which explicitly minimises a neural model's sensitivity to small changes in its hidden representation. Based on evaluation over four sentiment analysis datasets using convolutional neural networks, we found our method to be both superior and complementary to conventional word-level dropout under varying levels of noise, and in a cross-domain evaluation.

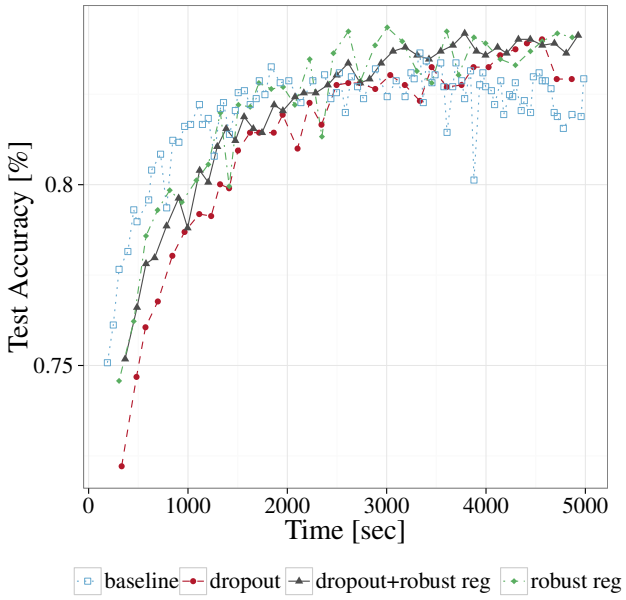


Figure 1: Time–accuracy evaluation over the different combinations of Word Dropout (dropout) and Robust Regularization (robust reg) over SST, without injecting noise.

For future work, we plan to apply our regularization method to other models and tasks to determine how generally applicable our method is. Also, we will explore methods for more realistic linguistic noise, such as lexical, syntactic and semantic noise, to develop models that are robust to the kinds of data often encountered at test time.

Acknowledgments

We are grateful to the anonymous reviewers for their helpful feedback and suggestions. This work was supported by the Australian Research Council (grant numbers FT130101105 and FT120100658). Also, we would like to thank the developers of Tensorflow (Abadi et al., 2015), which was used for the experiments in this paper.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg,

Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Technical report, Google Research.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Zsolt Bitvai and Trevor Cohn. 2015. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 180–185.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369.

Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2015. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. In *Proceedings of the NIPS 2014 Deep Learning and Representation Learning Workshop*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 368–378.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

James Martens. 2010. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, pages 735–742.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, pages 833–840.

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

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob

Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*.

Pedro Tabacof and Eduardo Valle. 2016. Exploring the space of adversarial images. In *Proceedings of the IEEE International Joint Conference on Neural Networks*.

Tim van Erven, Wojciech Kotłowski, and Manfred K. Warmuth. 2014. Follow the leader with dropout perturbations. In *Proceedings of the 27th Conference on Learning Theory*, pages 949–974.

Stefan Wager, Sida Wang, and Percy S. Liang. 2013. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*, pages 351–359.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 643–648.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Modified Dirichlet Distribution: Allowing Negative Parameters to Induce Stronger Sparsity*

Kewei Tu
School of Information Science and Technology
ShanghaiTech University, Shanghai, China
tukw@shanghaitech.edu.cn

Abstract

The Dirichlet distribution (Dir) is one of the most widely used prior distributions in statistical approaches to natural language processing. The parameters of Dir are required to be positive, which significantly limits its strength as a sparsity prior. In this paper, we propose a simple modification to the Dirichlet distribution that allows the parameters to be negative. Our modified Dirichlet distribution (mDir) not only induces much stronger sparsity, but also simultaneously performs smoothing. mDir is still conjugate to the multinomial distribution, which simplifies posterior inference. We introduce two simple and efficient algorithms for finding the mode of mDir. Our experiments on learning Gaussian mixtures and unsupervised dependency parsing demonstrate the advantage of mDir over Dir.

1 Dirichlet Distribution

The Dirichlet distribution (Dir) is defined over probability vectors $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ with positive parameter vector $\boldsymbol{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$:

$$\mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{n} x_i^{\alpha_i - 1}$$

where the normalization factor $\mathrm{B}(\boldsymbol{\alpha})$ is the multivariate beta function. When the elements in $\boldsymbol{\alpha}$ are larger than one, Dir can be used as a smoothness prior that prefers more uniform probability vectors, with larger $\boldsymbol{\alpha}$ values inducing more smoothness. When the elements in $\boldsymbol{\alpha}$ are less than one, Dir can be seen as a sparsity prior that prefers sparse probability vectors, with smaller $\boldsymbol{\alpha}$ values inducing stronger sparsity. To better understand its sparsity preference, we take the logarithm of Dir:

$$\log \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}) = \sum_{i=1}^{n} (\alpha_i - 1) \log x_i + \text{constant}$$

Since $\alpha_i - 1$ is negative, the closer x_i is to zero, the higher the log probability becomes. The coefficient $\alpha_i - 1$ controls the strength of the sparsity preference. However, α_i is required to be positive in Dir because otherwise the normalization factor becomes divergent. Consequently, the strength of the sparsity preference is upper bounded. This becomes problematic when a strong prior is needed, for instance, when the training dataset is large relative to the model size (e.g., in unsupervised learning of an unlexicalized probabilistic grammar) and thus the likelihood may dominate the posterior without a strong prior.

2 Modified Dirichlet Distribution

We make a simple modification to the Dirichlet distribution that allows the parameters in $\boldsymbol{\alpha}$ to become negative. To handle the divergent normalization factor, we require that each x_i must be lower bounded by a small positive constant ϵ. Our modified Dirichlet distribution (mDir) is defined as follows.

$$\mathrm{mDir}(\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon) = \begin{cases} 0 & \text{if } \exists i, x_i < \epsilon \\ \frac{1}{Z(\boldsymbol{\alpha}, \epsilon)} \prod_{i=1}^{n} x_i^{\alpha_i - 1} & \text{otherwise} \end{cases}$$

where we require $0 < \epsilon \leq \frac{1}{n}$ and do *not* require α_i to be positive. With fixed values of $\boldsymbol{\alpha}$ and ϵ, the

*This work was supported by the National Natural Science Foundation of China (61503248).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1986–1991,
Austin, Texas, November 1-5, 2016.

unnormalized probability density is always bounded and hence the normalization factor $Z(\boldsymbol{\alpha}, \epsilon)$ is finite.

It is easy to show that mDir is still conjugate to the multinomial distribution. Similar to Dir, mDir can be used as a smoothness/sparsity prior depending on the values of $\boldsymbol{\alpha}$. Because $\boldsymbol{\alpha}$ is no longer required to be positive, we can achieve very strong sparsity preference by using a highly negative vector of $\boldsymbol{\alpha}$. Note that here we no longer achieve sparsity in its strict sense; instead, by sparsity we mean most elements in $\boldsymbol{x}$ reach their lower bound ϵ. Parameter ϵ can thus be seen as a smoothing factor that prevents any element in $\boldsymbol{x}$ to become too small. Therefore, with proper parameters, mDir is able to simultaneously achieve sparsity and smoothness. This can be useful in many applications where one wants to learn a sparse multinomial distribution in an iterative way without premature pruning of components.

2.1 Finding the Mode

If $\forall i, \alpha_i - 1 \leq 0$, then the mode of mDir can be shown to be:

$$x_i = \begin{cases} 1 - (n-1)\epsilon & \text{if } i = \arg\max_i \alpha_i \\ \epsilon & \text{otherwise} \end{cases}$$

Otherwise, we can find the mode with Algorithm 1. The algorithm first lets $x_i = \epsilon$ if $\alpha_i \leq 1$ and otherwise lets x_i be proportional to $\alpha_i - 1$. It then looks for variables in $\boldsymbol{x}$ that are less than ϵ, increases them to ϵ, and renormalizes the rest of the variables. The renormalization may decrease some additional variables below ϵ, so the procedure is repeated until all the variables are larger than or equal to ϵ.

Theorem 1. *If $\exists i, \alpha_i > 1$, then Algorithm 1 correctly finds a mode of* $\mathrm{mDir}(\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon)$.

Proof. First, we can show that for any i such that $\alpha_i \leq 1$, we must have $x_i = \epsilon$ at the mode. This is because if $x_i > \epsilon$, then we can increase the probability density by first decreasing x_i to ϵ (hence increasing $x_i^{\alpha_i - 1}$), and then increasing some other variable x_j with $\alpha_j > 1$ to satisfy the normalization condition (hence also increasing $x_j^{\alpha_j - 1}$). This is consistent with the output of the algorithm.

Once we fix the value to ϵ for any variable x_i s.t. $\alpha_i \leq 1$, the log probability density function becomes strictly concave on the simplex specified by the linear constraints $\sum_i x_i = 1$ and $x_i \geq \epsilon$.

Algorithm 1 Mode-finding of $\mathrm{mDir}(\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon)$

```
S ← {i|α_i ≤ 1}
T ← ∅
repeat
    T ← T ⋃ S
    for i ∈ T do
        x_i ← ε
    end for
    z ← Σ_{i∉T} (α_i − 1)
    for i ∉ T do
        x_i ← ((α_i − 1)/z) × (1 − ε|T|)
    end for
    S ← {i|x_i < ε}
until S = ∅
return ⟨x_1, ..., x_n⟩
```

The strict concavity can be proved by showing that the log probability density function is twice differentiable and the Hessian is negative definite at any point of the simplex.

With a concave function and linear constraints, the KKT conditions are sufficient for optimality. We need to show that the output of the algorithm satisfies the following KKT conditions:

- Stationarity: $\forall i, \frac{\alpha_i - 1}{x_i} = -\mu_i + \lambda$
- Primal feasibility: $\forall i, x_i \geq \epsilon$ and $\sum_i x_i = 1$
- Dual feasibility: $\forall i, \mu_i \geq 0$
- Complementary slackness: $\forall i, \mu_i(x_i - \epsilon) = 0$

Let $x_i^{(k)}$ and $T^{(k)}$ be the values of x_i and T after k iterations of the algorithm. Suppose the algorithm terminates after K iterations. So the output of the algorithm is $\langle x_1^{(K)}, \ldots, x_n^{(K)} \rangle$, which we will prove satisfies the KKT conditions.

For any i s.t. $x_i^{(K)} > \epsilon$, we set $\mu_i = 0$ and $\lambda = \frac{\alpha_i - 1}{x_i^{(K)}}$ to satisfy all the conditions involving $x_i^{(K)}$.

For any j s.t. $x_j^{(K)} = \epsilon$, suppose $x_j^{(k)} < \epsilon$, i.e., x_j falls below ϵ in iteration k and is set to ϵ afterwards. Pick some i s.t. $i \notin T^{(K)}$. After iteration k and $k+1$ respectively, we have:

$$\frac{x_i^{(k)}}{\alpha_i - 1} = \frac{1 - \epsilon \|T^{(k)}\| - \sum_{j' \in T^{(k+1)} \setminus T^{(k)}} x_{j'}^{(k)}}{\sum_{j' \notin T^{(k+1)}} \alpha_{j'} - 1}$$

$$\frac{x_i^{(k+1)}}{\alpha_i - 1} = \frac{1 - \epsilon \|T^{(k+1)}\|}{\sum_{j' \notin T^{(k+1)}} \alpha_{j'} - 1}$$

Algorithm 2 Fast mode-finding of mDir($\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon$)

$\langle \alpha_{k_1}, \ldots, \alpha_{k_n} \rangle \leftarrow \langle \alpha_1, \ldots, \alpha_n \rangle$ in ascending order
$s_n \leftarrow \alpha_{k_n} - 1$
for $i = n-1, \ldots, 1$ **do**
    $s_i = s_{i+1} + \alpha_{k_i} - 1$ ▷ So $s_i = \sum_{j \geq i} (\alpha_{k_j} - 1)$
end for
$t \leftarrow 0$
for $i = 1, \ldots, n$ **do**
    $x_{k_i} \leftarrow \frac{\alpha_{k_i} - 1}{s_i} \times (1 - \epsilon\, t)$
    if $x_{k_i} < \epsilon$ **then**
        $x_{k_i} \leftarrow \epsilon\,, \quad t \leftarrow t + 1$
    end if
end for
return $\langle x_1, \ldots, x_n \rangle$

Because for any $j' \in T^{(k+1)} \backslash T^{(k)}$ we have $x_{j'}^{(k)} < \epsilon$, from the two equations above we can deduce that $x_i^{(k)} > x_i^{(k+1)}$, i.e., x_i monotonically decreases over iterations. Therefore,

$$(\alpha_j - 1) \times \frac{x_i^{(K)}}{\alpha_i - 1} < (\alpha_j - 1) \times \frac{x_i^{(k)}}{\alpha_i - 1} = x_j^{(k)} < \epsilon$$

So we get

$$\frac{\alpha_j - 1}{\epsilon} < \frac{\alpha_i - 1}{x_i^{(K)}} = \lambda$$

So we set $\mu_j = \lambda - \frac{\alpha_j - 1}{\epsilon}$ and all the conditions involving $x_j^{(K)}$ are also satisfied. The proof is now complete. □

The worst-case time complexity of Algorithm 1 is $O(n^2)$, but in practice when ϵ is small, the algorithm almost always terminates after only one iteration, leading to linear running time. We also provide a different mode-finding algorithm with better worst-case time complexity $\Theta(n \log n)$ (Algorithm 2). It differs from Algorithm 1 in that the elements of $\boldsymbol{\alpha}$ are first sorted, so we can finish computing $\boldsymbol{x}$ in one pass. It can be more efficient than Algorithm 1 when both ϵ and n are larger. Its correctness can be proved in a similar way to that of Algorithm 1.

2.2 Related Distribution

The closest previous work to mDir is the pseudo-Dirichlet distribution (Larsson and Ugander, 2011). It also allows negative parameters to achieve stronger sparsity. However, the pseudo-Dirichlet distribution is no longer conjugate to the multinomial distribution. Consequently, its maximum a posteriori inference becomes complicated and has no time-complexity guarantee.

3 Learning Mixtures of Gaussians

We first evaluate mDir in learning mixtures of Gaussians from synthetic data. The ground-truth model contains two bivariate Gaussian components with equal mixing probabilities (Figure 5(a)). From the ground-truth we sampled two training datasets of 20 and 200 data points. We then tried to fit a Gaussian mixture model with five components.

Three approaches were tested: maximum likelihood estimation using expectation-maximization (denoted by EM), which has no sparsity preference; mean-field variational Bayesian inference with a Dir prior over the mixing probabilities (denoted by VB-Dir), which is the most frequently used inference approach for Dir with $\alpha < 1$; maximum a posteriori estimation using expectation-maximization with a mDir prior over the mixing probabilities (denoted by EM-mDir). The Dir and mDir priors that we used are both symmetric, i.e., all the elements in vector $\boldsymbol{\alpha}$ have the same value, denoted by α. For mDir, we set $\epsilon = 10^{-5}$. We ran each approach under each parameter setting for 300 times with different random initialization and then reported the average results. During learning, we pruned a Gaussian component whenever its covariance matrix becomes numerically singular (which means the component is estimated from only one or two data samples).

Figure 1–4 show the average test set log likelihood and the effective numbers of mixture components of the models learned with different values of parameter α from 20 and 200 samples respectively. For VB-Dir, we show the results with the α value as low as 10^{-5}. Further decreasing α did not improve the results. It can be seen that both VB-Dir and EM-mDir can achieve better test set likelihood and lower effective numbers of components than EM with proper α values. EM-mDir outperforms VB-Dir even with positive α values, and its performance is further boosted when α becomes negative. The improvement of EM-mDir when α becomes negative is smaller in the 20-sample case than in the 200-sample case. This is because when the train-

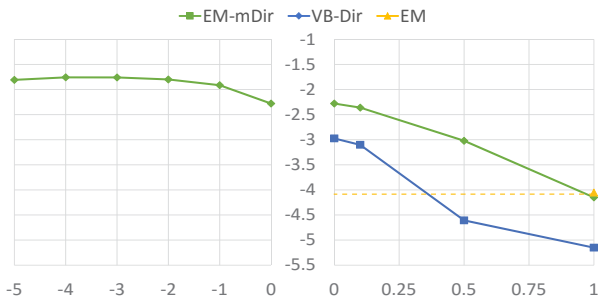


Figure 1: Test set log likelihood vs. the value of α (20 training samples)

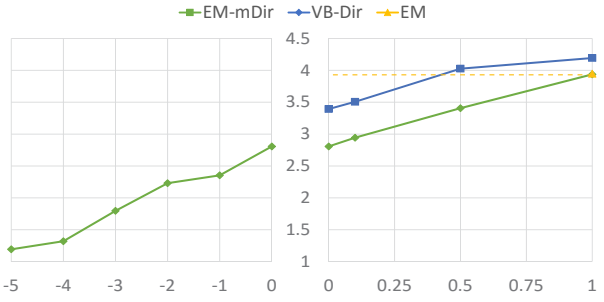


Figure 2: Effective number of components vs. the value of α (20 training samples)

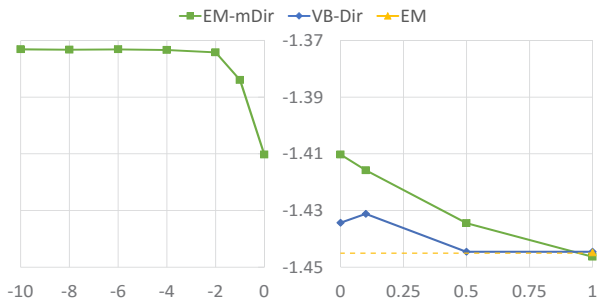


Figure 3: Test set log likelihood vs. the value of α (200 training samples)

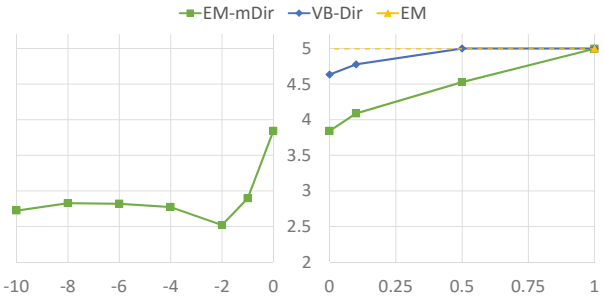


Figure 4: Effective number of components vs. the value of α (200 training samples)

ing dataset is small, a small positive α value may already be sufficient in inducing enough sparsity.

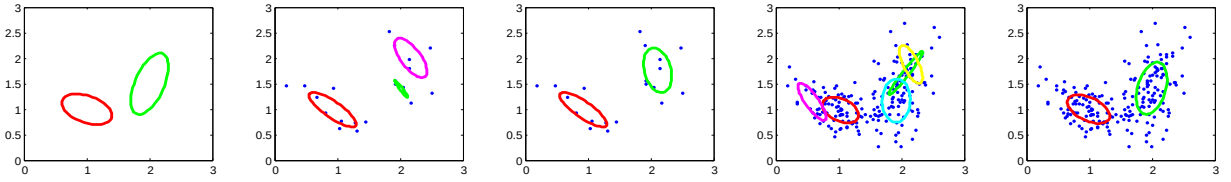
Figure 5(b)–(e) show the typical models learned by VB-Dir and EM-mDir. When the training dataset is small, both Dir and mDir are effective sparsity priors that help prune unnecessary mixture components, though mDir can be more effective with a negative α value. When the training dataset is large, however, the Dir prior is overwhelmed by the likelihood in posterior inference and cannot effectively prune mixture components. On the other hand, with a highly negative α value, mDir is still effective as a sparsity prior.

4 Unsupervised Dependency Parsing

Unsupervised dependency parsing aims to learn a dependency grammar from unannotated text. Previous work has shown that sparsity regularization improves the performance of unsupervised dependency parsing (Johnson et al., 2007; Gillenwater et al., 2010). In our experiments, we tried to learn a dependency model with valence (DMV) (Klein and Manning, 2004) from the Wall Street Journal corpus, with section 2-21 for training and section 23 for testing. Following previous work, we used sentences of length ≤ 10 with punctuation stripped off. Since DMV is an unlexicalized model, the number of dependency rules is small relative to the training corpus size. This suggests that a strong prior can be helpful in counterbalancing the influence of the training data.

We tested six approaches. With a mDir prior, we tried EM, hard EM, and softmax-EM with $\sigma = 0.5$ (Tu and Honavar, 2012) (denoted by EM-mDir, HEM-mDir, SEM-mDir). With a Dir prior, we tried variational inference, hard variational inference, and softmax variational inference with $\sigma = 0.5$ (Tu and Honavar, 2012) (denoted by VB-Dir, HVB-Dir, SVB-Dir). Again, we used symmetric Dir and mDir priors. For mDir, we set $\epsilon = 10^{-4}$ by default.

Figure 6 shows the directed accuracy of parsing the test corpus using the learned dependency models. It can be seen that with positive α values, Dir and mDir have very similar accuracy under the standard, hard and softmax versions of inference respectively. With negative α values, the accuracy of EM-mDir decreases; but for HEM-mDir and SEM-mDir, the accuracy is significantly improved with moder-



(a) Ground-truth (b) VB-Dir, $\alpha = 10^{-5}$ (c) EM-mDir, $\alpha = -2$ (d) VB-Dir, $\alpha = 10^{-5}$ (e) EM-mDir, $\alpha = -30$

Figure 5: The ground-truth model and four typical models learned by VB-Dir and EM-mDir. (b),(c): 20 training samples. (d),(e): 200 training samples.

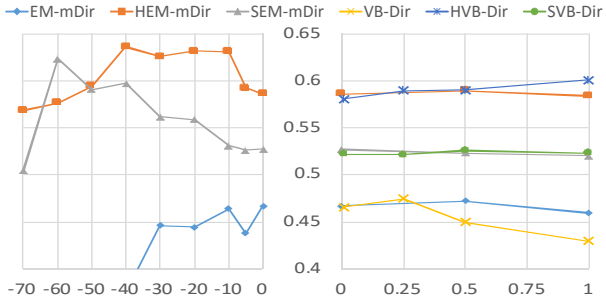


Figure 6: Parsing accuracy vs. the value of α

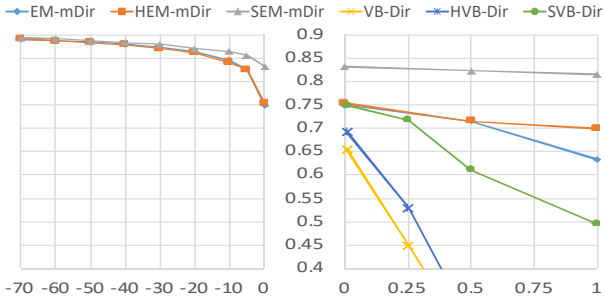


Figure 7: Sparsity of the learned grammars vs. the value of α

ately negative α values. HEM-mDir consistently produces accuracy around 0.63 with a large range of α values (from -10 to -40), which is on a par with the best published results in learning the original DMV model (Cohen and Smith, 2009; Gillenwater et al., 2010; Berg-Kirkpatrick et al., 2010), even though these previous approaches employed more sophisticated features and advanced regularization techniques than ours.

Figure 7 shows the degree of sparsity of the learned dependency grammars. We computed the percentage of dependency rules with probabilities below 10^{-3} to measure the degree of sparsity. It can be seen that even with positive α values, mDir leads to significantly more sparse grammars than Dir does. With negative values of α, mDir can induce even more sparsity.

Figure 8 plots the parsing accuracy with different values of parameter ϵ in mDir (α is set to -20). The best accuracy is achieved when ϵ is neither too large nor too small. This is because if ϵ is too large, the probabilities of dependency rules become too uniform to be discriminative. On the other hand, if ϵ is too small, then the probabilities of many dependency rules may become too small in the early stages of learning and never be able to recover. Similar observation was made by Johnson et al. (2007) when doing maximum a posteriori estimation with a Dir prior (hence with $\epsilon = 0$).

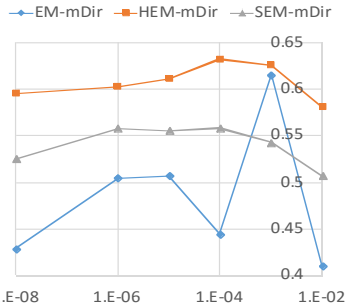


Figure 8: Parsing accuracy vs. the value of ϵ

5 Conclusion

We modify the Dirichlet distribution to allow negative values of parameter $\boldsymbol{\alpha}$ so that it induces stronger sparsity when used as a prior of a multinomial distribution. A second parameter ϵ is introduced which prevents divergence of the normalization factor and also acts as a smoothing factor. Our modified Dirichlet distribution (mDir) is still conjugate to the multinomial distribution. We propose two efficient algorithms for finding the mode of mDir. Our experiments on learning Gaussian mixtures and unsupervised dependency parsing show the advantage of mDir over the Dirichlet distribution.

References

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

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *HLT-NAACL*, pages 74–82.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *ACL '10: Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199, Morristown, NJ, USA. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for pcfgs via markov chain monte carlo. In *HLT-NAACL*, pages 139–146.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Martin O Larsson and Johan Ugander. 2011. A concave regularization technique for sparse mixture models. In *Advances in Neural Information Processing Systems*, pages 1890–1898.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334. Association for Computational Linguistics.

Gated Word-Character Recurrent Language Model

Yasumasa Miyamoto
Center for Data Science
New York University
yasumasa.miyamoto@nyu.edu

Kyunghyun Cho
Courant Institute of
Mathematical Sciences
& Centre for Data Science
New York University
kyunghyun.cho@nyu.edu

Abstract

We introduce a recurrent neural network language model (RNN-LM) with long short-term memory (LSTM) units that utilizes both character-level and word-level inputs. Our model has a gate that adaptively finds the optimal mixture of the character-level and word-level inputs. The gate creates the final vector representation of a word by combining two distinct representations of the word. The character-level inputs are converted into vector representations of words using a bidirectional LSTM. The word-level inputs are projected into another high-dimensional space by a word lookup table. The final vector representations of words are used in the LSTM language model which predicts the next word given all the preceding words. Our model with the gating mechanism effectively utilizes the character-level inputs for rare and out-of-vocabulary words and outperforms word-level language models on several English corpora.

1 Introduction

Recurrent neural networks (RNNs) achieve state-of-the-art performance on fundamental tasks of natural language processing (NLP) such as language modeling (RNN-LM) (Józefowicz et al., 2016; Zoph et al., 2016). RNN-LMs are usually based on the word-level information or subword-level information such as characters (Mikolov et al., 2012), and predictions are made at either word level or subword level respectively.

In word-level LMs, the probability distribution over the vocabulary conditioned on preceding words is computed at the output layer using a softmax function. [1] Word-level LMs require a predefined vocabulary size since the computational complexity of a softmax function grows with respect to the vocabulary size. This closed vocabulary approach tends to ignore rare words and typos, as the words do not appear in the vocabulary are replaced with an out-of-vocabulary (OOV) token. The words appearing in vocabulary are indexed and associated with high-dimensional vectors. This process is done through a word lookup table.

Although this approach brings a high degree of freedom in learning expressions of words, information about morphemes such as prefix, root, and suffix is lost when the word is converted into an index. Also, word-level language models require some heuristics to differentiate between the OOV words, otherwise it assigns the exactly same vector to all the OOV words. These are the major limitations of word-level LMs.

In order to alleviate these issues, we introduce an RNN-LM that utilizes both character-level and word-level inputs. In particular, our model has a gate that adaptively choose between two distinct ways to represent each word: a word vector derived from the character-level information and a word vector stored in the word lookup table. This gate is trained to make this decision based on the input word.

According to the experiments, our model with the gate outperforms other models on the Penn Treebank (PTB), BBC, and IMDB Movie Review datasets. Also, the trained gating values show that the gating mechanism effectively utilizes the character-level

[1] softmax function is defined as $f(x_i) = \frac{\exp x_i}{\sum_k \exp x_k}$.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1992–1997,
Austin, Texas, November 1-5, 2016.

information when it encounters rare words.

Related Work Character-level language models that make word-level prediction have recently been proposed. Ling et al. (2015a) introduce the compositional character-to-word (C2W) model that takes as input character-level representation of a word and generates vector representation of the word using a bidirectional LSTM (Graves and Schmidhuber, 2005). Kim et al. (2015) propose a convolutional neural network (CNN) based character-level language model and achieve the state-of-the-art perplexity on the PTB dataset with a significantly fewer parameters.

Moreover, word–character hybrid models have been studied on different NLP tasks. Kang et al. (2011) apply a word–character hybrid language model on Chinese using a neural network language model (Bengio et al., 2003). Santos and Zadrozny (2014) produce high performance part-of-speech taggers using a deep neural network that learns character-level representation of words and associates them with usual word representations. Bojanowski et al. (2015) investigate RNN models that predict characters based on the character and word level inputs. Luong and Manning (2016) present word–character hybrid neural machine translation systems that consult the character-level information for rare words.

2 Model Description

The model architecture of the proposed word–character hybrid language model is shown in Fig. 1.

Word Embedding At each time step t, both the word lookup table and a bidirectional LSTM take the same word w_t as an input. The word-level input is projected into a high-dimensional space by a word lookup table $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, where $|V|$ is the vocabulary size and d is the dimension of a word vector:

$$\mathbf{x}^{\text{word}}_{w_t} = \mathbf{E}^\top \mathbf{w}_{w_t}, \tag{1}$$

where $\mathbf{w}_{w_t} \in \mathbb{R}^{|V|}$ is a one-hot vector whose i-th element is 1, and other elements are 0. The character–level input is converted into a word vector by using a bidirectional LSTM. The last hidden states of forward and reverse recurrent networks are linearly

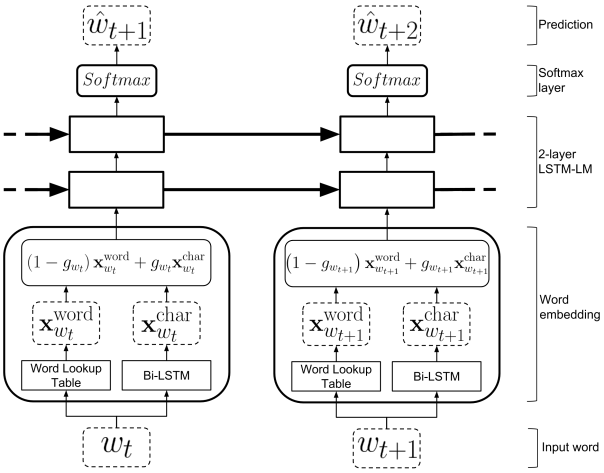


Figure 1: The model architecture of the gated word-character recurrent language model. w_t is an input word at t. $\mathbf{x}^{\text{word}}_{w_t}$ is a word vector stored in the word lookup table. $\mathbf{x}^{\text{char}}_{w_t}$ is a word vector derived from the character-level input. g_{w_t} is a gating value of a word w_t. $\hat{w}_{t+1}$ is a prediction made at t.

combined:

$$\mathbf{x}^{\text{char}}_{w_t} = \mathbf{W}^f \mathbf{h}^f_{w_t} + \mathbf{W}^r \mathbf{h}^r_{w_t} + \mathbf{b}, \tag{2}$$

where $\mathbf{h}^f_{w_t}, \mathbf{h}^r_{w_t} \in \mathbb{R}^d$ are the last states of the forward and the reverse LSTM respectively. $\mathbf{W}^f, \mathbf{W}^r \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are trainable parameters, and $\mathbf{x}^{\text{char}}_{w_t} \in \mathbb{R}^d$ is the vector representation of the word w_t using a character input. The generated vectors $\mathbf{x}^{\text{word}}_{w_t}$ and $\mathbf{x}^{\text{char}}_{w_t}$ are mixed by a gate g_{w_t} as

$$\begin{aligned} g_{w_t} &= \sigma\left(\mathbf{v}_g^\top \mathbf{x}^{\text{word}}_{w_t} + b_g\right) \\ \mathbf{x}_{w_t} &= (1 - g_{w_t})\,\mathbf{x}^{\text{word}}_{w_t} + g_{w_t}\mathbf{x}^{\text{char}}_{w_t}, \end{aligned} \tag{3}$$

where $\mathbf{v}_g \in \mathbb{R}^d$ is a weight vector, $b_g \in \mathbb{R}$ is a bias scalar, $\sigma(\cdot)$ is a sigmoid function. This gate value is independent of a time step. Even if a word appears in different contexts, the same gate value is applied. Hashimoto and Tsuruoka (2016) apply a very similar approach to compositional and non-compositional phrase embeddings and achieve state-of-the-art results on compositionality detection and verb disambiguation tasks.

Language Modeling The output vector $\mathbf{x}_{w_t}$ is used as an input to a LSTM language model. Since the word embedding part is independent from the language modeling part, our model retains the flexibility to change the architecture of the language modeling part. We use the architecture similar to the non-regularized LSTM model by Zaremba et al. (2014).

Model	PTB		BBC		IMDB	
	Validation	Test	Validation	Test	Validation	Test
Gated Word & Char, adaptive	117.49	113.87	**78.56**	**87.16**	71.99	72.29
Gated Word & Char, adaptive (Pre-train)	117.03	112.90	80.37	87.51	71.16	71.49
Gated Word & Char, $g = 0.25$	119.45	115.55	79.67	88.04	71.81	72.14
Gated Word & Char, $g = 0.25$ (Pre-train)	**117.01**	**113.52**	80.07	87.99	**70.60**	**70.87**
Gated Word & Char, $g = 0.5$	126.01	121.99	89.27	94.91	106.78	107.33
Gated Word & Char, $g = 0.5$ (Pre-train)	117.54	113.03	82.09	88.61	109.69	110.28
Gated Word & Char, $g = 0.75$	135.58	135.00	105.54	111.47	115.58	116.02
Gated Word & Char, $g = 0.75$ (Pre-train)	179.69	172.85	132.96	136.01	106.31	106.86
Word Only	118.03	115.65	84.47	90.90	72.42	72.75
Character Only	132.45	126.80	88.03	97.71	98.10	98.59
Word & Character	125.05	121.09	88.77	95.44	77.94	78.29
Word & Character (Pre-train)	122.31	118.85	84.27	91.24	80.60	81.01
Non-regularized LSTM (Zaremba, 2014)	120.7	114.5	-	-	-	-

Table 1: Validation and test perplexities on Penn Treebank (PTB), BBC, IMDB Movie Reviews datasets.

One step of LSTM computation corresponds to

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f \mathbf{x}_{w_t} + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i \mathbf{x}_{w_t} + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\tilde{\mathbf{c}}_t &= \tanh\left(\mathbf{W}_{\tilde{c}} \mathbf{x}_{w_t} + \mathbf{U}_{\tilde{c}} \mathbf{h}_{t-1} + \mathbf{b}_{\tilde{c}}\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o \mathbf{x}_{w_t} + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right),
\end{aligned} \tag{4}$$

where $\mathbf{W}_s, \mathbf{U}_s \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_s \in \mathbb{R}^d$ for $s \in \{f, i, \tilde{c}, o\}$ are parameters of LSTM cells. $\sigma(\cdot)$ is an element-wise sigmoid function, $\tanh(\cdot)$ is an element-wise hyperbolic tangent function, and $\odot$ is an element-wise multiplication.

The hidden state $\mathbf{h}_t$ is affine-transformed followed by a softmax function:

$$\Pr\left(w_{t+1} = k | w_{<t+1}\right) = \frac{\exp\left(\mathbf{v}_k^\top \mathbf{h}_t + b_k\right)}{\sum_{k'} \exp\left(\mathbf{v}_{k'}^\top \mathbf{h}_t + b_{k'}\right)}, \tag{5}$$

where $\mathbf{v}_k$ is the k-th column of a parameter matrix $\mathbf{V} \in \mathbb{R}^{d \times |V|}$ and b_k is the k-th element of a bias vector $\mathbf{b} \in \mathbb{R}^d$. In the training phase, we minimizes the negative log-likelihood with stochastic gradient descent.

3 Experimental Settings

We test five different model architectures on the three English corpora. Each model has a unique word embedding method, but all models share the same LSTM language modeling architecture, that has 2 LSTM layers with 200 hidden units, $d = 200$. Except for the character only model, weights in the language modeling part are initialized with uniform random variables between -0.1 and 0.1. Weights of a bidirectional LSTM in the word embedding part are initialized with Xavier initialization (Glorot and Bengio, 2010). All biases are initialized to zero.

Stochastic gradient decent (SGD) with mini-batch size of 32 is used to train the models. In the first k epochs, the learning rate is 1. After the k-th epoch, the learning rate is divided by l each epoch. k manages learning rate decay schedule, and l controls speed of decay. k and l are tuned for each model based on the validation dataset.

As the standard metric for language modeling, perplexity (PPL) is used to evaluate the model performance. Perplexity over the test set is computed as $\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log p_{(w_i|w_{<i})}\right)$, where N is the number of words in the test set, and $p_{(w_i|w_{<i})}$ is the conditional probability of a word w_i given all the preceding words in a sentence. We use Theano (2016) to implement all the models. The code for the models is available from `https://github.com/nyu-dl/gated_word_char_rlm`.

3.1 Model Variations

Word Only (baseline) This is a traditional word-level language model and is a baseline model for our experiments.

Character Only This is a language model where each input word is represented as a character se-

		Train	Validation	test
PTB	# Sentences	42k	3k	4k
	# Word	888k	70k	79k
BBC	# Sentences	37k	2k	2k
	# Word	890k	49k	53k
IMDB	# Sentences	930k	153k	152k
	# Word	21M	3M	3M

Table 2: The size of each dataset.

quence similar to the C2W model in (Ling et al., 2015a). The bidirectional LSTMs have 200 hidden units, and their weights are initialized with Xavier initialization. In addition, the weights of the forget, input, and output gates are scaled by a factor of 4. The weights in the LSTM language model are also initialized with Xavier initialization. All biases are initialized to zero. A learning rate is fixed at 0.2.

Word & Character This model simply concatenates the vector representations of a word constructed from the character input $\mathbf{x}_{w_t}^{\text{char}}$ and the word input $\mathbf{x}_{w_t}^{\text{word}}$ to get the final representation of a word $\mathbf{x}_{w_t}$, i.e.,

$$\mathbf{x}_{w_t} = \left[\begin{array}{c} \mathbf{x}_{w_t}^{\text{char}}; \mathbf{x}_{w_t}^{\text{word}} \end{array} \right]. \quad (6)$$

Before being concatenated, the dimensions of $\mathbf{x}_{w_t}^{\text{char}}$ and $\mathbf{x}_{w_t}^{\text{word}}$ are reduced by half to keep the size of $\mathbf{x}_{w_t}$ comparably to other models.

Gated Word & Character, Fixed Value This model uses a globally constant gating value to combine vector representations of a word constructed from the character input $\mathbf{x}_{w_t}^{\text{char}}$ and the word input $\mathbf{x}_{w_t}^{\text{word}}$ as

$$\mathbf{x}_{w_t} = (1-g)\,\mathbf{x}_{w_t}^{\text{word}} + g\mathbf{x}_{w_t}^{\text{char}}, \quad (7)$$

where g is some number between 0 and 1. We choose $g = \{0.25, 0.5, 0.75\}$.

Gated Word & Character, Adaptive This model uses adaptive gating values to combine vector representations of a word constructed from the character input $\mathbf{x}_{w_t}^{\text{char}}$ and the word input $\mathbf{x}_{w_t}^{\text{word}}$ as the Eq (3).

3.2 Datasets

Penn Treebank We use the Penn Treebank Corpus (Marcus et al., 1993) preprocessed by Mikolov et al. (2010). We use 10k most frequent words and 51 characters. In the training phase, we use only sentences with less than 50 words.

BBC We use the BBC corpus prepared by Greene & Cunningham (2006). We use 10k most frequent words and 62 characters. In the training phase, we use sentences with less than 50 words.

IMDB Movie Reviews We use the IMDB Move Review Corpus prepared by Maas et al. (2011). We use 30k most frequent words and 74 characters. In the training phase, we use sentences with less than 50 words. In the validation and test phases, we use sentences with less than 500 characters.

3.3 Pre-training

For the word–character hybrid models, we applied a pre-training procedure to encourage the model to use both representations. The entire model is trained only using the word-level input for the first m epochs and only using the character-level input in the next m epochs. In the first m epochs, a learning rate is fixed at 1, and a smaller learning rate 0.1 is used in the next m epochs. After the $2m$-th epoch, both the character-level and the word-level inputs are used. We use $m = 2$ for PTB and BBC, $m = 1$ for IMDB.

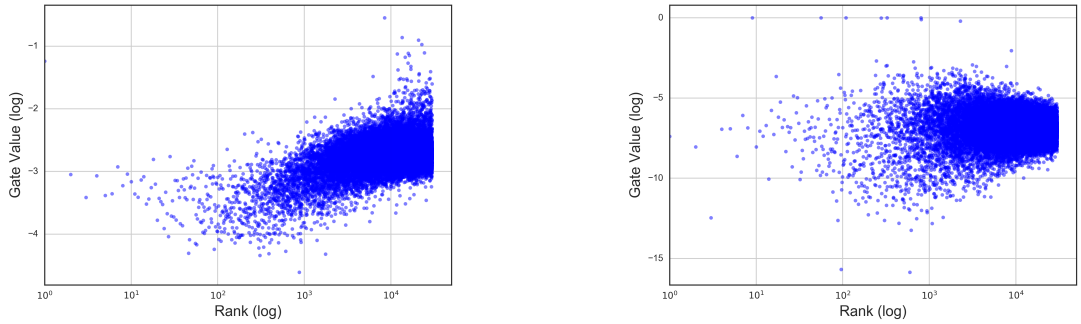
Lample et al. (2016) report that a pre-trained word lookup table improves performance of their word & character hybrid model on named entity recognition (NER). In their method, word embeddings are first trained using skip-n-gram (Ling et al., 2015b), and then the word embeddings are fine-tuned in the main training phase.

4 Results and Discussion

4.1 Perplexity

Table 1 compares the models on each dataset. On the PTB and IMDB Movie Review dataset, the gated word & character model with a fixed gating value, $g_{\text{const}} = 0.25$, and pre-training achieves the lowest perplexity . On the BBC datasets, the gated word & character model without pre-training achieves the lowest perplexity.

Even though the model with fixed gating value performs well, choosing the gating value is not clear and might depend on characteristics of datasets such as size. The model with adaptive gating values does not require tuning it and achieves similar perplexity.



(a) Gated word & character.

(b) Gated word & character with pre-taining.

Figure 2: A log-log plot of frequency ranks and gating values trained in the gated word & character models with/without pre-training.

4.2 Values of Word–Character Gate

The BBC and IMDB datasets retain out-of-vocabulary (OOV) words while the OOV words have been replaced by <unk> in the Penn Treebank dataset. On the BBC and IMDB datasets, our model assigns a significantly high gating value on the unknown word token UNK compared to the other words.

We observe that pre-training results the different distributions of gating values. As can be seen in Fig. 2 (a), the gating value trained in the gated word & character model without pre-training is in general higher for less frequent words, implying that the recurrent language model has learned to exploit the spelling of a word when its word vector could not have been estimated properly. Fig. 2 (b) shows that the gating value trained in the gated word & character model with pre-training is less correlated with the frequency ranks than the one without pre-training. The pre-training step initializes a word lookup table using the training corpus and includes its information into the initial values. We hypothesize that the recurrent language model tends to be word–input–oriented if the informativeness of word inputs and character inputs are not balanced especially in the early stage of training.

Although the recurrent language model with or without pre-training derives different gating values, the results are still similar. We conjecture that the flexibility of modulating between word-level and character-level representations resulted in a better language model in multiple ways.

Overall, the gating values are small. However, this does not mean the model does not utilize the character-level inputs. We observed that the word vectors constructed from the character-level inputs usually have a larger L2 norm than the word vectors constructed from the word-level inputs do. For instance, the mean values of L2 norm of the 1000 most frequent words in the IMDB training set are 52.77 and 6.27 respectively. The small gate values compensate for this difference.

5 Conclusion

We introduced a recurrent neural network language model with LSTM units and a word–character gate. Our model was empirically found to utilize the character-level input especially when the model encounters rare words. The experimental results suggest the gate can be efficiently trained so that the model can find a good balance between the word-level and character-level inputs.

Acknowledgments

This work is done as a part of the course DS-GA 1010-001 Independent Study in Data Science at the Center for Data Science, New York University. KC thanks the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016). YM thanks Kentaro Hanaki, Israel Malkin, and Tian Wang for their helpful feedback. KC and YM thanks the anonymous reviewers for their insightful comments and suggestions.

References

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

[Bojanowski et al.2015] Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. 2015. Alternative structures for character-level rnns. *CoRR*, abs/1511.06303.

[dos Santos and Zadrozny2014] Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1818–1826.

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

[Graves and Schmidhuber2005] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

[Greene and Cunningham2006] Derek Greene and Padraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 377–384.

[Hashimoto and Tsuruoka2016] Kazuma Hashimoto and Yoshimasa Tsuruoka. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. *CoRR*, abs/1603.06067.

[Józefowicz et al.2016] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

[Kang et al.2011] Moonyoung Kang, Tim Ng, and Long Nguyen. 2011. Mandarin word-character hybrid-input neural network language model. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 625–628.

[Kim et al.2015] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

[Lample et al.2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

[Ling et al.2015a] Wang Ling, Tiago Luís, Luís Marujo, Rámon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. EMNLP.

[Ling et al.2015b] Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W. Black, Isabel Trancoso, and Chu-Cheng Lin. 2015b. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1367–1372.

[Luong and Manning2016] Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788.

[Maas et al.2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150.

[Marcus et al.1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

[Mikolov et al.2012] Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. 2012. Subword language modeling with neural networks.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

[Zaremba et al.2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

[Zoph et al.2016] Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, fast noise-contrastive estimation for large rnn vocabularies. NAACL.

Unsupervised Word Alignment by Agreement Under ITG Constraint

Hidetaka Kamigaito[1] kamigaito@lr.pi.titech.ac.jp
Akihiro Tamura[2] akihiro.tamura@nict.go.jp
Hiroya Takamura[1] takamura@pi.titech.ac.jp
Manabu Okumura[1] oku@pi.titech.ac.jp
Eiichiro Sumita[2] eiichiro.sumita@nict.go.jp
[1]Tokyo Institute of Technology
[2]National Institute of Information and Communication Technology

Abstract

We propose a novel unsupervised word alignment method that uses a constraint based on Inversion Transduction Grammar (ITG) parse trees to jointly unify two directional models. Previous agreement methods are not helpful for locating alignments with long distances because they do not use any syntactic structures. In contrast, the proposed method symmetrizes alignments in consideration of their structural coherence by using the ITG constraint softly in the posterior regularization framework (Ganchev et al., 2010). The ITG constraint is also compatible with word alignments that are not covered by ITG parse trees. Hence, the proposed method is robust to ITG parse errors compared to other alignment methods that directly use an ITG model. Compared to the HMM (Vogel et al., 1996), IBM Model 4 (Brown et al., 1993), and the baseline agreement method (Ganchev et al., 2010), the experimental results show that the proposed method significantly improves alignment performance regarding the Japanese-English KFTT and BTEC corpus, and in translation evaluation, the proposed method shows comparable or statistical significantly better performance on the Japanese-English KFTT and IWSLT 2007 corpus.

1 Introduction

Word alignment is an important component of statistical machine translation (SMT) systems such as phrase-based SMT (Koehn et al., 2003) and hierarchical phrase-based SMT (Chiang, 2007). In addition, word alignment is utilized for multi-lingual tasks other than SMT, such as bilingual lexicon extraction (Liu et al., 2013). The most conventional approaches to word alignment are the IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996), which align each source word to a single target word (i.e., directional models). In these models, bidirectional word alignments are traditionally induced by combining the Viterbi alignments in each direction using heuristics (Och and Ney, 2003). Matusov et al. (2004) exploited a symmetrized posterior probability for bidirectional word alignments. In these methods, each directional model is independently trained.

Previous researches have improved bidirectional word alignments by jointly training two directional models to agree with each other (Liang et al., 2006; Graça et al., 2008; Ganchev et al., 2010). Such a constraint on the agreement in a training phase is one of the most effective approaches to word alignment. However, none of the previous agreement constraints have taken into account syntactic structures. Therefore, they have difficulty recovering the alignments with long distances, which frequently occur, especially in grammatically different language pairs.

Some unsupervised word alignment models such as DeNero and Klein (2007) and Kondo et al. (2013), have been based on syntactic structures. In particular, it has been proven that Inversion Transduction Grammar (ITG) (Wu, 1997), which captures structural coherence between parallel sentences, helps in word alignment (Zhang and Gildea, 2004; Zhang and Gildea, 2005). However, ITG has not been introduced into an agreement constraint so far.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1998–2004,
Austin, Texas, November 1-5, 2016.

We propose an alignment method that uses an ITG constraint to encourage agreement between two directional models in consideration of their structural coherence. Our ITG constraint is based on the Viterbi alignment decided by a bracketing ITG parse tree, and used as a soft constraint in the posterior regularization framework (Ganchev et al., 2010). In addition, our ITG constraint works also on word alignments that are not covered by ITG parse trees, as a standard symmetric constraint. Hence, the proposed method is robust to ITG parse errors compared to an alignment method that uses an ITG directly in model training (e.g., Zhang and Gildea (2004, 2005)).

Word alignment evaluations show that the proposed method achieves significant gains in F-measure and alignment error rate (AER) on the KFTT (Neubig, 2011) and the BTEC Japanese-English (Ja-En) corpus (Takezawa et al., 2002). Machine translation evaluations show that our constraint significantly outperforms or is comparable to the baseline symmetric constraint (Ganchev et al., 2010) in BLEU on the KFTT Ja-En and IWSLT 2007 Ja-En corpus (Fordyce, 2007).

2 ITG Constraint in the Posterior Regularization Framework

2.1 Overview

The proposed method introduces an ITG constraint into the posterior regularization framework (Ganchev et al., 2010) in model training. The proposed model is trained as follows, where agreement constraints are imposed in the E-step of the EM algorithm[1]:

E-step:

1. Calculate a source-to-target posterior probability $\overrightarrow{p_\theta}(\boldsymbol{z}|\boldsymbol{x})$ and a target-to-source posterior probability $\overleftarrow{p_\theta}(\boldsymbol{z}|\boldsymbol{x})$ for each bilingual sentence $\boldsymbol{x} = \{\boldsymbol{f}, \boldsymbol{e}\}$ under the current model parameters $\boldsymbol{\theta}$, where $\boldsymbol{z}$ denotes an alignment in a sentence pair $\boldsymbol{x}$. In particular, $z_{i,j}$=1, if f_i is aligned to e_j (otherwise $z_{i,j}$=0).
2. Repeat the following steps for all sentence pairs in the training data.
(a) Find the Viterbi alignment $\boldsymbol{z}^*$ through ITG parsing (see Section 2.2). Here, $z^*_{i,j}$=1, if f_i is aligned to e_j (otherwise $z^*_{i,j}$=0).
(b) Symmetrize $\overrightarrow{p_\theta}(\boldsymbol{z}|\boldsymbol{x})$ and $\overleftarrow{p_\theta}(\boldsymbol{z}|\boldsymbol{x})$ under the constraint of $\boldsymbol{z}^*$ (see Section 2.3 and 2.4).

M-step:

1. Estimate all parameters $\boldsymbol{\theta}$ based on the symmetrized posterior probabilities $\overrightarrow{q_\lambda}(\boldsymbol{z}|\boldsymbol{x})$ and $\overleftarrow{q_\lambda}(\boldsymbol{z}|\boldsymbol{x})$ (see Section 2.3 and 2.4).

2.2 ITG Parsing

In this section, we present our ITG parsing method, which uses bracketing ITG (Wu, 1997). The rules of the bracketing ITG are as follows: $A \rightarrow \langle Y/Z \rangle$, $A \rightarrow [Y/Z]$, $A \rightarrow f_i/e_j$, $A \rightarrow f_i/\epsilon$, and $A \rightarrow \epsilon/e_j$, where A, Y, and Z are non-terminal symbols, f_i and e_j are terminal strings, ϵ is a null symbol, $\langle\rangle$ denotes the inversion of two phrase positions, and [] denotes the reversion of two phrase positions.

In general, a bracketing ITG has $O(|\boldsymbol{f}|^3|\boldsymbol{e}|^3)$ time complexity for parsing a sentence pair $\{\boldsymbol{f}, \boldsymbol{e}\}$, where $|\boldsymbol{f}|$ and $|\boldsymbol{e}|$ are the lengths of $\boldsymbol{f}$ and $\boldsymbol{e}$. For efficient ITG parsing, we use the two-step parsing approach (Xiao et al., 2012), which has been proposed to induce Synchronous Context Free Grammar (SCFG) using n-best pruning[2] with time complexity $O(|\boldsymbol{f}|^3)$. Because ITG is a kind of SCFG, this method can be adopted for our ITG parsing. Our two-step parsing first parses a bilingual sentence in the bottom up manner, and then derives the Viterbi alignment $\boldsymbol{z}^*$ in the top down manner.

To parse a bilingual sentence $\boldsymbol{x} = \{\boldsymbol{f}, \boldsymbol{e}\}$, we define the probability for each ITG rule. The probability of a rule $A \rightarrow f_i/e_j$ is defined as:

$$P(A \rightarrow f_i/e_j) = \frac{\overrightarrow{p}_\theta(z_{i,j}=1|\boldsymbol{x}) + \overleftarrow{p}_\theta(z_{i,j}=1|\boldsymbol{x})}{2}.$$

We provide a constant value p_{null}[3] both to $P(A \rightarrow \epsilon/e_j)$ and $P(A \rightarrow f_i/\epsilon)$. To reduce computational cost, the probabilities of phrasal rules $P(A \rightarrow \langle Y/Z \rangle)$ and $P(A \rightarrow [Y/Z])$ are not trained, which are set to 0.5 following Saers et al. (2012). In addition to the probability of each ITG rule, we must provide a probability to an one-to-many alignment because the two step parsing approach must pre-compute probabilities for all one-to-many alignments in the first step. An one-to-many alignment

[1] Step 1 in the E and M steps can be performed in the same way as in Ganchev et al. (2010).

[2] We set n to 30 in our experiments.

[3] We set p_{null} to 10^{-5}.

can be decomposed to a rule $A \rightarrow f_i/e_j$ and some $A \rightarrow \epsilon/e_j$ rules under the ITG form. We select a set of rules with the highest probability for an one-to-many alignment using Viterbi algorithm, which has a complexity of $O(|\boldsymbol{e}|)$.

2.3 Previous Agreement Constraint

This section provides an overview of the previous agreement constraint proposed by Ganchev et al. (2010), which is our baseline. In the posterior regularization framework, source-to-target and target-to-source posterior probabilities $\overrightarrow{p}_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ and $\overleftarrow{p}_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ are replaced with $\overrightarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$ and $\overleftarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$, defined as follows:

$$\overrightarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) = \overrightarrow{p}_{\theta}(\boldsymbol{z}|\boldsymbol{x}) \cdot exp^{(-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}^{\text{agree}}(\boldsymbol{x},\boldsymbol{z}))}/Z_{\overrightarrow{q}},$$
$$\overleftarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) = \overleftarrow{p}_{\theta}(\boldsymbol{z}|\boldsymbol{x}) \cdot exp^{(-\boldsymbol{\lambda}\cdot\boldsymbol{\phi}^{\text{agree}}(\boldsymbol{x},\boldsymbol{z}))}/Z_{\overleftarrow{q}},$$

where $Z_{\overrightarrow{q}}$ is a normalization term for $\sum_{\boldsymbol{z}} \overrightarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x}) = 1$ ($Z_{\overleftarrow{q}}$ is analogous) and $\boldsymbol{\lambda}$ is a vector of weight parameters that controls the balance between two directional posterior probabilities. Here, $\boldsymbol{\phi}^{\text{agree}}$ is a feature of agreement constraint, which assigns each alignment direction to a sign (i.e., +1 or -1). In particular, $\boldsymbol{\phi}^{\text{agree}}$ is defined as follows:

$$\phi^{\text{agree}}_{i,j}(\boldsymbol{x},\boldsymbol{z}) = \begin{cases} +1 & (\boldsymbol{z} \in \overleftarrow{\boldsymbol{Z}}) \wedge (z_{i,j}{=}1), \\ -1 & (\boldsymbol{z} \in \overrightarrow{\boldsymbol{Z}}) \wedge (z_{i,j}{=}1), \\ 0 & otherwise, \end{cases}$$

where $\overrightarrow{\boldsymbol{Z}}$ and $\overleftarrow{\boldsymbol{Z}}$ are sets of possible alignments generated by source-to-target and target-to-source alignment models, respectively. So that $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ and $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ become equal probabilities for each i,j (i.e., $\overrightarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$ and $\overleftarrow{q}_{\boldsymbol{\lambda}}(\boldsymbol{z}|\boldsymbol{x})$ are symmetrical), the agreement constraint is defined as follows:

$$\forall i, \forall j, \overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x}) - \overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x}) = 0. \quad (1)$$

To satisfy the constraint (1), each $\lambda_{i,j}$ is updated by a stochastic gradient descent in the E-step of EM algorithm.

2.4 Proposed ITG Constraint

This section presents the proposed ITG constraint based on the Viterbi alignment $\boldsymbol{z}^*$, which has previously been identified by the bracketing ITG parsing. The ITG constraint uses a feature $\boldsymbol{\phi}^{\text{ITG}}$ instead of $\boldsymbol{\phi}^{\text{agree}}$:

$$\phi^{\text{ITG}}_{i,j}(\boldsymbol{x},\boldsymbol{z}) = \begin{cases} 0 & \overleftarrow{Y}(i,j) \wedge (z^*_{i,j}{=}1) \wedge (\delta_{i,j}(\boldsymbol{x},\boldsymbol{z}){<}0), \\ +1 & \overleftarrow{Y}(i,j) \wedge (z^*_{i,j}{=}1) \wedge (\delta_{i,j}(\boldsymbol{x},\boldsymbol{z}){>}0), \\ -1 & \overrightarrow{Y}(i,j) \wedge (z^*_{i,j}{=}1) \wedge (\delta_{i,j}(\boldsymbol{x},\boldsymbol{z}){<}0), \\ 0 & \overrightarrow{Y}(i,j) \wedge (z^*_{i,j}{=}1) \wedge (\delta_{i,j}(\boldsymbol{x},\boldsymbol{z}){>}0), \\ +1 & \overleftarrow{Y}(i,j) \wedge (z^*_{i,j}{\neq}1), \\ -1 & \overrightarrow{Y}(i,j) \wedge (z^*_{i,j}{\neq}1), \\ 0 & otherwise, \end{cases}$$

where $\overleftarrow{Y}(i,j) = (\boldsymbol{z} \in \overleftarrow{\boldsymbol{Z}}) \wedge (z_{i,j}{=}1)$, $\overrightarrow{Y}(i,j) = (\boldsymbol{z} \in \overrightarrow{\boldsymbol{Z}}) \wedge (z_{i,j}{=}1)$, and $\delta_{i,j}(\boldsymbol{x},\boldsymbol{z}) = \overrightarrow{p}_{\theta}(z_{i,j}{=}1|\boldsymbol{x}) - \overleftarrow{p}_{\theta}(z_{i,j}{=}1|\boldsymbol{x})$. Similarly to $\boldsymbol{\phi}^{\text{agree}}$, $\boldsymbol{\phi}^{\text{ITG}}$ is imposed on $\overrightarrow{q}_{\boldsymbol{\lambda}_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ and $\overleftarrow{q}_{\boldsymbol{\lambda}_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ under the constraint (1). If $z^*_{i,j} \neq 1$, our feature $\phi^{\text{ITG}}_{i,j}$ operates similarly to $\phi^{\text{agree}}_{i,j}$ according to the last three rules. If $z^*_{i,j}{=}1$, $\boldsymbol{\phi}^{\text{ITG}}$ adjusts probabilities of alignments $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ and $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ by increasing the lower probability without decreasing the higher probability according to the first four rules. For example, when $z^*_{i,j} = 1$ and $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}=1|\boldsymbol{x})$ is larger than $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}=1|\boldsymbol{x})$, $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ is increased until $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ equals $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ according to the second and fourth rules. When $z^*_{i,j}{=}1$ and $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ is larger than $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$, $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}{=}1|\boldsymbol{x})$ is increased until $\overleftarrow{q}_{\lambda_{i,j}}(z_{i,j}=1|\boldsymbol{x})$ equals $\overrightarrow{q}_{\lambda_{i,j}}(z_{i,j}=1|\boldsymbol{x})$ according to the first and third rules. As a result, probabilities of word alignments in $\boldsymbol{z}^*$ tend to be higher than those of the other alignments.

Task	Corpus	Train	Dev	Test
Word Alignment	Hansard	1.13M	37	447
	KFTT	330k	653	582
	BTEC	10k	0	10k
Machine Translation	KFTT	330k	1.17k	1.16k
	IWSLT2007	40k	2.5k	489

Table 1: The numbers of parallel sentences for each data set.

3 Evaluation

We compared our proposed ITG constraint (*itg*) with the baseline agreement constraint (Ganchev et al., 2010) (*sym*) on word alignment and machine translation tasks. In word alignment evaluations, we used the French-English (Fr-En) Hansard Corpus (Mihalcea and Pedersen, 2003), Ja-En KFTT[4] (Neubig,

[4]We used the cleaned dataset distributed on the KFTT official web site (http://www.phontron.com/kftt/index.html).

	Hansard Fr-En		KFTT Ja-En		BTEC Ja-En	
Method	F-measure	AER	F-measure	AER	F-measure	AER
HMM+*none*	0.7900	0.0646	0.4623	0.5377	0.4425	0.5575
HMM+*sym*	**0.7923**	**0.0597**	0.4678	0.5322	0.4534	0.5466
HMM+*itg*	0.7869	0.0629	0.4690	0.5310	0.4499	0.5501
IBM Model 4+*none*	0.7780	0.0775	0.5379	0.4621	0.4454	0.5546
IBM Model 4+*sym*	0.7800	0.0693	0.5545	0.4455	0.4761	0.5239
IBM Model 4+*itg*	0.7791	0.0710	**0.5613**	**0.4387**	**0.4809**	**0.5191**

Table 2: Word alignment performance.

Method	KFTT Ja-En	IWSLT2007 Ja-En
HMM+*none*	18.9	46.4
HMM+*sym*	18.9	46.3
HMM+*itg*	19.2	**47.0**
IBM Model 4+*none*	18.8	46.7†
IBM Model 4+*sym*	19.3†	45.9
IBM Model 4+*itg*	**19.4**	46.7

Table 3: Machine translation performance.

2011), and Ja-En BTEC Corpus (Takezawa et al., 2002). We used the first 10K sentence pairs in the training data for the IWSLT 2007 translation task, which were manually annotated with word alignment (Chooi-Ling et al., 2010), as the BTEC Corpus. In translation evaluations, we used the KFTT and Ja-En IWSLT 2007 translation tasks[5].

Table 1 shows each corpus size. In each training data set, all words were lowercased and sentences with over 80 words on either side were removed.

3.1 Word Alignment Evaluation

We measured the performance of word alignment with AER and F-measure (Och and Ney, 2003). We used only sure alignments for calculating F-measure (Fraser and Marcu, 2007)[6]. We introduced *itg* and *sym* into the HMM and IBM Model 4. Training is bootstrapped from IBM Model 1, followed by HMM and IBM Model 4. All models were trained with five consecutive iterations. In the many-to-many alignment extraction, we used the filtering method (Matusov et al., 2004), where a threshold is optimized on the corresponding AER of the baseline model (i.e., HMM+*sym* or IBM Model 4+*sym*)[7].

Table 2 shows the results of word alignment evaluations[8], where *none* denotes that the model has no constraint. In KFTT and BTEC Corpus, *itg* achieved significant improvement against *sym* and *none* on IBM Model 4 ($p \leq 0.05$)[9]. However, in the Hansard Corpus, *itg* shows no improvement against *sym*. This indicates that capturing structural coherence by *itg* yields a significant benefit to word alignment in a linguistically different language pair such as Ja-En. For example, some function words appear more than once in both a source and target sentence, and they are not symmetrically aligned with each other, especially in regards to the Ja-En language pair. Although the baseline methods tend to be unable to align such long-distance word pairs, the proposed method can correctly catch them because *itg* can determine the relation of long-distance words. We discuss more details about the effectiveness of the ITG constraint in Section 4.1.

3.2 Translation Evaluation

We measured translation performance with BLEU (Papineni et al., 2002). All language models are 5-gram and trained using SRILM (Stolcke and others, 2002) on target side sentences in the training data. When extracting phrases, we apply the method proposed by Matusov et al. (2004), where many-to-many alignments are generated based on the averages of the posterior probabilities from two directional models[10].

We used the Moses phrase-based SMT systems (Koehn et al., 2007) for decoding. We set the distortion-limit parameter to infinite[11], and other pa-

[5]BTEC Corpus is a subset of IWSLT 2007. To uniform tokenization, we retokenized all Japanese sentences both in IWSLT 2007 and BTEC Corpus using ChaSen (Asahara and Matsumoto, 2000).

[6]Since there exists no distinction for sure-possible alignments in the KFTT and BTEC data sets, we treat all alignments of them as sure alignments.

[7]We tried values from 0.1 to 1.0 at an interval of 0.1.

[8]The values in bold indicate the best score.

[9]The statistical significance test was performed by the paired bootstrap resampling (Koehn, 2004).

[10]The posterior thresholds were decided in the same way as the word alignment evaluation.

[11]This setting is generally used for Ja-En translation tasks (Murakami et al., 2007).

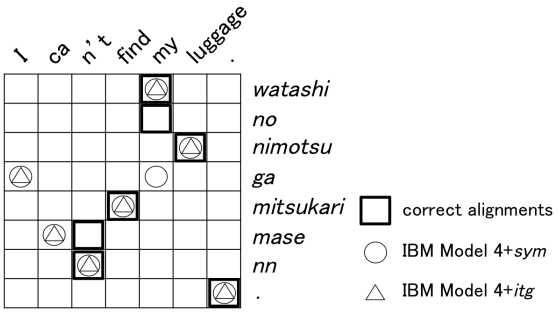


Figure 1: Word alignment examples on the BTEC corpus.

rameters as default settings. Parameter tuning was conducted by 100-best batch MIRA (Cherry and Foster, 2012) with 25 iterations.

Table 3 shows the average BLEU of five different tunings[12]. In both KFTT and IWSLT 2007, *itg* achieved significant improvement against both *none* and *sym* on HMM model. On IBM Model4, *itg* significantly outperforms *none* and is comparable to *sym* in KFTT, while *itg* significantly outperforms *sym* and is comparable to *none* in IWSLT 2007.

4 Discussion

4.1 Effects of ITG Constraints on Word Alignment and Translation

We discuss the effect of our ITG constraint on word alignment and machine translation. As described in Section 2, the ITG constraint is imposed in the E-step of the EM algorithm, not in decoding steps. Therefore, for the sentences that are not contained in the training corpus, the word alignments are calculated using the emission, transition and fertility tables trained with the constraint. It means that the effects of the constraint are implicitly reflected in the alignment results. On the other hand, the effects of the constraint are directly reflected in the machine translation results because the phrase tables are extracted from the posterior probabilities calculated in training steps. Therefore, our ITG constraint has a potential to achieve a large improvement of machine translation performance relative to an improvement of alignment performance, such as IBM Model 4+*itg* vs. IBM Model 4+*sym* on the BTEC corpus. We would like to improve our model by imposing our ITG constraint on decoding steps in future.

[12]The values in bold represent the best score, and † indicates that the comparisons are not significant over the corresponding model (i.e., HMM+*itg* or IBM Model 4+*itg*) according to the bootstrap resampling test ($p \leq 0.05$). We used multeval (Clark et al., 2011) for significance testing.

4.2 Comparison between Symmetric and ITG Constraint

In KFTT, *itg* is comparable to *sym* on IBM Model 4 in machine translation; however, *itg* achieved significant improvement in terms of word alignment, which follows the previous reports that better word alignment does not always result in better translation (Ganchev et al., 2008; Yang et al., 2013). On the other hand, in BTEC, *itg* outperforms *sym* both on word alignment and machine translation. Figure 1 shows that IBM Model 4+*sym* often generates wrong gappy alignments such as "*ga* (Ja)-I (En)" and "*ga* (Ja)-my (En)". These wrong alignments disturb the phrase extraction, because excessively long phrase pairs are extracted by bridging the gaps in wrong alignments or simply no phrase pairs are extracted from wrong gappy alignments. Consequently, the phrase table generated by IBM Model 4+*sym* tend to be sparse and contain longer phrase pairs than the one generated by IBM Model 4+*itg*.

5 Conclusions

We have proposed a novel alignment method that uses an ITG constraint based on bracketing ITG parse trees as a soft constraint of the posterior regularization framework. Due to the ITG constraint, the proposed method can symmetrize two directional alignments based on their structural coherence. Our evaluations have shown that the proposed ITG constraint significantly improves the baseline word alignment performance on the Ja-En KFTT and BTEC corpus, and significantly improves, or at least keeps, the baseline machine translation performance of KFTT and the Ja-En IWSLT 2007 task. This indicates that the proposed method yields a significant benefit to linguistically different language pairs.

In future work, we plan to incorporate a phrasal ITG (Cherry and Lin, 2007) instead of a bracketing ITG to efficiently handle many-to-many alignments.

References

Masayuki Asahara and Yuji Matsumoto. 2000. Extended models and tools for high-performance part-of-speech tagger. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 21–27. Association for Computational Linguistics.

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

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

Colin Cherry and Dekang Lin. 2007. Inversion Transduction Grammar for Joint Phrasal Translation Modeling. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 17–24, Rochester, New York, April. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Goh Chooi-Ling, Watanabe Taro, Yamamoto Hirofumi, and Sumita Eiichiro. 2010. Constraining a generative word alignment model with discriminative output.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Cameron S Fordyce. 2007. Overview of the IWSLT 2007 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation 2007*, pages 1–12.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better Alignments = Better Translations? In *Proceedings of ACL-08: HLT*, pages 986–993, Columbus, Ohio, June. Association for Computational Linguistics.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Joao V Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation Maximization and Posterior Constraints. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 569–576. Curran Associates, Inc.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Shuhei Kondo, Kevin Duh, and Yuji Matsumoto. 2013. Hidden Markov Tree Model for Word Alignment. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 503–511, Sofia, Bulgaria, August. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2013. Topic models + word alignment = a flexible framework for extracting bilingual dictionary from comparable corpus. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 212–221, Sofia, Bulgaria, August. Association for Computational Linguistics.

Evgeny Matusov, Richard Zens, and Hermann Ney. 2004. Symmetric Word Alignments for Statistical Machine Translation. In *Proceedings of COLING 2004, the 20th International Conference on Compu-*

tational Linguistics, pages 219–225, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Rada Mihalcea and Ted Pedersen. 2003. An Evaluation Exercise for Word Alignment. In Rada Mihalcea and Ted Pedersen, editors, *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10.

Jin'ichi Murakami, Tokuhisa Masato, and Satoru Ikehara. 2007. Statistical machine translation using large j/e parallel corpus and long phrase tables. In *Proceedings of the International Workshop on Spoken Language Translation 2007*, pages 151–155.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Markus Saers, Karteek Addanki, and Dekai Wu. 2012. From Finite-State to Inversion Transductions: Toward Unsupervised Bilingual Grammar Induction. In *Proceedings of COLING 2012, the 24th International Conference on Computational Linguistics*, pages 2325–2340, Mumbai, India, December. The COLING 2012 Organizing Committee.

Andreas Stolcke et al. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. Toward a Broad-coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, pages 147–152.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational Linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

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

Xinyan Xiao, Deyi Xiong, Yang Liu, Qun Liu, and Shouxun Lin. 2012. Unsupervised Discriminative Induction of Synchronous Grammar for Machine Translation. In *Proceedings of COLING 2012, the 24th International Conference on Computational Linguistics*, pages 2883–2898, Mumbai, India, December. The COLING 2012 Organizing Committee.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word Alignment Modeling with Context Dependent Deep Neural Network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hao Zhang and Daniel Gildea. 2004. Syntax-Based Alignment: Supervised or Unsupervised? In *Proceedings of COLING 2004, the 20th International Conference on Computational Linguistics*, pages 418–424, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Hao Zhang and Daniel Gildea. 2005. Stochastic Lexicalized Inversion Transduction Grammar for Alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 475–482, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Training with Exploration Improves a Greedy Stack LSTM Parser

Miguel Ballesteros[◇] **Yoav Goldberg**[♣] **Chris Dyer**[♠] **Noah A. Smith**[♡]
[◇]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♣]Computer Science Department, Bar-Ilan University, Ramat Gan, Israel
[♠]Google DeepMind, London, UK
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
miguel.ballesteros@upf.edu, yoav.goldberg@gmail.com,
cdyer@google.com, nasmith@cs.washington.edu

Abstract

We adapt the greedy stack LSTM dependency parser of Dyer et al. (2015) to support a training-with-exploration procedure using dynamic oracles (Goldberg and Nivre, 2013) instead of assuming an error-free action history. This form of training, which accounts for model predictions at training time, improves parsing accuracies. We discuss some modifications needed in order to get training with exploration to work well for a probabilistic neural network dependency parser.

1 Introduction

Natural language parsing can be formulated as a series of decisions that read words in sequence and incrementally combine them to form syntactic structures; this formalization is known as transition-based parsing, and is often coupled with a greedy search procedure (Yamada and Matsumoto, 2003; Nivre, 2003; Nivre, 2004; Nivre, 2008). The literature on transition-based parsing is vast, but all works share in common a classification component that takes into account features of the current parser state[1] and predicts the next action to take conditioned on the state. The state is of unbounded size.

Dyer et al. (2015) presented a parser in which the parser's unbounded state is embedded in a fixed-dimensional continuous space using recurrent neural networks. Coupled with a recursive tree composition function, the feature representation is able to capture information from the entirety of the state, without resorting to locality assumptions that were common in most other transition-based parsers. The use of a novel stack LSTM data structure allows the parser to maintain a constant time per-state update, and retain an overall linear parsing time.

The Dyer et al. parser was trained to maximize the likelihood of gold-standard transition sequences, given words. At test time, the parser makes *greedy* decisions according to the learned model. Although this setup obtains very good performance, the training and testing conditions are mismatched in the following way: at training time the historical context of an action is always derived from the gold standard (i.e., perfectly correct past actions), but at test time, it will be a model prediction.

In this work, we adapt the training criterion so as to explore parser states drawn not only from the training data, but also from the model as it is being learned. To do so, we use the method of Goldberg and Nivre (2012; 2013) to dynamically chose an optimal (relative to the final attachment accuracy) action given an imperfect history. By interpolating between algorithm states sampled from the model and those sampled from the training data, more robust predictions at test time can be made. We show that the technique can be used to improve the strong parser of Dyer et al.

2 Parsing Model and Parameter Learning

Our departure point is the parsing model described by Dyer et al. (2015). We do not describe the model in detail, and refer the reader to the original work. At each stage t of the parsing process, the parser state is

[1]The term "state" refers to the collection of previous decisions (sometimes called the history), resulting partial structures, which are typically stored in a stack data structure, and the words remaining to be processed.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2005–2010,
Austin, Texas, November 1-5, 2016.

encoded into a vector $\mathbf{p}_t$, which is used to compute the probability of the parser action at time t as:

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)}, \quad (1)$$

where $\mathbf{g}_z$ is a column vector representing the (output) embedding of the parser action z, and q_z is a bias term for action z. The set $\mathcal{A}(S, B)$ represents the valid transition actions that may be taken in the current state. Since $\mathbf{p}_t$ encodes information about all previous decisions made by the parser, the chain rule gives the probability of any valid sequence of parse transitions $\boldsymbol{z}$ conditional on the input:

$$p(\boldsymbol{z} \mid \boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{z}|} p(z_t \mid \mathbf{p}_t). \quad (2)$$

The parser is trained to maximize the conditional probability of taking a "correct" action at each parsing state. The definition of what constitutes a "correct" action is the major difference between a static oracle as used by Dyer et al. (2015) and the dynamic oracle explored here.

Regardless of the oracle, our training implementation constructs a computation graph (nodes that represent values, linked by directed edges from each function's inputs to its outputs) for the negative log probability for the oracle transition sequence as a function of the current model parameters and uses forward- and backpropagation to obtain the gradients respect to the model parameters (Lecun et al., 1998, section 4).

2.1 Training with Static Oracles

With a static oracle, the training procedure computes a canonical reference series of transitions for each gold parse tree. It then runs the parser through this canonical sequence of transitions, while keeping track of the state representation $\mathbf{p}_t$ at each step t, as well as the distribution over transitions $p(z_t \mid \mathbf{p}_t)$ which is predicted by the current classifier for the state representation. Once the end of the sentence is reached, the parameters are updated towards maximizing the likelihood of the reference transition sequence (Equation 2), which equates to maximizing the probability of the correct transition, $p(z_{g_t} \mid \mathbf{p_t})$, at each state along the path.

2.2 Training with Dynamic Oracles

In the static oracle case, the parser is trained to predict the best transition to take at each parsing step, assuming all previous transitions were correct. Since the parser is likely to make mistakes at test time and encounter states it has not seen during training, this training criterion is problematic (Daumé III et al., 2009; Ross et al., 2011; Goldberg and Nivre, 2012; Goldberg and Nivre, 2013, *inter alia*). Instead, we would prefer to train the parser to behave optimally even after making a mistake (under the constraint that it cannot backtrack or fix any previous decision). We thus need to include in the training examples states that result from wrong parsing decisions, together with the optimal transitions to take in these states. To this end we reconsider which training examples to show, and what it means to behave optimally on these training examples. The framework of training with exploration using dynamic oracles suggested by Goldberg and Nivre (2012; 2013) provides answers to these questions. While the application of dynamic oracle training is relatively straightforward, some adaptations were needed to accommodate the probabilistic training objective. These adaptations mostly follow Goldberg (2013).

Dynamic Oracles. A *dynamic oracle* is the component that, given a gold parse tree, provides the optimal set of possible actions to take for any valid parser state. In contrast to static oracles that derive a canonical state sequence for each gold parse tree and say nothing about states that deviate from this canonical path, the dynamic oracle is well defined for states that result from parsing mistakes, and they may produce more than a single gold action for a given state. Under the dynamic oracle framework, an action is said to be optimal for a state if the best tree that can be reached after taking the action is no worse (in terms of accuracy with respect to the gold tree) than the best tree that could be reached prior to taking that action.

Goldberg and Nivre (2013) define the arc-decomposition property of transition systems, and show how to derive efficient dynamic oracles for transition systems that are arc-decomposable.[2] Unfortunately, the arc-standard transition system does

[2]Specifically: for every parser configuration $\mathbf{p}$ and group of

not have this property. While it is possible to compute dynamic oracles for the arc-standard system (Goldberg et al., 2014), the computation relies on a dynamic programming algorithm which is polynomial in the length of the stack. As the dynamic oracle has to be queried for each parser state seen during training, the use of this dynamic oracle will make the training runtime several times longer. We chose instead to switch to the arc-hybrid transition system (Kuhlmann et al., 2011), which is very similar to the arc-standard system but is arc-decomposable and hence admits an efficient $O(1)$ dynamic oracle, resulting in only negligible increase to training runtime. We implemented the dynamic oracle to the arc-hybrid system as described by Goldberg (2013).

Training with Exploration. In order to expose the parser to configurations that are likely to result from incorrect parsing decisions, we make use of the probabilistic nature of the classifier. During training, instead of following the gold action, we sample the next transition according to the output distribution the classifier assigns to the current configuration. Another option, taken by Goldberg and Nivre, is to follow the one-best action predicted by the classifier. However, initial experiments showed that the one-best approach did not work well. Because the neural network classifier becomes accurate early on in the training process, the one-best action is likely to be correct, and the parser is then exposed to very few error states in its training process. By sampling from the predicted distribution, we are effectively increasing the chance of straying from the gold path during training, while still focusing on mistakes that receive relatively high parser scores. We believe further formal analysis of this method will reveal connections to reinforcement learning and, perhaps, other methods for learning complex policies.

Taking this idea further, we could increase the number of error-states observed in the training process by changing the sampling distribution so as to bias it toward more low-probability states. We do this by raising each probability to the power of α ($0 < \alpha \leq 1$) and re-normalizing. This transformation keeps the relative ordering of the events, while shifting probability mass towards less frequent events. As we show below, this turns out to be very beneficial for the configurations that make use of external embeddings. Indeed, these configurations achieve high accuracies and sharp class distributions early on in the training process.

The parser is trained to maximize the likelihood of a correct action z_g at each parsing state $\mathbf{p}_t$ according to Equation 1. When using the dynamic oracle, a state $\mathbf{p}_t$ may admit multiple correct actions $\boldsymbol{z_g} = \{z_{g_i}, \ldots, z_{g_k}\}$. Our objective in such cases is the marginal likelihood of all correct actions,[3]

$$p(\boldsymbol{z_g} \mid \mathbf{p}_t) = \sum_{z_{g_i} \in \boldsymbol{z_g}} p(z_{g_i} \mid \mathbf{p}_t). \quad (3)$$

3 Experiments

Following the same settings of Chen and Manning (2014) and Dyer et al (2015) we report results[4] in the English PTB and Chinese CTB-5. Table 1 shows the results of the parser in its different configurations. The table also shows the best result obtained with the static oracle (obtained by rerunning Dyer et al. parser) for the sake of comparison between static and dynamic training strategies.

	English		Chinese	
Method	UAS	LAS	UAS	LAS
Arc-standard (Dyer et al.)	92.40	90.04	85.48	83.94
Arc-hybrid (static)	92.08	89.80	85.66	84.03
Arc-hybrid (dynamic)	92.66	90.43	86.07	84.46
Arc-hybrid (dyn., $\alpha = 0.75$)	92.73	90.60	86.13	84.53
+ pre-training:				
Arc-standard (Dyer et al.)	93.04	90.87	86.85	85.36
Arc-hybrid (static)	92.78	90.67	86.94	85.46
Arc-hybrid (dynamic)	93.15	91.05	87.05	85.63
Arc-hybrid (dyn., $\alpha = 0.75$)	**93.56**	**91.42**	**87.65**	**86.21**

Table 1: Dependency parsing: English (SD) and Chinese.

The score achieved by the dynamic oracle for English is 93.56 UAS. This is remarkable given that the parser uses a completely greedy search procedure. Moreover, the Chinese score establishes the state-of-the-art, using the same settings as Chen and Manning (2014).

arcs A, if each arc in A can be derived from $\mathbf{p}$, then a valid tree structure containing *all* of the arcs in A can also be derived from $\mathbf{p}$. This is a sufficient condition, but whether it is necessary is unknown; hence the question of an efficient, $O(1)$ dynamic oracle for the augmented system is open.

[3] A similar objective was used by Riezler et al (2000), Charniak and Johnson (2005) and Goldberg (2013) in the context of log-linear probabilistic models.

[4] The results on the development sets are similar and only used for optimization and validation.

	Catalan		Chinese		Czech		English		German		Japanese		Spanish	
Method	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Arc-standard, static + PP	89.60	85.45	79.68	75.08	77.96	71.06	91.12	88.69	88.09	85.24	93.10	92.28	89.08	85.03
+ pre-training	–	–	82.45	78.55	–	–	91.59	89.15	88.56	86.15	–	–	90.76	87.48
Arc-hybrid, dyn. + PP	90.45	86.38	80.74	76.52	85.68	79.38	91.62	89.23	89.80	87.29	93.47	92.70	89.53	85.69
+ pre-training	–	–	**83.54**	**79.66**	–	–	**92.22**	**89.87**	**90.34**	**88.17**	–	–	**91.09**	87.95
Y'15	–	–	–	–	85.2	77.5	90.75	88.14	89.6	86.0	–	–	88.3	85.4
A'16 + pre-training	**91.24**	**88.21**	81.29	77.29	**85.78**	**80.63**	91.44	89.29	89.12	86.95	**93.71**	**92.85**	91.01	**88.14**
A'16-beam	92.67	89.83	84.72	80.85	88.94	84.56	93.22	91.23	90.91	89.15	93.65	92.84	92.62	89.95

Table 2: Dependency parsing results. The dynamic oracle uses $\alpha = 0.75$ (selected on English; see Table 1). PP refers to pseudo-projective parsing. Y'15 and A'16 are beam = 1 parsers from Yazdani and Henderson (2015) and Andor et al. (2016), respectively. A'16-beam is the parser with beam larger than 1 by Andor et al. (2016). Bold numbers indicate the best results among the greedy parsers.

The error-exploring dynamic-oracle training always improves over static oracle training controlling for the transition system, but the arc-hybrid system slightly under-performs the arc-standard system when trained with static oracle. Flattening the sampling distribution ($\alpha = 0.75$) is especially beneficial when training with pretrained word embeddings.

In order to be able to compare with similar greedy parsers (Yazdani and Henderson, 2015; Andor et al., 2016)[5] we report the performance of the parser on the multilingual treebanks of the CoNLL 2009 shared task (Hajič et al., 2009). Since some of the treebanks contain nonprojective sentences and arc-hybrid does not allow nonprojective trees, we use the pseudo-projective approach (Nivre and Nilsson, 2005). We used predicted part-of-speech tags provided by the CoNLL 2009 shared task organizers. We also include results with pretrained word embeddings for English, Chinese, German, and Spanish following the same training setup as Dyer et al. (2015); for English and Chinese we used the same pretrained word embeddings as in Table 1, for German we used the monolingual training data from the WMT 2015 dataset and for Spanish we used the Spanish Gigaword version 3. See Table 2.

4 Related Work

Training greedy parsers on non-gold outcomes, facilitated by dynamic oracles, has been explored by several researchers in different ways (Goldberg and Nivre, 2012; Goldberg and Nivre, 2013; Goldberg et al., 2014; Honnibal et al., 2013; Honnibal and Johnson, 2014; Gómez-Rodríguez et al., 2014; Björkelund and Nivre, 2015; Tokgöz and Eryiğit, 2015; Gómez-Rodríguez and Fernández-González, 2015; Vaswani and Sagae, 2016). More generally, training greedy search systems by paying attention to the expected classifier behavior during test time has been explored under the imitation learning and learning-to-search frameworks (Abbeel and Ng, 2004; Daumé III and Marcu, 2005; Vlachos, 2012; He et al., 2012; Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015). Directly modeling the probability of making a mistake has also been explored for parsing (Yazdani and Henderson, 2015). Generally, the use of RNNs to conditionally predict actions in sequence given a history is spurring increased interest in training regimens that make the learned model more robust to test-time prediction errors. Solutions based on curriculum learning (Bengio et al., 2015), expected loss training (Shen et al., 2015), and reinforcement learning have been proposed (Ranzato et al., 2016). Finally, abandoning greedy search in favor of approximate global search offers an alternative solution to the problems with greedy search (Andor et al., 2016), and has been analyzed as well (Kulesza and Pereira, 2007; Finley and Joachims, 2008), including for parsing (Martins et al., 2009).

5 Conclusions

Dyer et al. (2015) presented stack LSTMs and used them to implement a transition-based dependency parser. The parser uses a greedy learning strategy which potentially provides very high parsing speed while still achieving state-of-the-art results. We have demonstrated that improvement by training the greedy parser on non-gold outcomes; dynamic

[5]We report the performance of these parsers in the most comparable setup, that is, with beam size 1 or greedy search.

oracles improve the stack LSTM parser, achieving 93.56 UAS for English, maintaining greedy search.

Acknowledgments

This work was sponsored in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, and in part by NSF CAREER grant IIS-1054319. Miguel Ballesteros was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA). Yoav Goldberg is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI), a Google Research Award and the Israeli Science Foundation (grant number 1555/15).

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. of ICML*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. arXiv:1506.03099.

Anders Björkelund and Joakim Nivre. 2015. Non-deterministic oracles for unrestricted non-projective transition-based dependency parsing. In *Proc. of IWPT*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In *Proc. of ICML*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of ICML*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

T. Finley and T. Joachims. 2008. Training structural SVMs when exact inference is intractable. In *In Proc. of ICML*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

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

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the association for Computational Linguistics*, 2.

Yoav Goldberg. 2013. Dynamic-oracle transition-based parsing with calibrated probabilistic output. In *Proc. of IWPT*.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proc. of ACL*.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proc. of EMNLP*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL*.

He He, Hal Daumé III, and Jason Eisner. 2012. Imitation learning by coaching. In *NIPS*.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proc. of CoNLL*.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proc. of ACL*.

A. Kulesza and F. Pereira. 2007. Structured learning with approximate inference. In *NIPS*.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*.

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

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proc. of ICLR*.

Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and em training. In *Proc. of ACL*.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of AISTAT*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. In *Proc. of ACL*.

Alper Tokgöz and Gülşen Eryiğit. 2015. Transition-based dependency DAG parsing using dynamic oracles. In *Proc. of ACL SRW*.

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

Andreas Vlachos. 2012. An investigation of imitation learning algorithms for structured prediction. In *Proc. of the European Workshop on Reinforcement Learning*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proc. of CoNLL*.

Capturing Argument Relationships for Chinese Semantic Role Labeling

Lei Sha, Tingsong Jiang, Sujian Li, Baobao Chang, Zhifang Sui
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
shalei, tingsong, lisujian, chbb, szf@pku.edu.cn

Abstract

In this paper, we capture the argument relationships for Chinese semantic role labeling task, and improve the task's performance with the help of argument relationships. We split the relationship between two candidate arguments into two categories: (1) Compatible arguments: if one candidate argument belongs to a given predicate, then the other is more likely to belong to the same predicate; (2) Incompatible arguments: if one candidate argument belongs to a given predicate, then the other is less likely to belong to the same predicate. However, previous works did not explicitly model argument relationships. We use a simple maximum entropy classifier to capture the two categories of argument relationships and test its performance on the Chinese Proposition Bank (CPB). The experiments show that argument relationships is effective in Chinese semantic role labeling task.

1 Introduction

Semantic Role Labeling (SRL) is defined as the task to recognize arguments for a given predicate and assign semantic role labels to them. Because of its ability to encode semantic information, there has been an increasing interest in SRL on many languages (Gildea and Jurafsky, 2002; Sun and Jurafsky, 2004). Figure 1 shows an example in Chinese Proposition Bank (CPB) (Xue and Palmer, 2003), which is a Chinese corpus annotated with semantic role labels.

Previous works of Chinese SRL include feature-based approaches and neural network based approaches. Feature-based approaches often extract a large number of handcrafted features from the sentence, and feed these features to statistical classifiers such as CRF, MaxEnt and SVM (Sun and Jurafsky, 2004; Xue, 2008; Ding and Chang, 2008; Ding and Chang, 2009; Sun, 2010). Neural network based approaches usually take Chinese SRL as sequence labeling task and use bidirectional recurrent neural network (RNN) with long-short-term memory (LSTM) to solve the problem (Wang et al., 2015).

However, both of the above two kinds of approaches identify each candidate argument separately without considering the relationship between arguments. We define two categories of argument relationships here: (1) **Compatible arguments**: if one candidate argument belongs to a given predicate, then the other is more likely to belong to the same predicate; (2) **Incompatible arguments**: if one candidate argument belongs to a given predicate, then the other is less likely to belong to the same predicate. For example, in Figure 1, the word "外商"(foreign businessman) and "企业"(entrepreneur) tend to be compatible arguments when the predicate word is "投资"(invest). On the other hand, "企业"(entrepreneur) and " 规定"(rule) are not likely to belong to the same predicate "投资"(invest).

In this paper, we propose to use a quadratic optimization method to explicitly model the relationship between candidate arguments to improve the performance of Chinese SRL. We train a maximum entropy classifier, and then use the classifier to predict argument relationships between any two candidate arguments in a sentence. Experiments show that argument relationships can greatly improve the performance of Chinese SRL.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2011–2016,
Austin, Texas, November 1-5, 2016.

Word:	保护	外商	投资	企业	合法	权益	六	项	规定
	protect	foreign businessman	invest	entrepreneur	legal	profit	six	item	rule
POS:	VV	NN	VV	NN	JJ	NN	CD	M	NN
SRL:	O	S-ARG0	rel	S-arg1	O	O	O	O	O

Figure 1: A sentence with semantic roles labeled from CPB. "rel" represents the predicate, English translation: "Six rules to protect foreign businessman's legal profits when investing entrepreneurs"

2 Related Work

Semantic Role Labeling (SRL) task was first proposed by Gildea and Jurafsky (2002). Previous approaches on Chinese SRL can be classified into two categories: (1) feature-based approaches (2) neural network based approaches.

Among feature-based approaches, Sun and Jurafsky (2004) did the preliminary work on Chinese SRL without any large semantically annotated corpus and produced promising results. Xue and Palmer (2003) proposed Chinese Proposition Bank (CPB), which leads to more complete and systematic research on Chinese SRL (Xue and Palmer, 2005; Xue, 2008; Ding and Chang, 2009). Sun et al. (2009) extended the work of Chen et al. (2006), performed Chinese SRL with shallow parsing, which took partial parses as inputs. Yang and Zong (2014) proposed multi-predicate SRL, which showed improvements both on English and Chinese Proposition Bank.

Neural network based approaches are free of handcrafted features, Collobert and Weston (2008) proposed a convolutional neural network for SRL. Their approach achieved competitive performance on English SRL without requiring task specific feature. Wang et al. (2015) proposed a bidirectional LSTM-RNN for Chinese SRL.

However, most of the aforementioned approaches did not take the compatible arguments and incompatible arguments into account. Inspired by Sha et al. (2016), our approach model the two argument relationships explicitly to achieve a better performance on Chinese SRL.

3 Capturing the Relationship Between Arguments

We found that there are two typical relationships between candidate arguments: (1) Compatible arguments: if one candidate argument belongs to one event, then the other is more likely to belong to the same event; (2) incompatible arguments: if one candidate argument belongs to one event, then the other is less likely to belong to the same event.

We trained a maximum entropy classifier to predict the relationship between two candidate arguments. We choose the following features:

1. PREDICATE: the predicate in the current sentence
2. ARGUMENT DISTANCE: the distance between the two candidate arguments in the sentence
3. Whether the two candidate arguments occur on the same side of the predicate
4. PARENT DEPENDENCY DISTANCE: the distance between the two candidate arguments' parents in the dependency parse tree
5. PARENT POS: if the two candidate arguments share the same parent, take the common parent's POS tag as a feature
6. Whether the two candidate arguments occur on the same side of the common parent if the two candidate arguments share the same parent

All the words in one sentence except for the predicate are candidate arguments. The word pairs in the ground truth Chinese SRL annotation (training data) are extracted as training data. The training examples are generated as follows: For an candidate argument pair, if both of them are labeled as semantic roles, we take it as positive example. For each positive example, we randomly exchange one of the arguments with an irrelevant argument[1] to get a negative example.

[1] an irrelevant argument is in the same sentence with the predicate, but it is not labeled as semantic role

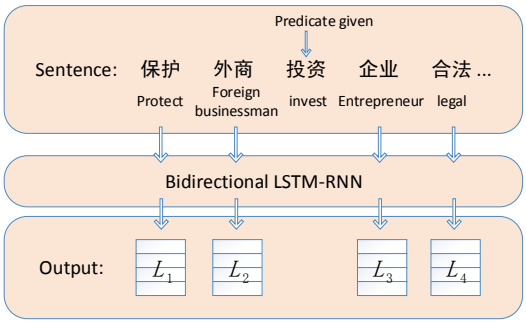


Figure 2: The bidirectional LSTM-RNN architecture.

The chinese sentences should be segmented to chinese words first. For a sentence with $n+1$ words, we denote $C \in \mathbb{R}^{n \times n}$ as the argument relationship matrix. In the testing procedure, the maximum entropy classifier is used to predict the relationship between argument i and argument j as C_{ij}.

When the output of the maximum entropy classifier is around 0.5, it is not easy to figure out whether it is the first relationship or the second, we call this kind of information "uncertain information"(unclear relationship). For a better performance, we **strengthen** the certain information and weaken the uncertain information. We transform the result of maximum entropy classifier as follows:

$$C(i,j) = \begin{cases} 1 & 0.8 < MaxEnt(i,j) \leq 1.0 \\ 0 & 0.2 \leq MaxEnt(i,j) \leq 0.8 \\ -1 & 0.0 \leq MaxEnt(i,j) < 0.2 \end{cases} \quad (1)$$

We set two thresholds, if the output of the maximum entropy classifier is larger than 0.8, we set $C_{i,j} = 1$ (compatible arguments), if the output is lower than 0.2, we set $C_{i,j} = -1$ (incompatible arguments), otherwise, we set $C_{i,j} = 0$ (unclear relationship). The threshold 0.8 and 0.2 are tuned by development set.

4 Quadratic Optimization Method (QOM)

4.1 Post-processing Module of Bidirectional LSTM-RNN

Our quadratic optimization method is a post-processing module of bidirectional LSTM-RNN(Wang et al., 2015). The simplified architecture of bidirectional LSTM-RNN is shown as Figure 2.

Each dimension of the output vector $L_i \in \mathbb{R}^{n_L}, i = 1 \cdots n$ corresponds to the score of a certain semantic role label. n_L represents the number of semantic role labels. Then we normalize L_i over semantic roles as Eq 2 shows.

$$\widetilde{L_i} = Normalize(L_i) \quad (2)$$

Each dimension of $\widetilde{L_i}$ represents the probability of a certain semantic role label.

Let $P_{Arg} \in \mathbb{R}^n$ be a probability vector, each dimension of which represents the probability that the current word has a semantic role as is shown in Eq 3. $P_{Role} \in \mathbb{R}^n$ is another probability vector, each dimension represents the probability of the most likely semantic role the current word may be labeled as is shown in Eq 4.

$$P_{Arg}(i) = \sum_j \widetilde{L_i}(j)[label(j) \neq \text{'0'}] \quad (3)$$

$$P_{Role}(i) = \max_j \widetilde{L_i}(j) \quad (4)$$

where $[\cdot]$ equals to 1 if the inner statement is true and 0 otherwise. $label(j) \neq$ '0' means the j-th word is not labeled with semantic role.

4.2 Quadratic Optimization

We use a n-dim vector X to represent the identification result of candidate arguments. Each entry of X is 0 or 1, 0 represents "noArg", 1 represents "arg". X can be assigned by maximizing $E(X)$ as defined by Eq 5.

$$\begin{aligned} X &= \operatorname*{argmax}_X E(X) \\ E(X) &= \lambda_1 X^T C X + \lambda_2 X^T P_{arg} \\ &\quad + (1 - \lambda_1 - \lambda_2) X^T P_{role} \end{aligned} \quad (5)$$

Here, $X^T C X$ means to add up all the relationship value if the two arguments are identified. Hence, the more the identified arguments are related, the larger the value $X^T C X$ is. $X^T P_{arg}$ is the sum of all chosen arguments probability. $X^T P_{role}$ is the sum of all the classified roles' probability.

Eq 5 means that, while we should select the semantic role with a larger probability, the argument relationship evaluation should also as large as possible.

We use Beam Search method (Algorithm 1) to search for the optimal assignment X. The hyper-parameters λ_1 and λ_2 can be chosen according to development set.

Input: Argument relationship matrix: C
the argument probabilities required by P^{arg}_{sum}
the role probabilities required by P^{role}_{sum}
Data: K: Beam size
n: Number of candidate arguments
Output: The best assignment X
Set beam $\mathcal{B} \leftarrow [\epsilon]$;
for $i \leftarrow 1 \cdots n$ **do**
  buf$\leftarrow \{z' \circ l | z' \in \mathcal{B}, l \in \{0, 1\}\}$;
  $\mathcal{B} \leftarrow [\epsilon]$;
  while $j \leftarrow 1 \cdots K$ **do**
    $x_{best} = \text{argmax}_{x \in \text{buf}} E(x)$;
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_{best}\}$;
    buf$\leftarrow$buf$-\{x_{best}\}$;
  end
end
Sort $\mathcal{B}$ descendingly according to $E(X)$;
return $\mathcal{B}[0]$;

Algorithm 1: Beam Search decoding algorithm for SRL. $\circ$ means to concatenate an element to the end of a vector.

5 Experiment

We conduct experiments to compare our model with previous landmark methods on the benchmark dataset CPB for Chinese SRL. We use Wang et al. (2015)'s model as baseline. The result reveals that our quadratic optimization method can further improve the result of bidirectional LSTM-RNN.

5.1 Experiment Settings

We conduct experiments on the standard benchmark dataset CPB 1.0[2]. We follow the same data setting as previous work (Xue, 2008; Sun et al., 2009), which divided the dataset into three parts: 648 files (from `chtb_081.fid` to `chtb_899.fid`) are used as the training set. The development set includes 40 files, from `chtb_041.fid` to `chtb_080.fid`. The test set includes 72 files, which are `chtb_001.fid` to `chtb_040.fid`, and `chtb_900.fid` to `chtb_931.fid`.

Method	$F_1(\%)$
Xue (2008)	71.90
Collobert and Weston (2008)	74.05
Sun et al. (2009)	74.12
Yang and Zong (2014)	75.31
Wang et al. (2015)	77.21
QOM - stengthen	76.24
QOM - feature 4,5,6	77.52
QOM	77.69

Table 1: Results comparison on CPB dataset.

The training dataset of the argument relationship matrix contains 1.6M cases (736K positive and 864K negative) which are randomly generated according to the ground truth in the training documents. We use Stanford Parser[3] for dependency parsing.

We tuned the coefficients λ_1 and λ_2 of Eq 5 on the development set, and finally we set $\lambda_1 = 0.10$ and $\lambda_2 = 0.45$.

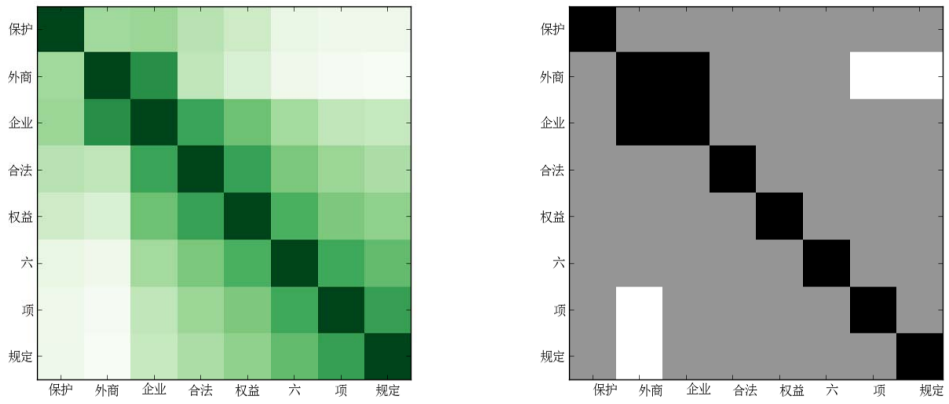
5.2 Chinese SRL Performance

Table 1 shows our SRL performance compared to previous landmark results. We can see that with quadratic optimization method as the post-processing module, our approach (QOM) outperforms Wang et al. (2015) by a large margin (Wilcoxon Signed Rank Test, $p < 0.05$). We also did some ablation test, in Table 1, "QOM - stengthen" is the result when we do not strengthen the argument relationship matrix. We can see that the uncertain information is very harmful to the performance, which worsen the accuracy for about 1%. "QOM - feature 4,5,6" is the performance when we do not use the dependency features when capturing the argument relationships since Wang et al. (2015) didn't use any dependency feature. We can see that event without dependency feature, our method still can outperform Wang et al. (2015)'s result.

Figure 3 visualizes the candidate argument relationship matrix. From this graph, we captured the compatible arguments ("外 商$_{\text{foreign businessman}}$" and "企 业$_{\text{entrepreneur}}$"), incompatible arguments ("项$_{\text{item}}$" and "外 商$_{\text{foreign businessman}}$"), ("规 定$_{\text{rule}}$"

[2]https://catalog.ldc.upenn.edu/LDC2005T23

[3]http://nlp.stanford.edu/software/lex-parser.shtml



Chinese	保护	外商	企业	合法	权益	六	项	规定
English translation	protect	foreign businessman	entrepreneur	legal	profit	six	item	rule

Figure 3: The Visualization of argument relationship Matrix, Left is the origin matrix. Right is the strengthened matrix. In the origin matrix, we can directly see the argument relationship we captured (the darker green means stronger relationship, lighter green means weaker relationship). After strengthening, on the right, the words with strong relationship are classified as compatible arguments (the black squares), weak relationship are classified as incompatible arguments (the white squares). Others (the grey squares) are unclear relationship.

and "外商$_{\text{foreign businessman}}$"). Therefore, "外商$_{\text{foreign businessman}}$" and "企业$_{\text{entrepreneur}}$" should be the roles of "投资$_{\text{invest}}$" simultaneously , ("项$_{\text{item}}$", "外商$_{\text{foreign businessman}}$") and ("规定$_{\text{rule}}$", "外商$_{\text{foreign businessman}}$") should not be the roles of "投资$_{\text{invest}}$" simultaneously.

6 Conclusion

In this paper, we propose to use a quadratic optimization method based on two kinds of argument relationships to improve the performance of Chinese SRL. We first train a maximum entropy classifier to capture the compatible arguments and incompatible arguments. Then we use quadratic optimization to improve the result of bidirectional LSTM-RNN(Wang et al., 2015). The experiment has proved the effectiveness of our approach. This method can also be used in other probabilistic methods.

Acknowledgements

We would like to thank our three anonymous reviewers for their helpful advice on various aspects of this work. This research was supported by the National Key Basic Research Program of China (No.2014CB340504) and the National Natural Science Foundation of China (No.61375074,61273318). The contact author for this paper is Baobao Chang and Zhifang Sui.

References

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 97–104. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Weiwei Ding and Baobao Chang. 2008. Improving chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 324–333. Association for Computational Linguistics.

Weiwei Ding and Baobao Chang. 2009. Word based chinese semantic role labeling with semantic chunking. *International Journal of Computer Processing Of Languages*, 22(02n03):133–154.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Rbpb: Regularization-based pattern balancing method for event extraction.

In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234, Berlin, Germany, August. Association for Computational Linguistics.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of chinese. In *Proceedings of NAACL-HLT*, volume 2004.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1475–1483. Association for Computational Linguistics.

Weiwei Sun. 2010. Improving chinese semantic role labeling with rich syntactic features. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 168–172. Association for Computational Linguistics.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1626–1631.

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the penn chinese treebank. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 47–54. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics*, 34(2):225–255.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *EMNLP*, pages 363–373.

BrainBench:
A Brain-Image Test Suite for Distributional Semantic Models

Haoyan Xu
University of Victoria
Victoria, BC, Canada
exu@uvic.ca

Brian Murphy
Queen's University Belfast
Belfast, Northern Ireland, UK
brian.murphy@qub.ac.uk

Alona Fyshe
University of Victoria
Victoria, BC, Canada
afyshe@uvic.ca*

Abstract

The brain is the locus of our language ability, and so brain images can be used to ground linguistic theories. Here we introduce BrainBench, a lightweight system for testing distributional models of word semantics. We compare the performance of several models, and show that the performance on brain-image tasks differs from the performance on behavioral tasks. We release our benchmark test as part of a web service.

1 Introduction

There is active debate over how we should test semantic models. In fact, in 2016 there was an entire workshop dedicated to the testing of semantic representations (RepEval, 2016). Several before us have argued for the usage of brain data to test semantic models (Anderson et al., 2013; Murphy et al., 2012; Anderson et al., 2015), as a brain image represents a snapshot of one person's own semantic representation. Still, testing semantic models against brain imaging data is rarely done by those not intimately involved in psycholinguistics or neurolinguistics. This may be due to a lack of familiarity with neuroimaging methods and publicly available datasets.

We present the first iteration of BrainBench, a new system that makes it easy to test semantic models using brain imaging data (Available at `http://www.langlearnlab.cs.uvic.ca/brainbench/`). Our system has methodology that is similar to popular tests based on behavioral data (see Section 2.2), and has the additional benefit of being fast enough to offer as a web service.

*Corresponding Author

2 The Tasks

Here we outline the set of tasks we used to evaluate several popular Distributional Semantic (DS) models.

2.1 Brain Image Data

For BrainBench we use two brain image datasets collected while participants viewed 60 concrete nouns with line drawings (Mitchell et al., 2008; Sudre et al., 2012). One dataset was collected using fMRI (Functional Magnetic Resonance Imaging) and one with MEG (Magnetoencephalography). Each dataset has 9 participants, but the participants sets are disjoint, thus there are 18 unique participants in all. Though the stimuli is shared across the two experiments, as we will see, MEG and fMRI are very different recording modalities and thus the data are not redundant.

fMRI measures the change in blood oxygen levels in the brain, which varies according to the amount of work being done by a particular brain area. An fMRI image is a 3D volume of the brain where each point in the volume (called a voxel) represents brain activity at a particular place in the brain. In the fMRI dataset used here, each voxel represents a 3mm x 3mm x 5mm area of the brain. Each of the 60 words was presented 6 times in random order, for a total of 360 brain images. The number of voxels depends on the size and shape of a person's brain, but there are around 20,000 voxels per participant in this dataset.

MEG measures the magnetic field caused by

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2017–2021,
Austin, Texas, November 1-5, 2016.

many neurons firing in the same direction at the same time. This signal is very weak, and so must be measured in a magnetically shielded room. The MEG machine is essentially a large helmet with 306 sensors that measure aspects of the magnetic fields at different locations in the brain. A MEG brain image is the time signals recorded from each of these sensors. Here, the sampling rate is 200 Hz. For each word, the MEG recording is 800ms long resulting in 306×160 data points. Each of the words was presented 20 times (in random order) for a total of 1200 brain images. For simplicity we will use the term "brain image feature" to refer to both fMRI voxels and MEG sensor/time points.

A non-trivial portion of our participants' brain activity may be driven by the low-level visual properties of the word/line-drawing stimulus, rather than by semantics. As there is a possibility of confounding visual properties with semantic properties, we have attempted to remove the activity attributable to visual properties from the brain images. In total we have 11 visual features which include things like the length of the word, the number of white pixels, and features of the line drawing (Sudre et al., 2012). To remove the visual stimulus' contribution to the signal, we train a regression model that predicts the signal in each brain image feature as a function of the 11 visual features. We then subtract the predicted value from the observed value of the brain image feature. This process is known as "partialling out" an effect. Thus, the signal that remains in the brain image will not be correlated with the visual stimuli, and should only be related to the semantics of the word itself (or noise).

Brain images are quite noisy, so we used the methodology from Mitchell et al. (2008) to select the most stable brain image features for each of the 18 participants. The stability metric assigns a high score to features that show strong self-correlation over presentations of the same word. We noticed that tuning the number of features to keep made little or no difference in the absolute ordering of the different DS models. Thus, we use the optimal number of features averaged over all 6 DS models described in Section 3: the top 13% of MEG sensor/time points, and 3% of fMRI voxels. Finally, we average all brain images corresponding to repetitions of the same word.

2.2 Behavioral Tasks

We include, for comparison, four popular word vector evaluation benchmarks.

MEN This dataset contains 3,000 word pairs, such that each word appears frequently in two separate corpora. Human participants were presented with two word pairs and asked to choose the word pair that was more related, resulting in a ranking of relatedness amongst word pairs (Bruni and Baroni, 2013).

SimLex-999 A word pairing task meant to specifically target similarity rather than the more broad "relatedness" (Hill et al., 2015).

WS-353-[SIM|REL] A set of 353 word pairs with relatedness ratings (Finkelstein et al., 2002). This dataset was subsequently split into sets where the pairs denote similarity and relatedness, named WS-353-SIM and WS-353-REL, respectively (Agirre et al., 2009).

3 Distributional Models

We test six semantic models against both the fMRI and behavioral datasets. The six models are:

Skip-gram: A neural network trained to predict the words before and after the current word, given the current word. We selected a model with 300 dimensions trained on the Google news corpus (Mikolov et al., 2013).

Glove: A regression-based model that combines global context information (term-document cooccurrence) with local information (small windows of word-word cooccurrence) (Pennington et al., 2014). This 300-dimensional model was trained on the Wikipedia and Gigaword 5 corpora combined.

RNN: A recurrent neural network with 640-dimensional hidden vectors. These models are trained to predict the next word in a sequence and have the ability to encode (theoretically) infinitely distant contextual information (Mikolov et al., 2011). The model was trained on broadcast news transcriptions.

Global: A neural network model that incorporates global and local information, like that of the Glove

model (Huang et al., 2012). This model is our smallest, with dimension 50, and was trained on Wikipedia.

Cross-lingual: A tool that projects distributional representations from multiple language into a shared representational space (Faruqui and Dyer, 2014). Here we use the German-English model (512 dimensions), trained on the WMT-2011 corpus.

Non-distributional: This model is based solely on hand-crafted linguistic resources (Faruqui and Dyer, 2015). Several resources like WordNet (Fellbaum, 1998) and FrameNet (Baker et al., 1998) are combined to make very sparse word vector representations. Due to their sparsity, these vectors are of very high dimension ($171,839$). This is a particularly interesting model because it is not built from a corpus (unlike every other model in this list).

Note that we are not aiming to compare the goodness of any of these distributional models, as they are trained on different corpora with different algorithms. Instead, we wish to compare the patterns of performance on behavioral benchmarks to that of a brain-image based task.

4 Methodology

Each of the behavioral tasks included here assigns a similarity score to word pairs. For each DS model we calculate the correlation between the vectors for every pair of words in the behavioral datasets. We then calculate the correlation between the DS vector correlations and the behavioral scores.

We follow a very similar methodology for the brain image datasets. Let us represent each DS model with a matrix $X \in \mathbb{R}^{w \times p}$ where w is the number of words for which we have brain images (here $w = 60$), and p is the number of dimensions in a particular DS model. From X we calculate the correlation between each pair of word vectors, resulting in a matrix $C_{DS} \in \mathbb{R}^{w \times w}$.

Let us represent each participant's brain images with a matrix $Y \in \mathbb{R}^{w \times v}$ where v is the number of selected brain image features. From this matrix we calculate the correlation between each pair of brain images, resulting in a matrix $C_{BI} \in \mathbb{R}^{w \times w}$ (BI for brain image). This final representation is similar to the behavioral tasks above, but now we have a similarity measure for *every* pair of words in our dataset.

Here is where the evaluation for brain imaging tasks differs from the behavioral tasks. Instead of measuring the correlation between C_{BI} and C_{DS}, as is done in Representational Similarity Analysis (RSA) (Kriegeskorte et al., 2008), we use the testing methodology from Mitchell et al. (2008), which we will refer to as the 2 vs. 2 test. The 2 vs. 2 test was developed to help detect statistically significant predictions on brain imaging data, and, compared to RSA, can better differentiate the performance of a model from chance. We perform a 2 vs. 2 test for all pairs of C_{DS} and C_{BI} (that is, for every pair of DS model and fMRI/MEG participant).

For each 2 vs. 2 test we select the same two words (rows) w_1, w_2 from C_{DS} and C_{BI}. We omit the columns which correspond to the correlation to w_1 and w_2, as they contain a perfect signal for the 2 vs. 2 test. We now have four vectors, $C_{DS}(w_1)$, $C_{DS}(w_2)$, $C_{BI}(w_1)$ and $C_{BI}(w_2)$, all of length $w - 2$. We compute the correlation (corr) between vectors derived from C_{DS} and C_{BI} to see if:

$$\text{corr}(C_{DS}(w_1), C_{BI}(w_1)) + \text{corr}(C_{DS}(w_2), C_{BI}(w_2))$$

(the correlation of correctly matched rows: w_1 to w_1 and w_2 to w_2) is greater than:

$$\text{corr}(C_{DS}(w_1), C_{BI}(w_2)) + \text{corr}(C_{DS}(w_2), C_{BI}(w_1))$$

(the correlation of incorrectly matched rows). If the correctly matched rows are more similar than incorrectly matched rows, then the 2 vs. 2 test is considered correct. We perform the 2 vs. 2 test for all possible pairs of words, for 1770 tests in total. The 2 vs. 2 accuracy is the percentage of 2 vs. 2 tests correct. Chance is 50%.

Our process of computing 2 vs. 2 accuracy over rows of a correlation matrix is different than the original methodology for these datasets (Mitchell et al., 2008; Sudre et al., 2012). Previous work trained regression models that took brain images as input and predicted the dimensions of a DS model as output. Training these regression models for all 1770 pairs of words takes hours to complete, whereas the test we suggest here is much faster, and the correlation matrices C_{BI} can be computed ahead of time. This makes the tests fast enough to offer as a web service. We hope our web offering will remove barriers to the wider adoption of brain-based tests from within the computational linguistics community.

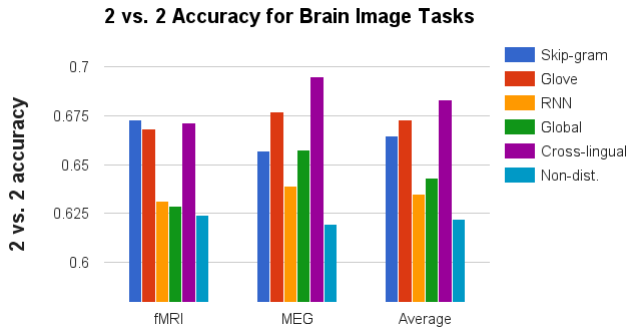


Figure 1: Performance of Distributional Semantic models on the brain-image datasets.

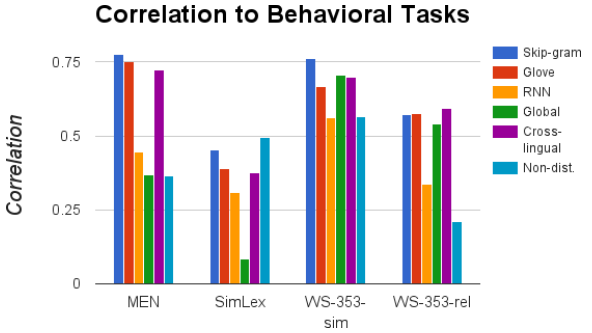


Figure 2: Performance of Distributional Semantic models on several benchmark behavioral tasks.

5 Results

Figure 1 shows the results for each of the DS models against the fMRI and MEG datasets. On average, the Skip-gram, Glove and Cross-lingual models perform quite well, whereas the multi-layer NNs (RNN, Global) perform less well. The one DS model to be built from hand-crafted resources (Non-distributional) performs poorly on both brain image tests.

As previously mentioned, we are not claiming to show that any one of the DS models is better than any other. Indeed, that would be comparing apples to oranges, as each DS model is trained with a different algorithm on a different corpus. Instead, notice that the *pattern* of performance for the fMRI task is remarkably similar to the pattern on the MEN behavioral task. This is interesting given that our dataset contains only 60 words and the MEN dataset contains > 700. On the MEG data, the Cross-lingual model performs best, and its performance pattern is unlike any of the behavioral tasks in Figure 2. The averaged BrainBench results are most similar to the results for WS-353-REL. However, averaging the results together may be misleading, as the fMRI and MEG result patterns are different.

6 Discussion

There are some caveats about the analyses herein. Firstly, the brain-based tests include only 60 concrete nouns, so they will necessarily favor distributional models with good noun representations, regardless of the representations of other parts of speech. We are currently working with various research groups to expand the number of brain-image datasets included in this benchmark to have a more diverse test base. The behavioral benchmarks were not reduced to include only the 60 words for which we have brain data, because this would have rendered the benchmarks essentially useless, as very rarely are a pair of the 60 words from the brain image data scored as a pair in the behavioral benchmarks.

7 Conclusion

We have presented our new system, BrainBench, which is a fast and lightweight alternative to previous methods for comparing DS models to brain images. Our proposed methodology is more similar to well-known behavioral tasks, as BrainBench also uses the similarity of words as a proxy for meaning. We hope that this contribution will bring brain imaging tests "to the masses" and encourage discussion around the testing of DS models against brain imaging data.

References

[Agirre et al.2009] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pas, and Aitor Soroa. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages 19–27.

[Anderson et al.2013] Andrew J Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio, and Marco Baroni. 2013. Of words , eyes and brains : Correlating image-based distributional semantic models with neural representations of concepts. In *Proceedings of*

the Conference on Empirical Methods on Natural Language Processing.

[Anderson et al.2015] Andrew James Anderson, Elia Bruni, Alessandro Lopopolo, Massimo Poesio, and Marco Baroni. 2015. Reading visually embodied meaning from the brain: Visually grounded computational models decode visual-object mental imagery induced by written text. *NeuroImage*, 120:309–322.

[Baker et al.1998] Collin F. Cf Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th annual meeting on Association for Computational Linguistics -*, volume 1, page 86. Association for Computational Linguistics.

[Bruni and Baroni2013] Elia Bruni and Marco Baroni. 2013. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, 48.

[Faruqui and Dyer2014] Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. *Proceedings of the European Association for Computational Linguistics*, pages 462–471.

[Faruqui and Dyer2015] Manaal Faruqui and Chris Dyer. 2015. Non-distributional Word Vector Representations. *Acl-2015*, pages 464–469.

[Fellbaum1998] Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

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

[Hill et al.2015] Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics*, 41(4):665–695.

[Huang et al.2012] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882.

[Kriegeskorte et al.2008] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2(November):4, jan.

[Mikolov et al.2011] Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011. RNNLM — Recurrent Neural Network Language Modeling Toolkit. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU)*, pages 1–4.

[Mikolov et al.2013] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

[Mitchell et al.2008] Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science (New York, N.Y.)*, 320(5880):1191–5, may.

[Murphy et al.2012] Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting Corpus-Semantic Models for Neurolinguistic Decoding. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 114–123, Montreal, Quebec, Canada.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe : Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar.

[RepEval2016] RepEval. 2016. RepEval workshop, ACL. `https://sites.google.com/site/repevalacl16/`.

[Sudre et al.2012] Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell. 2012. Tracking Neural Coding of Perceptual and Semantic Features of Concrete Nouns. *NeuroImage*, 62(1):463–451, may.

Evaluating Induced CCG Parsers on Grounded Semantic Parsing

Yonatan Bisk[1*] **Siva Reddy**[2*] **John Blitzer**[3] **Julia Hockenmaier**[4] **Mark Steedman**[2]
[1]ISI, University of Southern California
[2]ILCC, School of Informatics, University of Edinburgh
[3]Google, Mountain View
[4]Department of Computer Science, University of Illinois at Urbana-Champaign
ybisk@isi.edu, siva.reddy@ed.ac.uk, blitzer@google.com,
juliahmr@illinois.edu, steedman@inf.ed.ac.uk,

Abstract

We compare the effectiveness of four different syntactic CCG parsers for a semantic slot-filling task to explore how much syntactic supervision is required for downstream semantic analysis. This extrinsic, task-based evaluation also provides a unique window into the semantics captured (or missed) by unsupervised grammar induction systems.

1 Introduction

The past several years have seen significant progress in unsupervised grammar induction (Carroll and Charniak, 1992; Yuret, 1998; Klein and Manning, 2004; Spitkovsky et al., 2010; Garrette et al., 2015; Bisk and Hockenmaier, 2015). But how useful are unsupervised syntactic parsers for downstream NLP tasks? What phenomena are they able to capture, and where would additional annotation be required? Instead of standard intrinsic evaluations – attachment scores that depend strongly on the particular annotation styles of the gold treebank – we examine the utility of unsupervised and weakly supervised parsers for semantics. We perform an extrinsic evaluation of unsupervised and weakly supervised CCG parsers on a grounded semantic parsing task that will shed light on the extent to which these systems recover semantic information. We focus on English to perform a direct comparison with supervised parsers (although unsupervised or weakly supervised approaches are likely to be most beneficial for domains or languages where supervised parsers are not available).

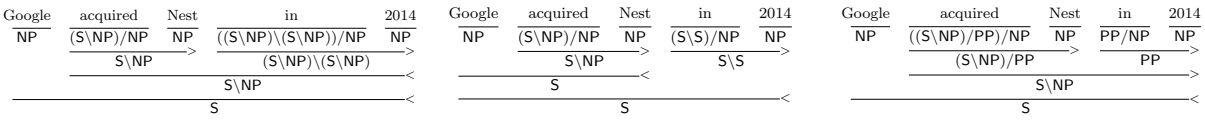
[*]Equal contribution

Specifically, we evaluate different parsing scenarios with varying amounts of supervision. These are designed to shed light on the question of how well syntactic knowledge correlates with performance on a semantic evaluation. We evaluate the following scenarios (all of which assume POS-tagged input): 1) no supervision; 2) a lexicon containing words mapped to CCG categories; 3) a lexicon containing POS tags mapped to CCG categories; 4) sentences annotated with CCG derivations (i.e., fully supervised). Our evaluation reveals which constructions are problematic for unsupervised parsers (and annotation efforts should focus on). Our results indicate that unsupervised syntax is useful for semantics, while a simple semi-supervised parser outperforms a fully unsupervised approach, and could hence be a viable option for low resource languages.

2 CCG Intrinsic Evaluations

CCG (Steedman, 2000) is a lexicalized formalism in which words are assigned syntactic types, also known as *supertags*, encoding subcategorization information. Consider the sentence *Google acquired Nest in 2014*, and its CCG derivations shown in Figure 1. In (a) and (b), the supertag of *acquired*, $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$, indicates that it has two arguments, and the prepositional phrase *in 2014* is an adjunct, whereas in (c) the supertag $((\mathsf{S}\backslash\mathsf{NP})/\mathsf{PP})/\mathsf{NP}$ indicates *acquired* has three arguments including the prepositional phrase. In (a) and (b), depending on the supertag of *in*, the derivation differs. When trained on labeled treebanks, (a) is preferred. However note that all these derivations could lead to the same semantics (e.g., to the logical form in Equation 1). Without syntactic su-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2022–2027,
Austin, Texas, November 1-5, 2016.



(a) ***in 2014*** modifies ***acquired Nest*** (b) ***in 2014*** modifies ***Google acquired Nest*** (c) ***acquired Google*** takes the argument ***in 2014***

Figure 1: Example of multiple valid derivations that can be grounded to the same Freebase logical form (Eq. 1) even though they differ dramatically in performance under parsing metrics (5, 4, or 3 "correct" supertags).

pervision, there may not be any reason for the parser to prefer one analysis over the other. One procedure to evaluate unsupervised induction methods has been to compare the assigned supertags to treebanked supertags, but this evaluation does not consider that multiple derivations could lead to the same semantics. This problem is also not solved by evaluating syntactic dependencies. Moreover, while many dependency standards agree on the head direction of simple constituents (e.g., noun phrases) they disagree on the most semantically useful ones (e.g., coordination and relative clauses).[1]

3 Our Proposed Evaluation

The above syntax-based evaluation metrics conceal the real performance differences and their effect on downstream tasks. Here we propose an extrinsic evaluation where we evaluate our ability to convert sentences to Freebase logical forms starting via CCG derivations. Our motivation is that most sentences can only have a single realization in Freebase, and any derivation that could lead to this realization is potentially a correct derivation. For example, the Freebase logical form for the example sentence in Figure 1 is shown below, and none of its derivations are penalized if they could result in this logical form.

$$\begin{aligned} &\lambda e.\,\text{business.acquisition}(e) \\ &\wedge\,\text{acquiring\_company}(e, \textsc{Google}) \\ &\wedge\,\text{company\_acquired}(e, \textsc{Nest}) \\ &\wedge\,\text{date}(e, 2014) \end{aligned} \tag{1}$$

Since grammar induction systems are traditionally trained on declarative sentences, we would ideally require declarative sentences paired with Freebase logical forms. But such datasets do not exist in the Freebase semantic parsing literature (Cai and Yates, 2013; Berant et al., 2013). To alleviate this problem, and yet perform Freebase semantic parsing, we propose an entity slot-filling task.

Entity Slot-Filling Task. Given a declarative sentence containing mentions of Freebase entities, we randomly remove one of the mentions to create a blank slot. The task is to fill this slot by translating the declarative sentence into a Freebase query. Consider the following sentence where the entity *Nest* has been removed:

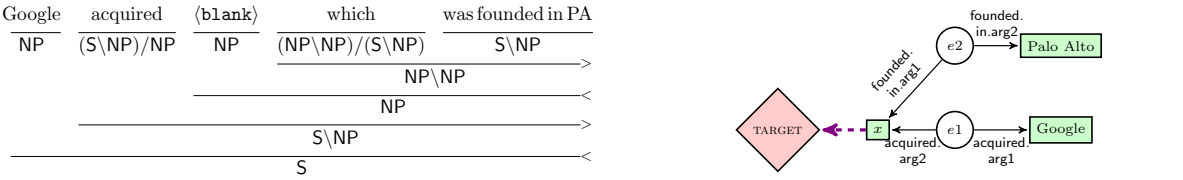
Google acquired _____ which was founded in Palo Alto

To correctly fill in the blank, one has to query Freebase for the entities acquired by *Google* (constraint 1) and founded in *Palo Alto* (constraint 2). If either of those constraints are not applied, there will be many entities as answers. For each question, we execute a single Freebase query containing all the constraints and retrieve a list of answer entities. From this list, we pick the first entity as our predicted answer, and consider the prediction as correct if the gold answer is the same as the predicted answer.

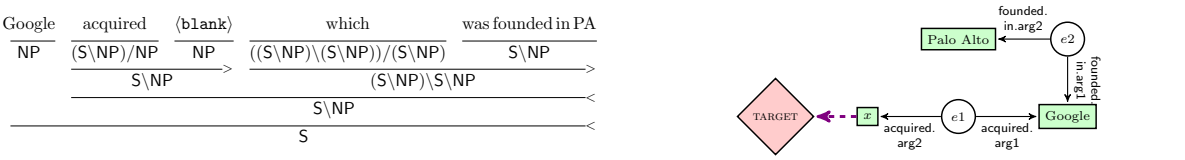
4 Sentences to Freebase Logical Forms

CCG provides a clean interface between syntax and semantics, i.e. each argument of a words syntactic category corresponds to an argument of the lambda expression that defines its semantic interpretation (e.g., the lambda expression corresponding to the category $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$ of the verb *acquired* is $\lambda f.\lambda g.\lambda e.\exists x.\exists y.\text{acquired}(e) \wedge f(x) \wedge g(y) \wedge arg_1(e,y) \wedge arg_2(e,x)$), and the logical form for the complete sentence can be constructed by composing word level lambda expressions following the syntactic derivation (Bos et al., 2004). In Figure 2 we show two syntactic derivations for the same sentence, and the corresponding logical forms and equivalent graph representations derived by GRAPHPARSER (Reddy et al., 2014). The graph representations are possible because GRAPHPARSER assumes access to coindexations of input CCG categories. We provide

[1] Please see Bisk and Hockenmaier (2013) for more details.



$\lambda e_1.\exists x e_2.\ \text{TARGET}(x) \wedge \text{acquired}(e_1) \wedge arg_1(e_1, \text{Google}) \wedge arg_2(e_1, x) \wedge \text{founded}(e_2) \wedge \mathbf{arg_2}(\mathbf{e_2}, x) \wedge \text{in}(e_2, \text{PaloAlto})$



$\lambda e_1.\exists x e_2.\ \text{TARGET}(x) \wedge \text{acquired}(e_1) \wedge arg_1(e_1, \text{Google}) \wedge arg_2(e_1, x) \wedge\ \text{founded}(e_2) \wedge \mathbf{arg_2}(\mathbf{e_2}, \text{Google}) \wedge \text{in}(e_2, \text{PaloAlto})$

Figure 2: The lexical categories for *which* determine the relative clause attachment and therefore the resulting ungrounded logical form. The top derivation correctly executes a query to retrieve companies founded in Palo Alto and acquired by Google. The bottom incorrectly asserts that Google was founded in Palo Alto.

co-indexation for all induced categories, including multiple co-indexations when an induced category is ambiguous. For example, $(S\backslash N)/(S\backslash N)$ refers to either $(S_x\backslash N_y)/(S_x\backslash N_y)$ indicating an auxiliary verb or $(S_x\backslash N_y)/(S_z\backslash N_y)$ indicating a control verb. Initially, the predicates in the expression/graph will be based entirely on the surface form of the words in the sentence. This is the "*ungrounded*" semantic representation.

Our next step is to convert these ungrounded graphs to Freebase graphs.[2] Like Reddy et al. (2014), we treat this problem as a graph matching problem. Using GRAPHPARSER we retrieve all the Freebase graphs that are isomorphic to the ungrounded graph, and select only the graphs that could correctly predict the blank slot, as candidate graphs. Using these candidate graphs, we train a structured perceptron that learns to rank grounded graphs for a given ungrounded graph.[3] We use ungrounded predicate and Freebase predicate alignments as our features.

5 Experiments

5.1 Training and Evaluation Datasets

Our dataset SPADES (**S**emantic **PA**rsing of **DE**clarative **S**entences) is constructed from the declarative sentences collected by Reddy et al. (2014) from CLUEWEB09 (Gabrilovich et al., 2013) based on the following constraints: 1) There exists at least one isomorphic Freebase graph to the ungrounded representation of the input sentence; 2) There are no variable nodes in the ungrounded graph (e.g., *Google acquired a company* is discarded whereas *Google acquired the company Nest* is selected). We split this data into training (85%), development (5%) and testing (10%) sentences (Table 1). We introduce empty slots into these sentences by randomly removing an entity. SPADES can be downloaded at `http://github.com/sivareddyg/graph-parser`.

	Sentences	Tokens	Types	Entities
Train	79,247	685,922	69,095	37,606
Dev	4,763	41,102	9,306	4,358
Test	9,309	80,437	15,180	7,431

Table 1: SPADES Corpus Statistics

There has been other recent interest in similar datasets for sentence completion (Zweig et al., 2012) and machine reading (Hermann et al., 2015), but unlike other corpora our data is tied directly to Freebase and requires the execution of a semantic parse to correctly predict the missing entity. This is made more explicit by the fact that one third of the entities in our test set are never seen during training, so without a general approach to query creation and execution there is a limit on a system's performance.

5.2 Our Models

We use different CCG parsers varying in the amounts of supervision. For the UNSUPERVISED scenario, we use Bisk and Hockenmaier (2015)'s parser which

[2] Note that there is one-to-one correspondence between Freebase graphs and Freebase logical forms.

[3] Please see Section 4.3 of Reddy et al. (2016) for details.

		CCGbank (Syntax)		Slot Filling (Semantics)			
		LF1	UF1	2	3	4	Overall
	Sentences			∼6K	∼3K	∼600	∼10K
	Bag-of-Words	–	–	50.8	36.8	**20.9**	45.2
Syntax	UNSUPERVISED	37.1	64.2	41.6	30.4	**24.5**	37.3
Syntax	SEMI-SUPERVISED-POS	53.0	68.5	45.9	33.7	**29.1**	41.4
Syntax	SEMI-SUPERVISED-WORD	53.5	68.9	46.8	38.2	**28.3**	43.2
Syntax	SUPERVISED	84.2	91.0	49.3	42.0	**30.9**	46.1

Table 2: Syntactic and semantic evaluation of the parsing models. Left: Simplified labeled F1 and undirected unlabeled F1 on CCGbank, Section 23. Right: Slot filling performance (by number of entities per sentence).

exploits a small set of universal rules to automatically induce and weight a large set of lexical categories. For the semi-supervised, we explore two options – SEMI-SUPERVISED-WORD and SEMI-SUPERVISED-POS. We use Bisk et al. in both settings but we constrain its lexicon manually rather than inducing it from scratch. In the former, we restrict the top 200 words in English to occur only with the CCG categories that comprise 95% of the occurrences of a word's use in Section 22 of WSJ/CCGbank. In the latter, we restrict the POS tags instead of words. For the SUPERVISED scenario, we use EasyCCG (Lewis and Steedman, 2014) trained on CCGbank.

Finally, in order to further demonstrate the amount of useful information being learned by our parsers, we present a competitive Bag-of-Words baseline, which is a perceptron classifier that performs "semantic parsing" by predicting either a Freebase or a null relation between the empty slot and every other entity in the sentence, using the words in the sentence as features. This naive approach is competitive on simple sentences with only two entities, rivaling even the fully supervised parser, but falters as complexity increases.

5.3 Results and Discussion

Our primary focus is a comparison of intrinsic syntactic evaluation with our extrinsic semantic evaluation. To highlight the differences we present Section 23 parsing performance for our four models (Table 2). Dependency performance is evaluated on both the simplified labeled F1 of Bisk and Hockenmaier (2015) and Undirected Unlabeled F1.

Despite the supervised parser performing almost twice as well as the semi-supervised parsers on CCGbank LF1 (53 vs 84), in our semantic evaluation we see a comparatively small gain in performance (43 vs 46). It is interesting that such weakly supervised models are able to achieve over 90% of the performance of a fully supervised parser. To explore this further, we break down the semantics performance of all our models by the number of entities in a sentence. Each sentence has two, three, or four entities, one of which will be dropped for prediction. The more entities there are in a sentence, the more likely the models are to misanalyze a relation leading to their making the wrong prediction. These results are presented on the right side of Table 2. There are still notable discrepancies in performance, which we analyze more closely in the next section.

Another interesting result is the drop in performance by the Bag-of-Words Model. As the number of entities in the sentence increase, the model weakens, performing worse than the unsupervised parser on sentences with four entities. It becomes non-trivial for it to isolate which entities and relations should be used for prediction. This seems to indicate that the unsupervised grammar is capturing more useful syntactic/semantic information than what is available from the words alone. Ensemble systems that incorporate syntax and a Bag-of-Words baseline may yield even better performance.

5.4 The Benefits of Annotation

The performance of SEMI-SUPERVISED-POS and SEMI-SUPERVISED-WORD suggests that when resources are scarce, it is beneficial to create a even a small lexicon of CCG categories. We analyze this further in Figure 3. Here we show how performance changes as a function of the number of labeled lexical types. Our values range from 0 to 1000 lexical types. We see syntactic improvements of 16pts and seman-

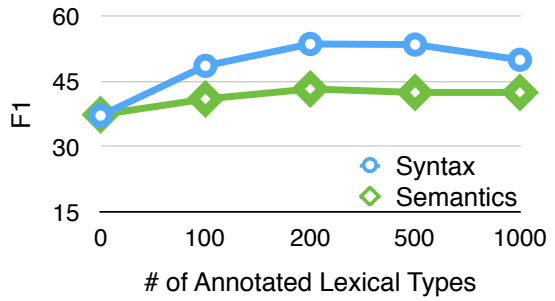


Figure 3: When our word based lexicon grows past 200 lexical types the semantic performance plateaus and the syntax begins to degrade. This is presumably due to the use of rare categories coupled with domain differences.

	Error	Example
Prevalent	Incorrect conjunction	*Stockholm, Sweden*
	Appositive	*, a chemist ,*
	Introductory clauses	*In Frankfurt, ...*
	Reduced relatives	*... , established in 1909, ...*
B&H 15	Verb chains	*is also headquartered*
	Possessive	*Anderson 's Foundation*
	PP Attachment	*of the foundation in Vancouver*

Table 3: Causes of semantic grounding errors with examples not previously isolated via intrinsic evaluation.

tic gains of 6pts with 200 words, before performance degrades. It is possible that increasing annotation may only benefit fully supervised models. Finally, when computing the most frequent lexical types we excluded commas. We found a 3pt performance drop when restricting commas to the category , (they are commonly conj in our data). Additional in-domain knowledge might further improve performance.

5.5 Common Errors

Bisk and Hockenmaier (2015) performed an in-depth analysis of the types of categories learned and correctly used by their models (the same models as this paper). Their analysis was based on syntactic evaluation against CCGbank. In particular, they found the most egregious "semantic" errors to be the misuse of verb chains, possessives and PP attachment (bottom of Table 3). Since we now have access to a purely semantic evaluation, we can therefore ask whether these errors exist here, and how common they are. We do this analysis in two steps. First, we manually analyzed parses for which the unsupervised model failed to predict the correct semantics, but where the supervised parser succeeded. The top of Table 3 presents several of the most common reasons for failure. These mistakes were more mundane (e.g. incorrect use of a conjunction) than failures to use complex CCG categories or analyze attachments.

Second, we can compare grammatical decisions made by the semi-supervised and unsupervised parsers against EasyCCG on sentences they successfully grounded. Bisk and Hockenmaier (2015) found that their unsupervised parser made mistakes on many very simple categories. We found the same result. When evaluating our parsers against the treebank we found the unsupervised model only correctly predicted transitive verbs 20% of the time and adverbs 39% of the time. In contrast, on our data, we produced the correct transitive category (according to EasyCCG) 65% of the time, and the correct adverb 68% of the time. These correct parsing decisions also lead to improved performance across many other categories (e.g. prepositions). This is likely due to our corpus containing simpler constructions. In contrast, auxiliary verbs, relative clauses, and commas still proved difficult or harder than in the treebank. This implies that future work should tailor the annotation effort to their specific domain rather than relying on guidance solely from the treebank.

6 Conclusion

Our goal in this paper was to present the first semantic evaluation of induced grammars in order to better understand their utility and strengths. We showed that induced grammars are learning more semantically useful structure than a Bag-of-Words model. Furthermore, we showed how minimal syntactic supervision can provide substantial gains in semantic evaluation. Our ongoing work explores creating a syntax-semantics loop where each benefits the other with no human (annotation) in the loop.

Acknowledgments

This paper is partly based on work that was done when the first and second authors were interns at Google, and on work that that was supported by NSF grant 1053856 to JH, and a Google PhD Fellowship to SR.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP Model for Inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, pages 75–88.

Yonatan Bisk and Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing,China, July.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1240.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA, June.

Glenn Carroll and Eugene Charniak. 1992. Two Experiments on Learning Probabilistic Dependency Grammars from Corpora. *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–15, March.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June.

Dan Garrette, Chris Dyer, Jason Baldridge, and Noah A Smith. 2015. Weakly-Supervised Grammar-Informed Bayesian CCG Parser Learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Dan Klein and Christopher D Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar, October.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, pages 1–16, June.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, September.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 601–610, Jeju Island, Korea, July.

Vector-space models for PPDB paraphrase ranking in context

Marianna Apidianaki
LIMSI, CNRS, Université Paris-Saclay
91403 Orsay, France
marianna.apidianaki@limsi.fr

Abstract

The PPDB is an automatically built database which contains millions of paraphrases in different languages. Paraphrases in this resource are associated with features that serve to their ranking and reflect paraphrase quality. This context-unaware ranking captures the semantic similarity of paraphrases but cannot serve to estimate their adequacy in specific contexts. We propose to use vector-space semantic models for selecting PPDB paraphrases that preserve the meaning of specific text fragments. This is the first work that addresses the substitutability of PPDB paraphrases in context. We show that vector-space models of meaning can be successfully applied to this task and increase the benefit brought by the use of the PPDB resource in applications.

1 Introduction

Paraphrases are alternative ways to convey the same information and can improve natural language processing by making systems more robust to language variability and unseen words. The paraphrase database (PPDB) (Ganitkevitch et al., 2013) contains millions of automatically acquired paraphrases in 21 languages associated with features that serve to their ranking. In PPDB's most recent release (2.0), such features include natural logic entailment relations, distributional and word embedding similarities, formality and complexity scores, and scores assigned by a supervised ranking model (Pavlick et al., 2015b). These features serve to identify good quality paraphrases but do not say much about their substitutability in context.

To judge the adequacy of paraphrases for specific instances of words or phrases, the surrounding context needs to be considered. This can be done using vector-space models of semantics which calculate the meaning of word occurrences in context based on distributional representations (Mitchell and Lapata, 2008; Erk and Padó, 2008; Dinu and Lapata, 2010; Thater et al., 2011). These models capture the influence of the context on the meaning of a target word through vector composition. More precisely, they represent the contextualised meaning of a target word w in context c by a vector obtained by combining the vectors of w and c using some operation such as component-wise multiplication or addition (Thater et al., 2011). We use this kind of representations to rank the PPDB paraphrases in context and retain the ones that preserve the semantics of specific text fragments. We evaluate the vector-based ranking models on data hand-annotated with lexical variants and compare the obtained ranking to confidence estimates available in the PPDB, highlighting the importance of context filtering for paraphrase selection.

2 Context-based paraphrase ranking

2.1 Paraphrase substitutability

The PPDB[1] provides millions of lexical, phrasal and syntactic paraphrases in 21 languages – acquired by applying bi- and multi-lingual pivoting on parallel corpora (Bannard and Callison-Burch, 2005) – and is largely exploited in applications (Denkowski and Lavie, 2010; Sultan et al., 2014; Faruqui et al.,

[1] http://paraphrase.org/#/download

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2028–2034,
Austin, Texas, November 1-5, 2016.

2015). PPDB paraphrases come into packages of different sizes (going from S to XXXL): smaller packages contain high-precision paraphrases while larger ones aim for high coverage. Until now, pivot paraphrases have been used as equivalence sets (i.e. all paraphrases available for a word are viewed as semantically equivalent) and their substitutability in context has not yet been addressed.

Substitutability might be restrained by several factors which make choosing the appropriate paraphrase for a word or phrase in different contexts a non-trivial task. In case of polysemous words, paraphrases describe different meanings and can lead to erroneous semantic mappings if substituted in texts (Apidianaki et al., 2014; Cocos and Callison-Burch, 2016). Even when paraphrases capture the same general sense, they are hardly ever equivalent synonyms and generally display subtle differences in meaning, connotation or usage (Edmonds and Hirst, 2002). Stylistic variation might also be present within paraphrase sets and substituting paraphrases that differ in terms of complexity and formality can result in a change in style (Pavlick and Nenkova, 2015). To increase paraphrase applicability in context, Pavlick et al. (2015a) propose to extract domain-specific pivot paraphrases by biasing the parallel training data used by the pivot method towards a specific domain. This customised model greatly improves paraphrase quality for the target domain but does not allow to rank and filter the paraphrases already in the PPDB according to specific contexts. To our knowledge, this is the first work that addresses the question of in-context substitutability of PPDB paraphrases. We show how existing substitutability models can be applied to this task in order to increase the usefulness of this large-scale resource in applications.

2.2 Vector-space models of paraphrase adequacy

Vector-based models of meaning determine a gradual concept of semantic similarity which does not rely on a fixed set of dictionary senses. They are used for word sense discrimination and induction (Schütze, 1998; Turney and Pantel, 2010) and can capture the contextualised meaning of words and phrases (Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2011). Vector composition methods build representations that go beyond individual words to obtain word meanings in context. Some models use explicit sense representations while others modify the basic meaning vector of a target word with information from the vectors of the words in its context. In the framework proposed by Dinu and Lapata (2010), for example, word meaning is represented as a probability distribution over a set of latent senses reflecting the out-of-context likelihood of each sense, and the contextualised meaning of a word is modeled as a change in the original sense distribution.[2] Reisinger and Mooney (2010) propose a multi-prototype vector-space model of meaning which produces multiple "sense-specific" vectors for each word, determined by clustering the contexts in which the word appears (Schütze, 1998). The cluster centroids serve as prototype vectors describing a word's senses and the meaning of a specific occurrence is determined by choosing the vector that minimizes the distance to the vector representing the current context. On the contrary, Thater et al. (2011) use no explicit sense representation. Their models allow the computation of vector representations for individual uses of words, characterising the specific meaning of a target word in its sentential context. When used for paraphrase ranking, these models derive a contextualised vector for a target word by reweighting the components of its basic meaning vector on the basis of the context of occurrence.[3] Paraphrase candidates for a target word are then ranked according to the cosine similarity of their basic vector representation to the contextualised vector of the target.[4]

3 Experimental Set-up

Data In our experiments, we use the CoInCo corpus (Kremer et al., 2014), a subset of the "Manually Annotated Sub-Corpus" MASC (Ide et al., 2010) which comprises more than 15K word in-

[2]The latent senses are induced using non-negative matrix factorization (NMF) (Lee and Seung, 2001) and latent Dirichlet allocation (LDA) (Blei et al., 2003).

[3]Depending on the model, the vector combination function might be addition or multiplication of vector elements.

[4]Thater et al.'s (2011) models delivered best results in paraphrase ranking on the CoInCo corpus (Kremer et al., 2014) and the SemEval-2007 Lexical Substitution dataset (McCarthy and Navigli, 2007).

	$\|P\| > 1$			$\|P\| \geq 1$
PPDB	# Instances	# Lemmas	Avg $\|P\|$	# Instances
S	2146	560	2.67	5573
M	3716	855	2.92	7771
L	6228	1394	3.57	10100
XL	13344	2822	10.33	14060
XXL	14507	3308	185.09	14593

Table 1: Number of CoInCo instances and distinct lemmas covered by each PPDB package.

stances manually annotated with single and multi-word substitutes. The manual annotations serve to evaluate the performance of the vector-space models on the task of ranking PPDB paraphrases. For each annotated English target word (noun, verb, adjective or adverb) in CoInCo, we collect the lexical paraphrases ($P = \{p_1, p_2, ..., p_n\}$) available for the word in each PPDB package (from S to XXL).[5] We do not filter by syntactic label as annotations often include substitutes of different grammatical categories. Table 1 shows the number of CoInCo tokens with paraphrases in each PPDB package and the average size of the retained paraphrase sets. The larger the size of the resource, the greater the coverage of target words in CoInCo. The last column of the table gives the total number of instances covered, including the ones with only one paraphrase. In the ranking experiments, we focus on lemmas having more than one paraphrase in the PPDB.[6]

Methodology We follow the methodology proposed in Kremer et al. (2014) to explore the extent to which vector-based models can select appropriate paraphrases for words in context. Given a target word w in a sentential context and a set of paraphrases P extracted for w from a PPDB package, the task is to rank the elements in P according to their adequacy as paraphrases of w in the given context.

We carry out experiments with three versions of the Thater et al. (2011) ranking model: (a) a *syntactically structured* model (Syn.Vec) that uses vectors recording co-occurrences based on dependency triples, explicitly recording syntactic role information within the vectors; (b) a *syntactically filtered model* (Filter.Vec) using dependency-based co-occurrence information without explicitly representing the syntactic role in the vector representations, as in Padó and Lapata (2007); (c) a bag of words model (Bow.Vec) using a window of $\pm$ 5 words. Co-occurrence counts were extracted from the English Gigaword corpus[7] analysed with Stanford dependencies (de Marneffe et al., 2006). The syntactic model vectors are based on dependency triples that occur at least 5 times in the corpus and have a PMI score of at least 2. The same thresholds apply to the bag of words model where the frequency threshold defines the minimum number of times that two words have been observed in the same context window. The task of the vector-space models for each target word instance is to rank the contents of the corresponding paraphrase set (which contains all the substitution candidates available for the target in the PPDB) so that the actual substitutes are ranked higher than the rest. For example, *newspaper, manuscript* and *document* are good paraphrase candidates for *paper* but we would expect *newspaper* to be ranked higher than the other two in this sentence: "*the paper's local administrator*".

A contextualised vector is derived from the basic meaning vector of a target word w by reinforcing its dimensions that are licensed by the context of the specific instance under consideration. In the Bow.Vec model, the context is made up of 5 words before and after the target while in the syntactic models, it corresponds to the target's direct syntactic dependents. The contextualised vector for w is obtained through vector addition and contains information about the context words. Paraphrase candidates are ranked according to the cosine similarity between the contextualised vector of the target word and the basic meaning vectors of the candidates. Following Kremer et al. (2014), we compare the resulting ranked list to the CoInCo gold standard annotation (the paraphrase set of the target instance) using Generalised Average Precision (GAP) (Kishida, 2005) and annotation frequency as weights. GAP scores range between 0 and 1: a score of 1 indicates a perfect ranking in which all correct substitutes precede all incorrect ones, and

[5] Since the XXL package covers almost all annotated instances in CoInCo (14,507 out of 15,629) and there are 185.09 paraphrases in average for each instance, we exclude the XXXL package from these experiments.

[6] We retain paraphrases of the lemmatised forms of the target words but these unsupervised ranking models can be easily applied to the whole PPDB resource and in different languages.

[7] http://catalog.ldc.upenn.edu/LDC2003T05

	PPDB	Bow.Vec	Syn.Vec	Filter.Vec	Google	AGiga	Ppdb1	Ppdb2	Parprob	Random (5)
$\|P\| > 1$	S	**0.91**	**0.91**	**0.91**	0.78	0.86	0.66	0.83	0.66	0.78
	M	0.91	0.91	**0.92**	0.79	0.87	0.68	0.84	0.68	0.79
	L	0.90	0.90	**0.91**	0.78	0.85	0.66	0.83	0.66	0.77
	XL	0.78	**0.79**	**0.79**	0.58	0.67	0.44	0.66	0.43	0.58
	XXL	0.53	0.56	0.57	0.27	0.36	0.12	**0.58**	0.12	0.27
$\|P\| \geq 1$	S	**0.97**	**0.97**	**0.97**	0.91	0.95	0.87	0.93	0.87	0.91
	M	**0.96**	**0.96**	**0.96**	0.90	0.94	0.85	0.92	0.85	0.90
	L	**0.94**	**0.94**	**0.94**	0.87	0.91	0.79	0.90	0.79	0.86
	XL	0.79	**0.80**	**0.80**	0.60	0.69	0.47	0.68	0.46	0.60
	XXL	0.54	0.56	0.58	0.28	0.37	0.13	**0.59**	0.14	0.28

Table 2: Average GAP scores for the contextual models, five paraphrase adequacy methods and the random ranking baseline against the gold CoInCo annotations. Scores reported for different sizes of the PPDB (from S to XXL).

correct high-weight substitutes precede low-weight ones. For calculating the GAP score, we assign a very low score (0.001) to paraphrases that are not present in CoInCo for a target word (i.e. not proposed by the annotators).

4 Results

The average GAP scores obtained by the three vector-space models (Bow.Vec, Syn.Vec and Filter.Vec) are shown in Table 2. The upper part of the table reports scores obtained for words with more than one paraphrase in the PPDB ($|P| > 1$) while the lower part gives the scores for all words.

We compare the GAP scores to five different rankings reflecting paraphrase quality in the PPDB (Pavlick et al., 2015b). We retain the following scores: 1. **AGigaSim** captures the distributional similarity of a phrase e_1 and its paraphrase e_2 computed according to contexts observed in the Annotated Gigaword corpus (Napoles et al., 2011); 2. **GoogleNgramSim** reflects the distributional similarity of e_1 and e_2 computed according to contexts observed in the Google Ngram corpus (Brants and Franz, 2006); 3. **ParProb**: the paraphrase probability of e_2 given the original phrase e_1 (Bannard and Callison-Burch, 2005); 4. **Ppdb1**: the heuristic scoring used for ranking in the original release of the PPDB (Ganitkevitch et al., 2013); 5. **Ppdb2**: the improved ranking of English paraphrases available in PPDB 2.0. The results are also compared to the output of a baseline where the paraphrases are randomly ranked. The reported baseline figures are PPDB package-specific since a different paraphrase set is retained from each package, and correspond to averages over 5 runs. The quality of the ranking produced by the baseline clearly decreases as the size of the PPDB resource increases due to the higher number of retained paraphrases which makes ranking harder.

The results in the upper part of the table show that the vector-space models provide a better ranking than the PPDB estimates and largely outperform the random baseline. The three models perform similarly on this ranking task according to average GAP with the syntactically-informed models getting slightly higher scores. Differences between Syn.Vec and Filter.Vec, as well as between Bow.Vec and the syntactic models, are highly significant in the XL and XXL packages (p-value < 0.001) as computed with approximate randomisation (Padó, 2006). In the L package, the difference between Syn.Vec and Filter.Vec is significant ($p < 0.05$) and the one between Bow.Vec and Filter.Vec is highly significant. Finally, in the M package, only the difference between Bow.Vec and Filter.Vec is significant ($p < 0.05$), while Syn.Vec and Filter.Vec seem to deal similarly well with the contents of this package.

Two PPDB ranking methods, AGiga and Ppdb2, obtain good results. AgigaSim reflects the distributional similarity of the paraphrases in the Annotated Gigaword corpus (Napoles et al., 2011). As noted by Kremer et al. (2014), the whole-document annotation in CoInCo faces the natural skewed distribution towards predominant senses which favors non-contextualised baseline models. The good performance of Ppdb2 is due to the use of a supervised scoring model trained on human judgments of paraphrase quality. The human judgments were

used to fit a regression to the features available in PPDB 1.0 plus numerous new features including cosine word embedding similarity, lexical overlap features, WordNet features and distributional similarity features.[8] The small difference observed between the Ppdb2 and the syntactic models score in the XXL package is highly significant. For the moment, Ppdb2 scores are available in the PPDB only for English. Since the vector-space methodology is unsupervised and language independent, it could be easily applied to paraphrase ranking in other languages. The performance of the models remains high with the XL package which contains paraphrase sets of reasonable size (about 10 paraphrases per word) and ensures a high coverage, and lowers in XXL which contains 185 paraphrases in average per word (cf. Table 1). To use this package more efficiently, one could initially reduce the number of erroneous paraphrases on the basis of the Ppdb2 score which provides a good ranking of the XXL package contents before applying the vector-based models.

The increase in GAP score observed when words with one paraphrase are considered shows that these paraphrases are often correct. Here too, the contextual models provide a better ranking than the out-of-context scores and outperform the random baseline. As in the previous case, the Ppdb2 score is slightly higher in the XXL package.

5 Conclusion

We have shown that vector-based models of semantics can be successfully applied to in-context ranking of PPDB paraphrases. Allowing for better context-informed substitutions, they can be used to filter PPDB paraphrases on the fly and select variants preserving the correct semantics of words and phrases in texts. This processing would be beneficial to numerous applications that need paraphrase support (e.g. summarisation, query reformulation and language learning), providing a practical means for exploiting the extensive multilingual knowledge available in the PPDB resource.

This study opens up many avenues for future work. Although tested on English, the proposed methodology can be applied to all languages in the PPDB even to the ones that do not dispose of a dependency parser (as shown by the high performance of the Bow.Vec models).

An ideal testbed for evaluation in a real application and on multiple languages is offered by MT evaluation. The METEOR-NEXT metric (Denkowski and Lavie, 2010) provides a straightforward framework for testing as it already exploits PPDB paraphrases for capturing sense correspondences between text fragments. In its current version, the metric views paraphrases as equivalent classes which can lead to erroneous sense mappings due to semantic distinctions present in the paraphrase sets. We have recently showed that the context-based filtering of semantic variants improves METEOR's correlation with human judgments of translation quality (Marie and Apidianaki, 2015). We believe that a context-based paraphrase ranking mechanism will enhance correct substitutions and further improve the metric. Last but not least, the paraphrase vectors can be used for mapping the contents of the PPDB resource to other multilingual resources for which vector representations are available (Camacho-Collados et al., 2015a; Camacho-Collados et al., 2015b). The interest of mapping paraphrases in the vector space to concepts found in existing semantic resources is twofold: it would permit to analyse the semantics of the paraphrases by putting them into correspondence with explicit concept representations and would serve to enrich other semantic resources (e.g. BabelNet synsets) with semantically similar paraphrases.

Handling phrasal paraphrases is another natural extension of this work. We consider using a vector space model of semantic composition to calculate the meaning of longer candidate paraphrases (Dinu et al., 2013; Paperno et al., 2014) and select appropriate substitutes for phrases in context.

Acknowledgments

We would like to thank Stefan Thater for sharing the vector-space models, Benjamin Marie for his support with the paraphrase ranking models and the anonymous reviewers for their valuable comments and suggestions.

[8] The features used for computing the paraphrase ranking in PPDB 2.0 are described in detail in Pavlick et al. (2015b).

References

Marianna Apidianaki, Emilia Verzeni, and Diana McCarthy. 2014. Semantic Clustering of Pivot Paraphrases. In *Proceedings of LREC*, Reykjavik, Iceland.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597–604, Ann Arbor, Michigan, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Thorsten Brants and Alex Franz. 2006. *The Google Web 1T 5-gram Corpus Version 1.1*. LDC2006T13, Philadelphia.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015a. Nasari: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado, May–June.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015b. A unified multilingual semantic representation of concepts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 741–751, Beijing, China, July.

Anne Cocos and Chris Callison-Burch. 2016. Clustering paraphrases by word sense. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*, San Diego, California, USA.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *To appear at LREC-06*.

Michael Denkowski and Alon Lavie. 2010. METEOR-NEXT and the METEOR Paraphrase Tables: Improved Evaluation Support for Five Target Languages. In *Proceedings of WMT/MetricsMATR*, pages 339–342, Uppsala, Sweden.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA, October.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. Dissect - distributional semantics composition toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, Sofia, Bulgaria, August.

Philip Edmonds and Graeme Hirst. 2002. Near-Synonymy and Lexical Choice. *Computational Linguistics*, 28(2):105–144.

Katrin Erk and Sebastian Padó. 2008. A Structured Vector Space Model for Word Meaning in Context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, Hawaii.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado.

Juri Ganitkevitch, Benjamin VanDurme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL*, Atlanta, Georgia, USA.

Nancy Ide, Collin Baker, Christiane Fellbaum, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 68–73, Uppsala, Sweden.

Kazuaki Kishida. 2005. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. Technical report, Technical Report NII-2005-014E.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus. In *Proceedings of EACL*, pages 540–549, Gothenburg, Sweden.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 556–562. MIT Press.

Benjamin Marie and Marianna Apidianaki. 2015. Alignment-based sense selection in METEOR and the RATATOUILLE recipe. In *Proceedings of WMT*, pages 385–391, Lisbon, Portugal.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL/HLT*, pages 236–244, Columbus, Ohio, USA.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90, Portland, Oregon.

Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 90–99, Baltimore, Maryland, June.

Ellie Pavlick and Ani Nenkova. 2015. Inducing lexical style properties for paraphrase and genre differentiation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 218–224, Denver, Colorado.

Ellie Pavlick, Juri Ganitkevitch, Tsz Ping Chan, Xuchen Yao, Benjamin Van Durme, and Chris Callison-Burch. 2015a. Domain-Specific Paraphrase Extraction. In *Proceedings of ACL/IJCNLP*, pages 57–62, Beijing, China.

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

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, Los Angeles, California.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:97–123.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Dls@cu: Sentence similarity from word alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 241–246, Dublin, Ireland, August.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word Meaning in Context: A Simple and Effective Vector Model. In *Proceedings of IJCNLP*, pages 1134–1143, Chiang Mai, Thailand.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Interpreting Neural Networks to Improve Politeness Comprehension

Malika Aubakirova
University of Chicago
aubakirova@uchicago.edu

Mohit Bansal
UNC Chapel Hill
mbansal@cs.unc.edu

Abstract

We present an interpretable neural network approach to predicting and understanding politeness in natural language requests. Our models are based on simple convolutional neural networks directly on raw text, avoiding any manual identification of complex sentiment or syntactic features, while performing better than such feature-based models from previous work. More importantly, we use the challenging task of politeness prediction as a testbed to next present a much-needed understanding of what these successful networks are actually learning. For this, we present several network visualizations based on activation clusters, first derivative saliency, and embedding space transformations, helping us automatically identify several subtle linguistics markers of politeness theories. Further, this analysis reveals multiple novel, high-scoring politeness strategies which, when added back as new features, reduce the accuracy gap between the original featurized system and the neural model, thus providing a clear quantitative interpretation of the success of these neural networks.

1 Introduction

Politeness theories (Brown and Levinson, 1987; Gu, 1990; Bargiela-Chiappini, 2003) include key components such as modality, indirection, deference, and impersonalization. Positive politeness strategies focus on making the hearer feel good through offers, promises, and jokes. Negative politeness examples include favor seeking, orders, and requests. Differentiating among politeness types is a highly nontrivial task, because it depends on factors such as a context, relative power, and culture. Danescu-Niculescu-Mizil et al. (2013) proposed a useful computational framework for predicting politeness in natural language requests by designing various lexical and syntactic features about key politeness theories, e.g., first or second person start vs. plural. However, manually identifying such politeness features is very challenging, because there exist several complex theories and politeness in natural language is often realized via subtle markers and non-literal cues.

Neural networks have been achieving high performance in sentiment analysis tasks, via their ability to automatically learn short and long range spatial relations. However, it is hard to interpret and explain what they have learned. In this paper, we first propose to address politeness prediction via simple CNNs working directly on the raw text. This helps us avoid the need for any complex, manually-defined linguistic features, while still performing better than such featurized systems. More importantly, we next present an intuitive interpretation of what these successful neural networks are learning, using the challenging politeness task as a testbed.

To this end, we present several visualization strategies: activation clustering, first derivative saliency, and embedding space transformations, some of which are inspired by similar strategies in computer vision (Erhan et al., 2009; Simonyan et al., 2014; Girshick et al., 2014), and have also been recently adopted in NLP for recurrent neural networks (Li et al., 2016; Kádár et al., 2016). The neuron activation clustering method not only rediscovers and extends several manually defined features from politeness theories, but also uncovers multiple novel strategies, whose importance we measure quantitatively. The first derivative saliency technique allows us to identify the impact of each phrase

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2035–2041,
Austin, Texas, November 1-5, 2016.

on the final politeness prediction score via heatmaps, revealing useful politeness markers and cues. Finally, we also plot lexical embeddings before and after training, showing how specific politeness markers move and cluster based on their polarity. Such visualization strategies should also be useful for understanding similar state-of-the-art neural network models on various other NLP tasks.

Importantly, our activation clusters reveal two novel politeness strategies, namely indefinite pronouns and punctuation. Both strategies display high politeness and top-quartile scores (as defined by Danescu-Niculescu-Mizil et al. (2013)). Also, when added back as new features to the original featurized system, they improve its performance and reduce the accuracy gap between the featurized system and the neural model, thus providing a clear, quantitative interpretation of the success of these neural networks in automatically learning useful features.

2 Related Work

Danescu-Niculescu-Mizil et al. (2013) presented one of the first useful datasets and computational approaches to politeness theories (Brown and Levinson, 1987; Goldsmith, 2007; Kádár and Haugh, 2013; Locher and Watts, 2005), using manually defined lexical and syntactic features. Substantial previous work has employed machine learning models for other sentiment analysis style tasks (Pang et al., 2002; Pang and Lee, 2004; Kennedy and Inkpen, 2006; Go et al., 2009; Ghiassi et al., 2013). Recent work has also applied neural network based models to sentiment analysis tasks (Chen et al., 2011; Socher et al., 2013; Moraes et al., 2013; Dong et al., 2014; dos Santos and Gatti, 2014; Kalchbrenner et al., 2014). However, none of the above methods focused on visualizing and understanding the inner workings of these successful neural networks.

There have been a number of visualization techniques explored for neural networks in computer vision (Krizhevsky et al., 2012; Simonyan et al., 2014; Zeiler and Fergus, 2014; Samek et al., 2016; Mahendran and Vedaldi, 2015). Recently in NLP, Li et al. (2016) successfully adopt computer vision techniques, namely first-order *saliency*, and present representation plotting for sentiment compositionality across RNN variants. Similarly, Kádár et al. (2016) analyze the omission scores and top-k contexts of hidden units of a multimodal RNN. Karpathy et al. (2016) visualize character-level language models. We instead adopt visualization techniques for CNN style models for NLP[1] and apply these to the challenging task of politeness prediction, which often involves identifying subtle and non-literal sociolinguistic cues. We also present a quantitative interpretation of the success of these CNNs on the politeness prediction task, based on closing the performance gap between the featurized and neural models.

3 Approach

3.1 Convolutional Neural Networks

We use one convolutional layer followed by a pooling layer. For a sentence $v_{1:n}$ (where each word v_i is a d-dim vector), a filter m applied on a window of t words, produces a convolution feature $c_i = f(m * v_{i:i+t-1} + b)$, where f is a non-linear function, and b is a bias term. A *feature map* $c \in R^{n-t+1}$ is applied on each possible window of words so that $c = [c_1, ..., c_{n-t+1}]$. This convolutional layer is then followed by a max-over-pooling operation (Collobert et al., 2011) that gives $C = max\{c\}$ of the particular filter. To obtain multiple features, we use multiple filters of varying window sizes. The result is then passed to a fully-connected softmax layer that outputs probabilities over labels.

4 Experimental Setup

4.1 Datasets

We used the two datasets released by Danescu-Niculescu-Mizil et al. (2013): Wikipedia (Wiki) and Stack Exchange (SE), containing community requests with politeness labels. Their 'feature development' was done on the Wiki dataset, and SE was used as the 'feature transfer' domain. We use a simpler train-validation-test split based setup for these datasets instead of the original leave-one-out cross-validation setup, which makes training extremely slow for any neural network or sizable classifier.[2]

[1] The same techniques can also be applied to RNN models.

[2] The result trends and visualizations using cross-validation were similar to our current results, in preliminary experiments. We will release our exact dataset split details.

4.2 Training Details

Our tuned hyperparameters values (on the dev set of Wiki) are a mini-batch size of 32, a learning rate of 0.001 for the Adam (Kingma and Ba, 2015) optimizer, a dropout rate of 0.5, CNN filter windows of 3, 4, and 5 with 75 feature maps each, and ReLU as the non-linear function (Nair and Hinton, 2010). For convolution layers, we use valid padding and strides of all ones. We followed Danescu-Niculescu-Mizil et al. (2013) in using SE only as a transfer domain, i.e., we do not re-tune any hyperparameters or features on this domain and simply use the chosen values from the Wiki setting. The split and other training details are provided in the supplement.

5 Results

Table 1 first presents our reproduced classification accuracy test results (two labels: positive or negative politeness) for the bag-of-words and linguistic features based models of Danescu-Niculescu-Mizil et al. (2013) (for our dataset splits) as well as the performance of our CNN model. As seen, without using any manually defined, theory-inspired linguistic features, the simple CNN model performs better than the feature-based methods.[3]

Next, we also show how the linguistic features baseline improves on adding our novelly discovered features (plus correcting some exising features), revealed via the analysis in Sec. 6. Thus, this reduces the gap in performance between the linguistic features baseline and the CNN, and in turn provides a quantitative reasoning for the success of the CNN model. More details in Sec. 6.

6 Analysis and Visualization

We present the primary interest and contribution of this work: performing an important qualitative and quantitative analysis of what is being learned by our neural networks w.r.t. politeness strategies.[4]

6.1 Activation Clusters

Activation clustering is a non-parametric approach (adopted from Girshick et al. (2014)) of computing each CNN unit's activations on a dataset and then analyzing the top-scoring samples in each cluster. We keep track of which neurons get maximally activated for which Wikipedia requests and analyze the most frequent requests in each neuron's cluster, to understand what each neuron reacts to.

Model	Wiki	SE
Bag-of-Words	80.9%	64.6%
Linguistic Features	82.6%	65.2%
With Discovered Features	83.8%	65.7%
CNN	85.8%	66.4%

Table 1: Accuracy Results on Wikipedia and Stack Exchange.

6.1.1 Rediscovering Existing Strategies

We find that the different activation clusters of our neural network automatically rediscover a number of strategies from politeness theories considered in Danescu-Niculescu-Mizil et al. (2013) (see Table 3 in their paper). We present a few such strategies here with their supporting examples, and the rest (e.g., Gratitude, Greeting, Positive Lexicon, and Counterfactual Modal) are presented in the supplement. The majority politeness label of each category is indicated by (+) and (-).

Deference (+) A way of sharing the burden of a request placed on the addressee. Activation cluster examples: {"*nice work so far on your rewrite...*"; "*hey, good work on the new pages...*"}

Direct Question (-) Questions imposed on the converser in a direct manner with a demand of a factual answer. Activation cluster examples: {"*what's with the radio , and fist in the air?*"; "*what level warning is appropriate?*"}

6.1.2 Extending Existing Strategies

We also found that certain activation clusters depicted interesting extensions of the politeness strategies given in previous work.

Gratitude (+) Our CNN learns a special shade of gratitude, namely it distinguishes a cluster consisting of the bigram *thanks for*. Activation cluster examples: {"*thanks for the good advice.*"; "*thanks for letting me know.*"}

Counterfactual Modal (+) Sentences with *Would you/Could you* get grouped together as expected; but in addition, the cluster contains requests with *Do you mind* as well as gapped 3-grams like *Can you ... please?*, which presumably implies that the combi-

[3]For reference, human performance on the original task setup of Danescu-Niculescu-Mizil et al. (2013) was 86.72% and 80.89% on the Wiki and SE datasets, respectively.

[4]We only use the Wiki train/dev sets for all analysis.

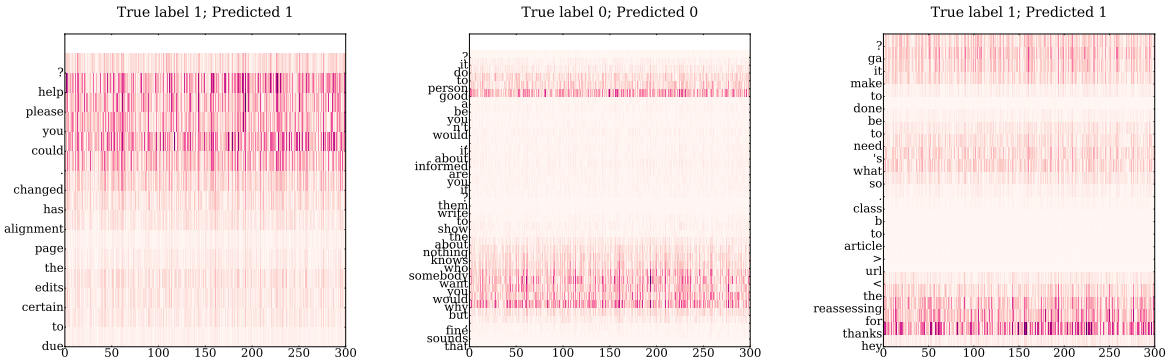


Figure 1: Saliency heatmaps for correctly classified sentences.

nation of a later *please* with future-oriented variants *can/will* in the request gives a similar effect as the conditional-oriented variants *would/could*. Activation cluster examples: {*can this be reported ... grid, please?*"; *do you mind having another look?*"}

6.1.3 Discovering Novel Strategies

In addition to rediscovering and extending politeness strategies mentioned in previous work, our network also automatically discovers some novel activation clusters, potentially corresponding to new politeness strategies.

Indefinite Pronouns (-) Danescu-Niculescu-Mizil et al. (2013) distinguishes requests with first and second person (plural, starting position, etc.). However, we find activations that also react to indefinite pronouns such as *something/somebody*. Activation cluster examples: {"*am i missing something here?*"; "*wait for anyone to discuss it.*"}

Punctuation (-) Though non-characteristic in direct speech, punctuation appears to be an important special marker in online communities, which in some sense captures verbal emotion in text. E.g., one of our neuron clusters gets activated on question marks "???" and one on ellipsis "...". Activation cluster examples: {"*now???*"; "*original article????*"; "*helllo?????*"}[5]

In the next section, via saliency heatmaps, we will further study the impact of indefinite pronouns in the final-decision making of the classifier. Finally, in Sec. 6.4, we will quantitatively show how our newly discovered strategies help directly improve the accuracy performance of the linguistic features baseline and achieve high politeness and top-quartile scores as per Danescu-Niculescu-Mizil et al. (2013).

[5]More examples are given in the supplement.

6.2 First Derivative Saliency

Inspired from neural network visualization in computer vision (Simonyan et al., 2014), the first derivative saliency method indicates how much each input unit contributes to the final decision of the classifier. If E is the input embedding, y is the true label, and $S_y(E)$ is the neural network output, then we consider gradients $\frac{\partial S_y(E)}{\partial e}$. Each image in Fig. 1 is a heatmap of the magnitudes of the derivative in absolute value with respect to each dimension.

The first heatmap gets signals from *please* (Please strategy) and *could you* (Counterfactual Modal strategy), but effectively puts much more mass on *help*. This is presumably due to the nature of Wikipedia requests such that the meaning boils down to asking for some help that reduces the social distance. In the second figure, the highest emphasis is put on *why would you*, conceivably used by Wikipedia administrators as an indicator of questioning. Also, the indefinite pronoun *somebody* makes a relatively high impact on the decision. This relates back to the activation clustering mentioned in the previous section, where indefinite pronouns had their own cluster. In the third heatmap, the neural network does not put much weight on the greeting-based start *hey*, because it instead focuses on the higher polarity[6] gratitude part after the greeting, i.e., on the words *thanks for*. This will be further connected in Sec. 6.3.

6.3 Embedding Space Transformations

We selected key words from Danescu-Niculescu-Mizil et al. (2013) and from our new activation clusters (Sec. 6.1) and plotted (via PCA) their embed-

[6]See Table 3 of Danescu-Niculescu-Mizil et al. (2013) for polarity scores of the various strategies.

	Strategy	Politeness	In top quartile	Examples
21.	Indefinite Pronouns	-0.13	39%	*am i missing something here?*
22.	Punctuation	-0.71	62%	*helllo?????*

Table 2: Extending Table 3 of Danescu-Niculescu-Mizil et al. (2013) with our novelly discovered politeness strategies.

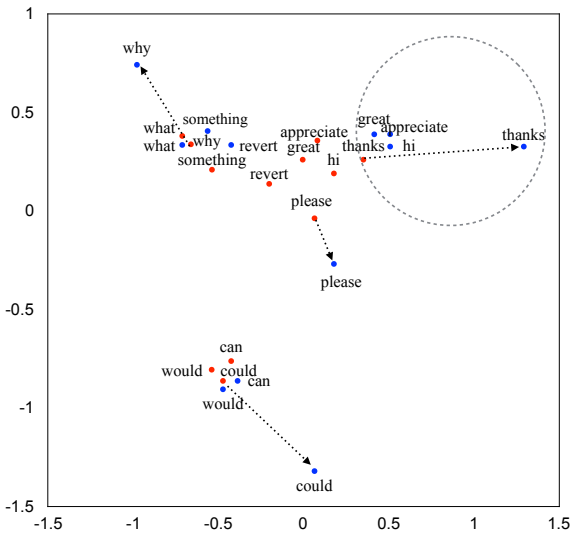


Figure 2: Projection before (red) and after (blue) training.

ding space positions before and after training, to help us gain insights into specific sentiment transformations. Fig. 2 shows that the most positive keys such as *hi*, *appreciate*, and *great* get clustered even more tightly after training. The key *thanks* gets a notably separated position on a positive spectrum, signifying its importance in the NN's decision-making (also depicted via the saliency heatmaps in Sec. 6.2).

The indefinite pronoun *something* is located near direct question politeness strategy keys *why* and *what*. *Please*, as was shown by Danescu-Niculescu-Mizil et al. (2013), is not always a positive word because its sentiment depends on its sentence position, and it moves further away from a positive key group. Counterfactual Modal keys *could* and *would* as well as *can* of indicative modal get far more separated from positive keys. Moreover, after the training, the distance between *could* and *would* increases but it gets preserved between *can* and *would*, which might suggest that *could* has a far stronger sentiment.

6.4 Quantitative Analysis

In this section, we present quantitative measures of the importance and polarity of the novelly discovered politeness strategies in the above sections, as well how they explain some of the improved performance of the neural model.

In Table 3 of Danescu-Niculescu-Mizil et al. (2013), the pronoun politeness strategy with the highest percentage in top quartile is 2nd Person (30%). Our extension Table 2 shows that our novelly discovered Indefinite Pronouns strategy represents a higher percentage (39%), with a politeness score of -0.13. Moreover, our Punctuation strategy also turns out to be a top scoring negative politeness strategy and in the top three among all strategies (after Gratitude and Deference). It has a score of -0.71, whereas the second top negative politeness strategy (Direct Start) has a much lower score of -0.43.

Finally, in terms of accuracies, our newly discovered features of Indefinite Pronouns and Punctuation improved the featurized system of Danescu-Niculescu-Mizil et al. (2013) (see Table 1).[7] This reduction of performance gap w.r.t. the CNN partially explains the success of these neural models in automatically learning useful linguistic features.

7 Conclusion

We presented an interpretable neural network approach to politeness prediction. Our simple CNN model improves over previous work with manually-defined features. More importantly, we then understand the reasons for these improvements via three visualization techniques and discover some novel high-scoring politeness strategies which, in turn, quantitatively explain part of the performance gap between the featurized and neural models.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by an IBM Faculty Award, a Bloomberg Research Grant, and an NVIDIA GPU donation to MB.

[7] Our NN visualizations also led to an interesting feature correction. In the 'With Discovered Features' result in Table 1, we also removed the existing pronoun features (#14-18) based on the observation that those had weaker activation and saliency contributions (and lower top-quartile %) than the new indefinite pronoun feature. This correction and adding the two new features contributed ∼50-50 to the total accuracy improvement.

References

Francesca Bargiela-Chiappini. 2003. Face and politeness: new (insights) for old (concepts). *Journal of pragmatics*, 35(10):1453–1469.

Penelope Brown and Stephen C Levinson. 1987. *Politeness: Some universals in language usage*, volume 4. Cambridge university press.

Long-Sheng Chen, Cheng-Hsiang Liu, and Hui-Ju Chiu. 2011. A neural network based approach for sentiment classification in the blogosphere. *Journal of Informetrics*, 5(2):313–322.

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

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of ACL*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 49–54.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*, pages 69–78.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341.

M Ghiassi, J Skinner, and D Zimbra. 2013. Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network. *Expert Systems with applications*, 40(16):6266–6282.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of CVPR*, pages 580–587.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

Daena J Goldsmith. 2007. Brown and levinsons politeness theory. *Explaining communication: Contemporary theories and exemplars*, pages 219–236.

Yueguo Gu. 1990. Politeness phenomena in modern chinese. *Journal of pragmatics*, 14(2):237–257.

Dániel Z Kádár and Michael Haugh. 2013. *Understanding politeness*. Cambridge University Press.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *arXiv preprint arXiv:1602.08952*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *Proceedings of ICLR Workshop*.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence*, 22(2):110–125.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1097–1105.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of NAACL*.

Miriam A Locher and Richard J Watts. 2005. Politeness theory and relational work. *Journal of Politeness Research. Language, Behaviour, Culture*, 1(1):9–33.

Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of CVPR*, pages 5188–5196. IEEE.

Rodrigo Moraes, Joao Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, page 271.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Bach, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of ICLR Workshop*.

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

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of ECCV*, pages 818–833. Springer.

Does 'well-being' translate on Twitter?

Laura K. Smith[1] **Salvatore Giorgi**[1] **Rishi Solanki**[2] **Johannes C. Eichstaedt**[1]
H. Andrew Schwartz[3] **Muhammad Abdul-Mageed**[4] **Anneke Buffone**[1] and **Lyle H. Ungar**[5]
[1]Department of Psychology, University of Pennsylvania
[2]Electrical and Systems Engineering, University of Pennsylvania
[3]Computer Science, Stony Brook University
[4]Library, Archival and Information Studies, University of British Columbia
[5]Computer and Information Science, University of Pennsylvania
lasm@sas.upenn.edu, sgiorgi@sas.upenn.edu

Abstract

We investigate whether psychological well-being translates across English and Spanish Twitter, by building and comparing source language and automatically translated weighted lexica in English and Spanish. We find that the source language models perform substantially better than the machine translated versions. Moreover, manually correcting translation errors does not improve model performance, suggesting that meaningful cultural information is being lost in translation. Further work is needed to clarify when automatic translation of well-being lexica is effective and how it can be improved for cross-cultural analysis.

1 Introduction

Interest in sentiment analysis spans academic and commercial domains, with wide-ranging applications (Pang and Lee, 2008; Liu, 2012). While the majority of tools for sentiment analysis have been developed for English text, ideally sentiment and emotion could be analyzed across many languages. Does one need to build models for each language of interest, or can models be applied cross-culturally? More generally, how do cultures differ in the language they use to express sentiment and feeling?

Sentiment in resource-poor languages has commonly been assessed by first translating text into English and then applying an English sentiment model (Mohammad et al., 2016). This approach is economical and efficient, as building each model of interest in every target language is resource-intensive. Yet it is not clear how much culturally specific information and accuracy are lost in the translation process, and specifically how this varies across languages, cultures, linguistic content, and corpora (e.g., social media vs. news). While extensive work has demonstrated that automatic machine translation (MT) methods are competitive when translating opinion in news and blogs, less research has examined the translation of sentiment on social media, and specifically on Twitter, known for its restriction of individual exchanges to short samples of text (140 characters) and informal language. Moreover, research has not focused on translating subjective well-being specifically.

Beyond sentiment, this paper investigates how expressions of personal well-being translate between English and Spanish on Twitter. We have English and Spanish speakers annotate Tweets in their native language for five components of subjective well-being (positive emotion, engagement, positive relationships, meaning, and accomplishment) (Seligman, 2011). We then compare how well models trained and tested in the same language compare to (a) models developed in one language, and then translated (using Google Translate) to the other language (e.g., how well English models translated to Spanish work on Spanish Tweets) and (b) how well models developed in one language work on Tweets translated from another language (e.g., how well English models work on Tweets translated from Spanish to English).

2 Related Work

There is a vast literature on sentiment analysis which space precludes us from surveying; see (Liu, 2012)

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2042–2047,
Austin, Texas, November 1-5, 2016.

for an excellent overview. A small but rapidly growing camp is developing methods to estimate personality and emotion, asking "how does she feel?" rather than "how much does she like the product?" (Mohammad and Kiritchenko, 2015; Park et al., 2014). In social media, the well-being of individuals as well as communities has been studied, on various platforms such as Facebook and Twitter (Bollen et al., 2011; Schwartz et al., 2013; Eichstaedt et al., 2015; Schwartz et al., 2016).

2.1 Translating sentiment

Past work has, on the whole, regarded state-of-the-art automatic translation for sentiment analysis optimistically. In assessing statistical MT, (Balahur and Turchi, 2012) found that modern SMT systems can produce reliable training data for languages other than English. Comparative evaluations between English and Romanian (Mihalcea et al., 2007) and English and both Spanish and Romanian (Banea et al., 2008) based on the English MPQA sentiment data suggest that, in spite of word ambiguity in either the source or target language, automatic translation is a viable alternative to the construction of models in target languages. Wan (2008) shows that it is useful to improve a system in a target language (Chinese) by applying ensemble methods exploiting sentiment-specific data and lexica from the target language and a source language (English). More recent work has examined how sentiment changes with translation between English and Arabic, also finding that automatic translation of English texts yields competitive results (Abdul-Mageed and Diab, 2014; Mohammad et al., 2016). However, translated texts tend to lose sentiment information such that the translated data is more neutral than the source language (Salameh et al., 2015).

It is less obvious how well expressions of emotion or subjective well-being translate between languages and cultures; the words for liking a phone or TV may be more similar across cultures than the ones for finding life and relationships satisfying, or work meaningful and engaging.

2.2 Well-being

In contrast to classic sentiment analysis, well-being is not restricted to positive and negative emotion. In 2011, the psychologist Martin Seligman proposed PERMA (Seligman, 2011), a five-dimensional model of well-being where 'P' stands for positive emotion, 'E' is engagement, 'R' is positive relationships, 'M' is meaning, and 'A' is a sense of accomplishment. PERMA is of interest to this translation context because while the 'P' dimension maps relatively cleanly onto traditional conceptions of sentiment (i.e., positive and negative emotion), PERMA also includes social and cognitive components which may be expressed with more variation across languages and cultures. In recent work, Schwartz et al. (2016) developed an English PERMA model using Facebook data. In this paper, we adopt a similar method when building our message-level models over Tweets.

Governments around the world are increasingly dedicating resources to the measurement of well-being to complement traditional economic indicators such as gross domestic product. Being able to measure well-being across regions is not only becoming more important for institutions and policymakers, but also for private sector entities that want to assess and promote the well-being of their organizations and customers. This raises the importance of translation, given that resources for the measurement of well-being are disproportionately available in English.

3 Methods

We collected Spanish data using the Twitter API, gathering 15.3 million geolocated Tweets between September and November 2015 using a latitude/longitude bounding box around Spain. This set was reduced to messages containing only Spanish using the Language Identification (LangID) Python package (Lui and Baldwin, 2012). We restricted to messages with an 80% or higher Spanish confidence score as given by LangID. This resulted in 6.1 million Tweets from 290,000 users. We selected 5,100 random messages from this set for annotation. English Tweets were similarly collected using the Twitter API, restricted to the US, and filtered to be (primarily) in English.

3.1 Annotating message-level data

Amazon's Mechanical Turk (MTurk) was used to annotate the 5,000 random English (American)

Tweets[1]. CrowdFlower, an online crowdsourcing platform similar to MTurk, but more widely used in Europe, was used to annotate our 5,100 random Spanish Tweets[1]. As the Tweets exclusively came from Spain, raters were restricted to fluent Spanish speakers who live in Spain.

On both MTurk and CrowdFlower, separate annotation tasks were set up for each of the 10 PERMA components (positive and negative dimensions for the 5 components). Workers were given the definition of the PERMA construct, directions on how to perform the task, and were presented with an example annotation task. During the task workers were asked to indicate "to what extent does this message express" the construct in question on a scale from 1 ("Not at all") to 7 ("Extremely"). Directions were presented in English for the English task, and in Spanish for the Spanish task. The Spanish instructions were translated manually from English by a bilingual English-Spanish speaker and verified by an additional bilingual speaker.

In the English task, two raters assessed each message. If the raters disagreed by more than 3 points, a rating was obtained from a third rater. It proved more difficult to get raters for the Spanish task, even on CrowdFlower. In some cases we were unable to obtain even a single annotation for a given Tweet and PERMA component.

3.2 Developing weighted lexica

Tweets were tokenzied using an emoticon-aware tokenizer, 'happy fun tokenizer'[1]. We then extracted unigrams and bigrams from each corpus, yielding vocabularies of 5,430 and 4,697 'words' in English and Spanish, respectively. The presence/absence of these unigrams and bigrams in each Tweet were used as features in Lasso (L1 penalized regression) (Tibshirani, 1996) models to predict the average annotation score for each of the crowdsourced PERMA labels. Separate models, each consisting of regression weights for each term in the lexicon, were built for each of the ten (five positive and five negative) PERMA components in both English and Spanish[1]. Each model was validated using 10-fold cross validation, with Pearson correlations averaged over the 10 positive/negative PERMA components. Results are presented in Table 1. The models were then transformed into a predictive lexicon using the methods described in (Sap et al., 2014), where the weights in the lexicon were derived from the above Lasso regression model.

Model	*r*
Spanish	0.36
English	0.36

Table 1: Performance as measured by Pearson *r* correlation averaged over the 10 positive/negative PERMA components using 10-fold cross validation.

3.3 Translating the models

We used Google Translate to translate both the original English and Spanish Tweets and the words in the models. We also created versions of the translated models in which we manually corrected apparent translation errors for 25 terms with the largest regression coefficients for each of the 10 PERMA components (the top 250 terms for each model).

3.4 Comparative evaluation

We evaluated how well the different models worked, computing the Pearson correlations between message-level PERMA scores predicted from the different models and the ground-truth annotations. Lexica were built on 80% of the messages and then evaluated on the remaining 20%. Figure 1 shows test accuracies. Comparing the English and Spanish source language and machine translated models, we observe substantially better performance when models were built over the same language they are applied to, i.e., using models built in Spanish to predict on Spanish Tweets. Translating the models (e.g., translating an English model to Spanish and using it on Spanish Tweets) or translating the Tweets (e.g., translating Spanish Tweets to English and using an English model) work substantially less well, with translating the Tweets giving marginally better performance than translating the models. Finally, we translate both the model and Tweets, giving slightly better performance than translating the Tweets alone. Complete PERMA lexica were then built over the entire message sets for public release.

3.5 Error Analysis

To quantify the errors in translation, we took the 25 top-weighted words in each component of the

[1] Available at www.wwbp.org.

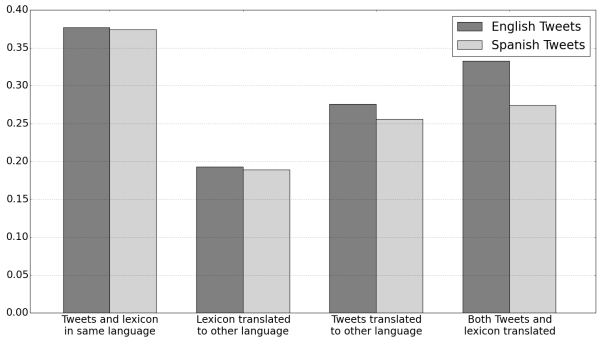


Figure 1: Performance (Pearson *r* correlation) between ground-truth annotations and predicted lexica scores averaged over the 10 PERMA components.

PERMA lexicon (250 terms total) and manually translated them with the help of a native Spanish speaker. The manual translations were then compared against the automatic translations. Out of the top 25 words we calculated the percentage of correct automatic translations (when manual and automatic translations matched) and averaged the percentages across positive and negative PERMA components. The average percentage of correct translations is listed in Table 2 as *correct trans*.

These correctly translated terms were then compared to the terms in the opposite source model (i.e., after translating English PERMA to Spanish, we compared the translations with Spanish PERMA). We calculated the percentage of the top 250 translated words missing in the 250 top words of the source lexicon for each PERMA component and averaged over the 10 components. This value is reported in Table 2 as *missing terms*. For terms that appeared in both the translated and source lexica we compared their respective weights, calculating both percentage of terms in which the weights were of different signs and percentage of terms with substantially different weights. Again, these percentages were averaged over the 10 PERMA components. Percentages are reported in Table 2 as *opp sign* and *weight diff*, respectively. To be considered "substantially different" the two weights must differ by a factor of 2. It is worth noting that at no point were the translated and source weights equal (within a tolerance of 10^{-5}).

We then looked at the errors term by term. Out of the 500 terms considered (top 250 words per source language) only one term weight changed signs between models: "mundo" (world). The weight for this term in the translated English to Spanish model was 0.42 whereas the weight in the Spanish model was -0.18, amounting to a 140% change. Next, for each source model we report terms with the largest and smallest differences in weight. These terms and weights are reported in Table 3. The language abbreviation ("en" or "es") listed under each PERMA component is used to denote the source language we translated from. For example, (en) indicates that we started with English PERMA, translated it into Spanish and then compared to Spanish PERMA.

source lang	correct trans	missing terms	opp sign	weight diff
English	83%	81%	0.5%	6.9%
Spanish	74%	91%	0.0%	4.8%

Table 2: Summary of translation errors. Percentages are averaged over the 10 PERMA components. *Source lang* is the language of the model which was translated, *correct trans* is the percentage of correct automatically translated words, *missing terms* is the percentage of correct automatic translations within the 250 top terms that did not appear in the top 250 words of other source model, *opp sign* is the percentage of terms whose sign switched between models, and *weight diff* is the percentage of terms whose weights between the two models were off by a factor of two.

PERMA	term	weight (en)	weight (es)	% chg
POS_M (en)	mundo* (world)	0.42	-0.18	143
NEG_A (en)	odio** (hate)	0.29	2.19	87
NEG_M (en)	nadie*** (no one)	0.23	0.24	4.2
NEG_R (es)	sad** (triste)	1.70	0.0012	100
NEG_P (es)	hate*** (odio)	1.81	1.75	3.3

Table 3: Examples of specific errors. Error types are denoted by asterisks: * denotes a change in sign, ** denotes the largest change in weight and *** denotes the smallest change in weight per source model. Language listed under each PERMA category is the language of the source model that was translated. The *% chg* column is percentage change relative to the larger weight. For clarity, under each term we include its translation.

4 Discussion

The difference in performance between source and machine translated models can be attributed to a few main problems. First, the translation might be inaccurate (e.g., from our corpus, "te" is not in fact "tea"). We manually corrected translation errors in the prediction models with the help of a native Spanish speaker, but found that translation error accounts for marginal discrepancy between the source language and machine translated models.

A second source of errors are translations which are technically accurate, yet do not translate culturally. For instance, even though "andaluces" translated correctly into "Andalusians," "Andalusia" (an autonomous community in Spain) does not invoke the same cultural meaning in English as it does for Spaniards. A machine would be hard-pressed to translate "Andalusia" into a relevant region within the U.S. that might invoke similar popular sentiment. Although Spanish and American people share some holidays, musicians, and sports heroes, many of these differ (e.g., "Iker Casillas" is not well known in the U.S. and "La selectividad" may be similar to the "SATs," but this is not captured in MT).

A third source of error stems from cultural differences, with certain topics resonating differently cross-culturally. For instance, when comparing the highest weighted positive terms across PERMA, religious language (e.g., "god," "blessed") appears in English but not Spanish, fitting with the popular notion that Americans are more religious than Europeans. Spanish PERMA's positive emotion component contains multiple highly weighted instances of laughter; none have high weights in the English model. Highly weighted English negative emotion terms are marked by general aggression (e.g., "kill," "stupid") whereas the highest weighted Spanish terms include derogatory terms for disliked people (e.g., "douchebag," "fool"). The American positive relationship component is marked by words like "friend" and "friends," while "sister" is weighted more highly in Spanish PERMA.

Note that this is fundamentally a problem of domain adaptation rather than MT, as our error analysis revealed that the majority of top-weighted terms were exclusive to one source model. Different cultures use different words (or at least vastly different word frequencies) when revealing the same kind of well-being. Exploring where the sentiment around a similar concept diverges across languages can provide insight to researchers studying cross-cultural variation.

4.1 Limitations

This work has significant limitations. First, the English and Spanish annotation processes, though kept as similar as possible, were not identical; annotations were gathered on different platforms, and due to our difficulty in recruiting Spanish raters, our total annotations per message varied across tasks. Additionally, the models were built over relatively small corpora of 5,000 English Tweets and 5,100 Spanish Tweets. These Tweets came from different time periods, which may further reduce similarity between the Spanish and English corpora. Finally, our method does not account for the presence of various sub-cultures within the United States and Spain.

5 Conclusion

In this work, we investigated how well expressions of subjective well-being translate across English and Spanish Twitter, finding that the source language models performed substantially better than the machine translated versions. Moreover, manually correcting translation errors in the top 250 terms of the lexica did not improve model performance, suggesting that meaningful cultural information was lost in translation.

Our findings suggest that further work is needed to understand when automatic translation of language-based models will lead to competitive sentiment translation on social media and how such translations can be improved. Cultural differences seem more important than language differences, at least for the tasks we studied here. We expect that language indicators of personality and emotion will similarly translate poorly, but that remains to be studied.

Acknowledgments

The authors acknowledge support from the Templeton Religion Trust (grant TRT-0048) and Bioibérica.

References

Muhammad Abdul-Mageed and Mona T Diab. 2014. Sana: A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis. In *Proceedings of the 9th edition of the Language Resources and Evaluation Conference*, LREC, pages 1162–1169.

Alexandra Balahur and Marco Turchi. 2012. Multilingual sentiment analysis using machine translation? In *Proceedings of the 3rd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA, pages 52–60.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 127–135.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*, ICWSM, pages 450–453.

Johannes C Eichstaedt, H Andrew Schwartz, Margaret L Kern, Gregory Park, Darwin R Labarthe, Raina M Merchant, et al. 2015. Psychological language on twitter predicts county-level heart disease mortality. *Psychological Science*, 26(2):159–169.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, ACL, pages 25–30.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 976–983.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How translation alters sentiment. *Journal of Artificial Intelligence Research*, 55:95–130.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Greg Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, David J Stillwell, Michal Kosinski, et al. 2014. Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*, 108:934–952.

Mohammad Salameh, Saif M Mohammad, and Svetlana Kiritchenko. 2015. Sentiment after translation: A case-study on arabic social media posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL, pages 767–777.

Maarten Sap, Greg Park, Johannes C Eichstaedt, Margaret L Kern, David J Stillwell, Michal Kosinski, et al. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods In Natural Language Processing*, EMNLP, pages 1146–1151.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Richard E Lucas, Megha Agrawal, et al. 2013. Characterizing geographic variation in well-being using tweets. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, ICWSM.

H Andrew Schwartz, Maarten Sap, Margaret L Kern, Johannes C Eichstaedt, Adam Kapelner, Megha Agrawal, et al. 2016. Predicting individual well-being through the language of social media. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 516–527.

Martin EP Seligman. 2011. *Flourish*. Free Press, New York, NY.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Xiaojun Wan. 2008. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 553–561.

Beyond Canonical Texts: A Computational Analysis of Fanfiction

Smitha Milli
Computer Science Division
University of California, Berkeley
smilli@berkeley.edu

David Bamman
School of Information
University of California, Berkeley
dbamman@berkeley.edu

Abstract

While much computational work on fiction has focused on works in the literary canon, user-created fanfiction presents a unique opportunity to study an ecosystem of literary production and consumption, embodying qualities both of large-scale literary data (55 billion tokens) and also a social network (with over 2 million users). We present several empirical analyses of this data in order to illustrate the range of affordances it presents to research in NLP, computational social science and the digital humanities. We find that fanfiction deprioritizes main protagonists in comparison to canonical texts, has a statistically significant difference in attention allocated to female characters, and offers a framework for developing models of reader reactions to stories.

1 Introduction

The development of large-scale book collections—such as Project Gutenberg, Google Books, and the HathiTrust—has given rise to serious effort in the analysis and computational modeling of fiction (Mohammad, 2011; Elsner, 2012; Bamman et al., 2014; Jockers, 2015; Chaturvedi et al., 2015; Vala et al., 2015; Iyyer et al., 2016). Of necessity, this work often reasons over historical texts that have been in print for decades, and where the only relationship between the author and the readers is mediated by the text itself. In this work, we present a computational analysis of a genre that defines an alternative relationship, blending aspects of literary production, consumption, and communication in a single, vibrant ecosystem: fanfiction.

Fanfiction is fan-created fiction based on a previously existing, original work of literature. For clarity we will use the term CANON to refer to the original work on which a fanfiction story is based (e.g. Harry Potter) and the term STORY to refer to a single fan-authored story for some canon.

Although stories are based on an original canonical work and feature characters from the canon, fans frequently alter and reinterpret the canon—changing its setting, playing out an alternative ending, adding an original character, exploring a minor character more deeply, or modifying the relationships between characters (Barnes, 2015; Van Steenhuyse, 2011; Thomas, 2011).

In this work, we present an empirical analysis of this genre, and highlight several unique affordances this data presents for contemporary research in NLP, computational social science, and the digital humanities. Our work is the first to apply computational methods to fanfiction; in presenting this analysis, we hope to excite other work in this area.

2 Fanfiction data

Our data, collected between March–April 2016, originates from `fanfiction.net`.[1] In this data, AUTHORS publish stories serially (one chapter at a time); REVIEWERS comment on those chapters.

A summary of data is presented in table 1. The scale of this data is large for text; at 55 billion to-

[1] While terms of service prohibit our release of this data, tools to collect and process it can be found here: `http://github.com/smilli/fanfiction`.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2048–2053,
Austin, Texas, November 1-5, 2016.

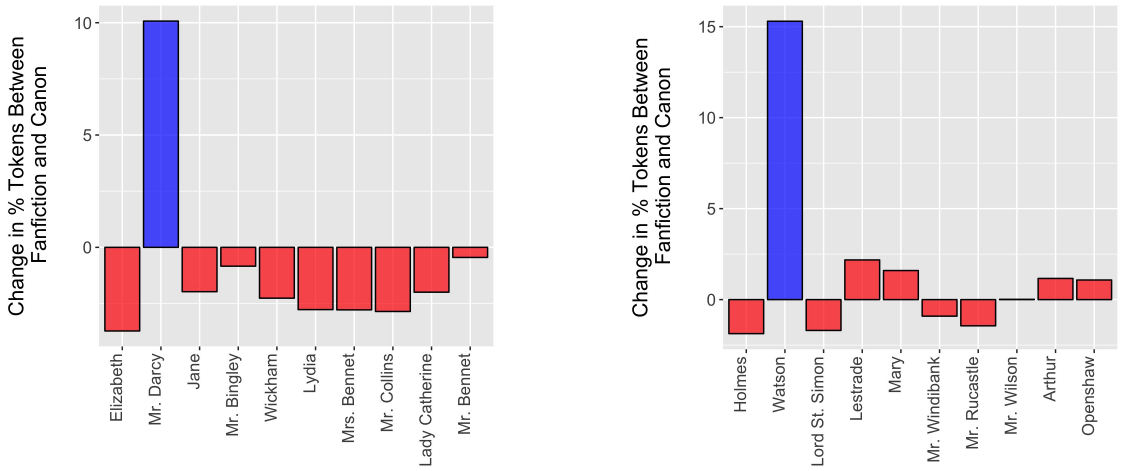


Figure 1: Difference in percent character mentions between fanfiction and canon for *Pride and Prejudice* (left) and *Sherlock Holmes* (right).

kens, it is over 50 times larger than the BookCorpus (Zhu et al., 2015) and over 10% the size of Google Books (at 468B tokens).

The dataset is predominantly written in English (88%), but also includes 317,011 stories in Spanish, 148,475 in French, 102,439 in Indonesian, and 73,575 in Portuguese. In total, 44 different languages are represented.

Type	Num of Type
Canons	9,246
Stories	5,983,038
Tokens	55,264,185,653
Reviews	159,914,877
Users	2,093,601
–Authors	1,364,729
–Reviewers	1,438,721
Languages	44

Table 1: Summary of the fanfiction.net corpus

3 Analysis of fanfiction

3.1 Differences between canons and fanfiction

The systematic ways in which fanfiction stories differ from their canonical works can give insight into the characteristics of a story that are desired by fans but may be missing from the mainstream canon. We investigate two questions: 1.) Is there a difference between the characters emphasized in fanfiction compared to the original canon? And 2.) Is gender presented differently in these two sources?

Character differences. In order to explore the differing attention to character, we consider fanfiction from ten canons whose original texts appear in Project Gutenberg; we selected the ten canons from unique authors with the most fanfiction stories associated with them.[2] To extract and compare characters, we run BookNLP (Bamman et al., 2014) on both the canonical work and the top 100 stories (by number of reviews) from each canon, and pair characters across canon/fanfiction with the same name.

To measure how the importance of individual characters varies between canon and fanfiction, we calculate the change in the percent of all character mentions a character has in the canon to the average percent of character mentions that same character has in fanfiction.

Across all canons we find that the most prominent character in the canon had at most a small increase in percent character mentions, while less prominent characters received large increases. The results for two illustrative examples, *Pride and Prejudice* and *Sherlock Holmes*, are shown in Figure 1. The percent of character mentions for the main protagonists (Elizabeth and Holmes) decreases in fanfiction, but the secondary characters of Mr. Darcy and Watson

[2] *Les Miserables* (3996 fanfiction stories), *Sherlock Holmes* (3283), *Pride and Prejudice* (3084), *Peter Pan* (2431), *Alice in Wonderland* (1446), *Anne of Green Gables* (620), *Jane Eyre* (331), *Little Women* (286), *The Scarlet Pimpernel* (255), and the *Secret Garden* (222).

Labels	Terms
Author encouragement	read story one reading chapters time best ever review long update please love soon story amazing really hope continue writing chapter great good keep really work story job forward awesome ca wait next chapter see na happens gon great read like well really story love chapter way one see interesting
Requests for story	would like know get think going could something really even
Emotional reactions	wow better beautiful getting fight adorable keeps team birthday tears oh god yes man yay damn hell dear yeah got poor lol cute howl evil bad hate baby feel lord xd loved funny love haha sweet lol ah cute aww

Table 2: Top 10 terms in the 10 manually grouped LDA topics.

show a large increase. These findings confirm the results of Xu et al. (2011), who find a greater increase in mentions of Mr. Darcy relative to Elizabeth in a different corpus of *Pride and Prejudice* fanfiction, and supports claims that fanfiction authors may delve deeper into characters that receive less attention in the canon (Jwa, 2012; Thomas, 2011).

Gender differences. Fanfiction has a predominantly female authorship and readership base (Barnes, 2015); these stories often oppose traditional gender norms present in the canon and showcase stronger female characters (Handley, 2012; Scodari and Felder, 2000; Leow, 2011; Busse, 2009).

In order to test whether fanfiction allocates more attention to female characters than canonical stories, we compare the percent of mentions of male and female characters using the same collection of stories from Gutenberg canons as above. 40.1% of character mentions in the canons are to women; in fanfiction, this ratio increases to 42.4%. This effect size is small (2.3% absolute difference), but in a bootstrap hypothesis test of the difference (using 10^6 bootstrapped samples), the difference is highly significant ($p < 0.001$), suggesting that fanfiction does indeed devote more attention to female characters.

3.2 Interaction between users

A unique characteristic of this data is the chapter-by-chapter reader reviews; any user can leave a review for a chapter of a story. Authors are also frequently reviewers of other stories (Thomas, 2011), forming an ecosystem with qualities of a social network.

709,849 authors in this data (52%) are also reviewers; if we define a network node to be a user and edge to be a reviewing relationship (a directed edge exists from $A \rightarrow B$ if A reviews one of B's works), this network contains 9.3M such directed edges, with an average outdegree of 13.2 and indegree of 15.6 (each author on average reviews for 13 other authors, and is reviewed by 16).

To explore the content of these reviews computationally, we sampled one story with more than 500 reviews from 500 different canons and ran Latent Dirichlet Allocation (Blei et al., 2003) with 10 topics on the text of the reviews (excluding names as stopwords). This is an exploratory analysis, but can give insight into the broad functions of reader responses in this domain.

Table 2 presents the results, grouping the topics into three exploratory categories: positive encouragement and pleas for updates, requests to the author about the progression of the story, and emotional reactions to the story. Prior studies that have examined the role of reviews as a form of interaction between the reader and the author have documented the first two categories extensively (Campbell et al., 2016; Magnifico et al., 2015; Lammers and Marsh, 2015; Black, 2006). However, despite a significant portion of the reviews consisting of the reader's emotional responses to the story, the way in which readers use reviews as a means of emotional engagement with the story itself has yet to be examined in such detail.

3.3 Predicting reader responses to character

The presence of reader reviews accompanying each fanfiction chapter presents a unique opportunity to develop a predictive model of how readers respond to text—given the text of a chapter, can we predict

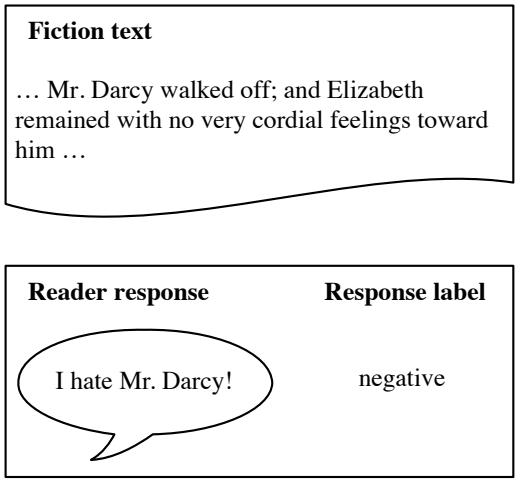


Figure 2: Illustration of data for predicting reader responses. Here we are using features derived only from FICTION TEXT to predict the RESPONSE LABEL.

how readers will react?

To test the feasibility of this task, we focus on reader responses to character. A RESPONSE to a character is operationalized as a sentence from a reader review mentioning that character and no other characters. We create an annotated dataset by randomly sampling a single character with at least 10 reader responses from each of the 500 stories described in §3.2. From this set, we randomly select exactly 10 responses for each character, to yield a total of 5,000 reader responses.

We then present these 5,000 responses to annotators on Amazon Mechanical Turk, and ask them to judge the sentiment *toward the character* expressed in the response as either positive, negative, neutral, or not applicable (in the case of character recognition errors). The overall agreement rate among annotators for this classification task is moderate (Fleiss' κ = 0.438), in part due to the difficulty of assessing the responders' attitudes from short text; while some responses wear their sentiment on their sleeve (*I knew I hated Brandon!*), others require more contextual knowledge to judge (*Ah Link* or *Akira appears!*).

In order to create a higher-precision subset we select responses with only unanimous positive or negative votes from 3 different annotators, yielding a total dataset of 1,069 response labels. We divide the dataset into 80% training/development and 20% for a held-out test (with no overlap in stories between training and test).

We also bootstrap additional semi-supervised data by training a sentiment classifier on the unigrams of the reader responses in the training data (with a 3-class accuracy of 75%; compared to majority baseline of 49.7%), predicting the sentiment label for all responses in the dataset, and selecting examples that a.) have 95% prediction confidence and b.) whose stories do not appear in the training or test data. We sample selected examples to respect the label distribution in the training data, yielding an additional 25,000 data points to supplement learning.

Our core task is to use *only* the text of the story (and not the reader response) to predict the corresponding response sentiment label in order to understand what aspects of the story (and a character's role within it) readers are reacting to. We experiment with several features to represent the characters:

- AGENT, PATIENT, PREDICATIVE, POSSESSIVE relations for each character (as output by BookNLP), both in the specific chapter and in the book overall (under the rationale that readers are responding to the *actions* that characters take).
- Unigrams spoken by the character, both in the chapter and in the book overall.
- Character gender.
- Character's frequency in the book (binary indicators of the decile of their frequency of mention).
- Skip-gram representations trained on 4.1B words of fanfiction text (200-dimensional, grouped into 1000 clusters using k-means clustering); these cluster identities form additional binary features by replacing the lexical indicators for the text features above.

We perform model selection using tenfold cross-validation on the training data alone, training ℓ_2-regularized logistic regression on one partition of the data, tuning the ℓ_2 regularization parameter on another, and assessing performance on a third (note none of the test data described above is seen during this stage).

A majority class (all-positive) baseline on the training data results in an accuracy rate of 75.6%;

only syntactic features yield a significant improvement, achieving an accuracy rate of 80.5%.

Table 3 lists the most informative features for predicting negatively-assessed characters. While these characteristic features have face validity and offer promise for understanding character in more depth, we do not see similar improvements in accuracy on the truly held-out test data (run once before submission); this same feature set achieves an accuracy rate of 70.4% (compared to a majority baseline on the test data of 71.4%). Part of this may be due to the sample size of the test data ($n = 199$); a bootstrap 95% confidence interval (Berg-Kirkpatrick et al., 2012) places the accuracy in the range [0.648, 0.754]. However, this more likely constitutes a negative result that reflects the inherent difficulty of the task; while syntactic features point to a real signal that readers are reacting to when writing their responses, literary character is of course far more complex, and more sophisticated representations of character—and of the readers who react to them—are likely warranted for real predictive performance on this task.

agent	patient	predicative	possessive
hissed	hate	pregnant	phone
sneered	done	human	state
shoved	see	afraid	tone
glared	hated	stubborn	face
paused	asked	person	spot
respond	face	boy	plan
caught	pissed	angry	wand
scowled	blame	stupid	pain
walked	shocked	free	emotions
had	used	mother	chakra

Table 3: Syntactic features most predictive of a negatively-assessed character.

4 Conclusion

In blending aspects of large-scale literary data and social network structure, fanfiction publication constitutes a vibrant ecosystem with ample textual evidence for the production and consumption of literary texts. In this work, we have briefly illustrated three aspects of this data that have the potential to yield interesting insight—the relationship between fanfiction stories and their original source material, the social network structure of authors and their respondents, and the possibility of predicting reader responses from serially published text. Many questions remain; in providing a quantitative description of this dataset, we hope to highlight its potential for analysis, and encourage other work in this domain.

Acknowledgments

We thank Cecilia Aragon and our anonymous reviewers for their helpful comments. This work is made possible by the use of computing resources from Berkeley Research Computing.

References

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland, June. Association for Computational Linguistics.

Jennifer L Barnes. 2015. Fanfiction as imaginary play: What fan-written stories can tell us about the cognitive science of fiction. *Poetics*, 48:69–82.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 995–1005, Stroudsburg, PA, USA. Association for Computational Linguistics.

RW Black. 2006. Not just the OMG standard: reader feedback and language, literacy, and culture in online fanfiction. In *Annual Meeting of The American Educational Research Association, San Francisco*, volume 10.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

Kristina Busse. 2009. In focus: Fandom and feminism: gender and the politics of fan production. *Cinema Journal*, 48(4):104–108.

Julie Campbell, Cecilia Aragon, Katie Davis, Sarah Evans, Abigail Evans, and David Randall. 2016. Thousands of positive reviews: Distributed mentoring in online fan communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 691–704, New York, NY, USA. ACM.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daume, and Chris Dyer. 2015. Modeling dynamic relationships between characters in literary novels.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *EACL*.

Christine Handley. 2012. Distressing damsels: narrative critique and reinterpretation in star wars fanfiction. *Fan Culture: Theory/Practice, Newcastle upon Tyne: Cambridge Scholars Publishing*, pages 97–118.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *North American Association for Computational Linguistics*.

Matthew Jockers. 2015. Revealing sentiment and plot arcs with the syuzhet package. `http://www.matthewjockers.net/2015/02/02/syuzhet/`.

Soomin Jwa. 2012. Modeling L2 writer voice: Discoursal positioning in fanfiction writing. *Computers and Composition*, 29(4):323–340.

Jayne C Lammers and Valerie L Marsh. 2015. Going public: An adolescent's networked writing on fanfiction.net. *Journal of Adolescent & Adult Literacy*, 59(3):277–285.

Hui Min Annabeth Leow. 2011. Subverting the canon in feminist fan fiction. *Transformative Works and Cultures*, 7.

Alecia Marie Magnifico, Jen Scott Curwood, and Jayne C Lammers. 2015. Words on the screen: broadening analyses of interactions among fanfiction writers and reviewers. *Literacy*, 49(3):158–166.

Saif Mohammad. 2011. From once upon a time to happily ever after: Tracking emotions in novels and fairy tales. *CoRR*, abs/1309.5909.

Christine Scodari and Jenna L Felder. 2000. Creating a pocket universe: Shippers, fan fiction, and the X-Files online. *Communication Studies*, 51(3):238–257.

Bronwen Thomas. 2011. What is fanfiction and why are people saying such nice things about it? *Storyworlds: A Journal of Narrative Studies*, 3(1):1–24.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 769–774, Lisbon, Portugal, September. Association for Computational Linguistics.

Veerle Van Steenhuyse. 2011. The writing and reading of fan fiction and transformation theory. *CLCWeb: Comparative Literature and Culture*, 13(4):4.

Jun Xu. 2011. Austen's fans and fans' Austen. *Journal of Literary Semantics*, 40(1):81–97.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.

Using Syntactic and Semantic Context to Explore Psychodemographic Differences in Self-reference

Masoud Rouhizadeh[†§], **Lyle Ungar**[§], **Anneke Buffone**[§], **H Andrew Schwartz**[†§]
[†]Stony Brook University, [§]University of Pennsylvania
mrouhizadeh@gmail.com, ungar@cis.upenn.edu, buffonea@sas.upenn.edu, has@cs.stonybrook.edu

Abstract

Psychological analysis of language has repeatedly shown that an individual's rate of mentioning 1st person singular pronouns predicts a wealth of important demographic and psychological factors. However, these analyses are performed out of context — syntactic and semantic — which may change the magnitude or even direction of such relationships. In this paper, we put "pronouns in their context", exploring the relationship between self-reference and age, gender, and depression depending on syntactic position and verbal governor. We find that pronouns are overall more predictive when taking dependency relations and verb semantic categories into account, and, the direction of the relationship can change depending on the semantic class of the verbal governor.

1 Introduction

Approximately 1 in 18 English words on Facebook are first-person singular pronouns.[1] Extensive work in psychological analyses of language has consistently found strong relation between first-person pronoun use and psychological attributes of individuals (Kendall, 1998; Pennebaker and Stone, 2003; Pennebaker, 2011; Twenge et al., 2012; Oishi et al., 2013; Carey et al., 2015). Although such findings have been replicated extensively, little is known about how the syntactic or semantic context of the pronouns may affect their relationship with human traits. Usage in subject or object position may vary, and the type of verb governing the reference may further change its relationship. For instance, while younger individuals are more likely to use 1st-person singular pronouns overall, older individuals may be more likely to use them as the subject of social verbs.

In this study we dive deep into this one type of word which makes up a large portion of our daily lives. We first look at the relationship between first person singular pronouns and age, gender, and depression. We then consider the syntactic position of the pronoun and its occurrence in the subject and direct object position. Next, we explore the self-referenced use of verbs compared to their general use across different semantic categories, followed by an examination of the rate of 1st-person singular pronoun as the subject and the object with different verb categories.

We ultimately show that pronoun relationships with human outcomes can change drastically depending on their syntactic position and the category of their verbal governor. To be more specific, our contributions include: (a) taking the role of context into account in the psychological analysis of personal pronouns, (b) distributional clustering of verbs using Canonical Correlation Analysis (CCA), and (c) exploring the integration of verbal semantic categories in the analysis of pronouns. Utilizing verb categories instead of actual verbs, enables generalization and less sparsity in the semantic comparison of the contexts in which personal pronouns are used.

[1]Within the study dataset, 5.45% of all words from self-identified English speakers were first-person pronouns.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2054–2059,
Austin, Texas, November 1-5, 2016.

2 Background

A wealth of studies have explored pronoun use with regard to age, gender, and personality types. In fact, a whole book, "The Secret Life of Pronouns" has been dedicated summarizing such studies which have built up over several decades of work (Pennebaker, 2011). [2]

We could not come close to a full survey of such work, but rather list some of the most notable and recent results for outcomes related to those of this study. Pennebaker et. al. (2003) and Chung & Pennebaker (2007) found that the use of self-references (i.e. 'I', 'me') decreases over age. Pennebaker et. al. (2003), and Argamon et. al. (2007) showed that females use significantly more first-person singular personal pronouns compared to males. Bucci and Freedman (1981), Weintraub (1981), and Zimmermann et. al. (2013) found that first-person singular pronouns are positively correlated with depressive symptoms. These analyses do not take the role of syntactic and semantic context into consideration which may indicate interesting information about psychological factors.

3 Method

Data Set: Facebook Status Updates. Our dataset consists of the status updates of 74,867 Facebook users who volunteered to share their posts in the "MyPersonality" application (Kosinski et al., 2013), sent between January 2009 and October 2011. The users met the following criteria: (a) have English as a primary language, (b) indicated their gender and age, (c) be less than 65 years old (due to data sparsity beyond this age), and (d) have at least 1,000 words in their status updates (in order to accurately estimate language usage rates). This dataset contains 309 million words within 15.4 million status updates. All users completed a 100-item personality questionnaire (an International Personality Item Pool (IPIP) proxy to the NEO-PI-R (Goldberg, 1999). User-level degree of depression (DDep) was estimated as the average response to seven depression facet items (nested within the larger Neuroticism item pool of the questionnaire) (Schwartz et al., 2014).

Dependency Features. We used dependency annotations in order to determine the syntactic function of personal pronouns i.e. subject (S) and direct object (DO). We obtained dependency parses of our corpus using Stanford Parser (Socher et al., 2013) that provides universal dependencies in (*relation, head, dependent*) triples. In the next step, we extracted the words in in the nominal subject (*"nsubj"*) and direct object (*"dobj"*) positions including *nsubj* 1st-person singular pronoun *"I"*, and *dobj* 1st-person singular pronoun *"me"*. We also extracted the corresponding verbs for each of the nominal subjects, and direct object words.

Verb categorization. In order to integrate the verbal semantic categories in the syntactic analysis of pronouns, we utilize two verb categorization methods (a) linguistically-driven Levin's Verb Classes, and (b) empirically-driven verb clustering based on CCA.

Levin's verb classes (Levin, 1993) includes around 3100 English verbs classified into 47 top level, 193 second and third level classes. This classification is based on Levin's hypothesis that the syntactic behavior of a verb is influenced by its semantic properties, indicating that identifying sets of verbs with comparable behavior at the syntax level will lead to coherent clusters of semantically similar verbs. In this paper we used all of the 193 second and third level Levin's classes (*Lev*). As an alternative way, we also used the 50 top most frequent sub-classes in our social media data (*LevTop*).

To derive empirically driven clusters we use Canonical Correlation Analysis (CCA), a multi-view dimensionality reduction technique. CCA has previously been used in word clustering methods such as multi-view learning of word embeddings (Dhillon et al., 2011), or multilingual word embeddings (Ammar et al., 2016). The advantage of a multi-view technique is that we can leverage both the subject and object context. More precisely, we performed sparse CCA on matrix x that includes 5k by 10k verb-by-nominal-subject (*nsubj*) co-occurrences, and matrix z that includes 5k by 10k verb-by-direct-object (*dobj*) co-occurrences. The output of CCA is a subject by component matrix

[2] To quantify the pervasiveness of pronoun studies in social science, we consider the citation count, via Google Scholar (July, 2016), to works mentioning "pronoun" by one of the top researchers, James W. Pennebaker, which number over 10,000.

Feature Set	Gender (AUC)	Age (MSE)	Dep (MSE)
$P(1p)$	.512	78.9	90.1
$P(1p\|r)$	.589	76.4	90.3
$P(1p\|r,c)$, *Lev*	.660	70.0	89.8
$P(1p\|r,c)$, *Lev* & sent	.695	**68.3**	**89.1**
$P(1p\|r,c)$, *LevTop*	.660	71.5	89.8
$P(1p\|r,c)$, *LevTop* & sent	**.669**	69.0	89.3
$P(1p\|r,c)$, *CCA-D*	.634	73.4	90.3
$P(1p\|r,c)$, *CCA-D* & sent	.649	71.5	89.7
$P(1p\|r,c)$, *CCA-KM*	.632	72.6	90.3
$P(1p\|r,c)$, *CCA-KM* & sent	.645	70.9	89.9

Table 1: Area under the ROC curve (AUC) for gender (higher is better), and Mean Square Error (MSE) for age and depression prediction (lower is better), and the prediction using 1st-per pronoun use overall, in subject and object position, and given verb categories.

Verb Clusters	r
1st person singular pronoun use	-.17
1st person singular nominal subject	
thank, celebrate, welcome, greet, applaud	.09
shake, freeze, melt, collect, bend, twist, squeeze	.08
hate, fear, regret, dislike, despise, dread, tolerate	-.16
write, draw, type, print, scratch, plot, sketch	-.10
1st person singular direct object	
join, pool, merge	.05
deny, suspect	.04
hate, fear, regret, dislike, despise, dread, tolerate	-.09
bore, worry, scare, bother, annoy, disappoint	-.08

Table 2: Linear regression coefficient of age and 1st person pronoun use in different verb clusters.

(u: *subject-view*), and object by component matrix (v: *object-view*). We then build matrix S by multiplying x by u and matrix O by multiplying z by v to get the verbs by CCA-components from *subject-view*, and verbs by object components from *object-view* respectively. In order to cluster verbs from direct CCA components, we use the average score of *subject-view* and *object-view* components, assigning verbs to those components for which they have a non-zero absolute weight (*CCA-D*). Sparse CCA zeros-out verbs from multiple components so as to assign verbs to components, but we also explore normal CCA and cluster the verbs using k-means ($k = 30$) clustering from the z-scaled values of S and O matrices (*CCA-KM*).

Both Levin's and CCA-based verb classes are derived from syntactic behavior. As a result, they often do not distinguish antonyms. For instance, Levin's "admire" verb class contains both 'love' and 'hate". Building on research showing positive and negative emotions differ across age and gender (Schwartz et al., 2013), we integrate valence information in our verb clustering. We used positive and negative sentiment scores from EmoLex word-emotion association lexicon (Mohammad and Turney, 2013), dividing each of our clusters into *positive*, *negative*, and *neutral* sub-classes.

Analysis. We explore the use of 1st-person singular pronouns across age and gender in different syntactic and semantic contexts. Features are encoded as the mean from maximum likelihood estimation over the probability of mentioning a first person singular pronoun in a given context.

(a) The overall usage first person singular pronoun:

$$P(1p) = P(PN = 1p)$$

(b) The probability of using first person singular pronoun in the *nsubj*, and the *dobj* positions:

$$P(1p|r) = P(PN = 1p \mid rel = r)$$

where $rel \in \{nsubj, dobj\}$.

(c) The probability of using first person singular pronoun in the *nsubj*, and the *dobj* positions of a given verb category:

$$P(1p|r,c) = P(PN = 1p \mid rel = r, vcat = c)$$

where $rel \in \{nsubj, dobj\}$ and $vcat$ is the set of all verb categories being considered.

4 Evaluation

The goal of our work is to expand the knowledge of how the first-person singular pronoun, one of the most common word types in English, is related to who we are – our demographics and psychological states. We work toward this goal in an empirical fashion, by first replicating known general relationships of 1st-person singular pronouns with gender, age, and depression, exploring how their use in different syntactic positions, and, finally, by looking at relationships within specific semantic contexts according to the verb classes described earlier.

Verb Clusters	β
1st person singular pronoun use	.11
1st person singular nominal subject	
love, enjoy, respect, adore, cherish, admire	.29
miss, like, appreciate, trust, support, value	.28
destroy	-.08
kick, shoot, slap, smash, shove, slam	-.07
1st person singular direct object	
make, blow, roll, hack, cast	.22
hold, handle, grasp, clutch, wield, grip	.18
hit, kick, strike, slap, smash, smack, bang, butt	-.10
add, mix, connect, link, combine, blend	-.04

Table 3: Logistic regression coefficient between gender and 1st person singular pronoun use in different verb clusters (positive is female).

Replication. We use standardized linear and logistic regression to correlate gender, age, and depression with $P(1p)$ (first-person singular pronoun use). We control for age in the case of gender, gender in the case of age, and both gender and age in the case of depression by including them as covariates in the regression and reporting the unique coefficient for the variable in question. Logistic regression is used for gender, since it is binary, while linear regression is used for the continuous age and depression variables. Confirming past results, we found significant relationships between first-person pronoun usage and gender ($\beta = .11$, $p < .001$), age ($r = -0.17$, $p < .001$), and depression score ($r = -0.06$, $p < .01$).

Syntactic Context. Taking dependency relationships into account ($P(1p|r)$), we observed shifts in the magnitude of correlations. Specifically, we found significant negative correlations between age and using 1st-person singular pronoun in the subject ($r = -0.12$, $p < .001$), and the object positions ($r = -0.17$, $p < .001$). For gender we found a significant positive correlation between being female and the probability of using 1st-person singular pronoun ($r = 0.11$, $p < .001$), and 1st-person singular pronoun in subject position ($r = 0.16$, $p < .001$). For depression a significant positive correlation between with $P(1p)$ ($r = 0.06$, $p < .05$), and using 1st-person singular pronoun in the subject position ($r = 0.07$, $p < .05$).

Syntactic and Semantic Context. Table 1 reports the area under the ROC curve (AUC) for gender prediction and the Mean Square Error (MSE) for predicting age and depression based on $P(1p)$, $P(1p|r)$, and $P(1p|r, c)$, driven from various categorization approaches. We used AUC since it can capture more differences in performance by evaluating the class probabilities of test instances rather than just finding whether it was right or wrong. We applied 10-fold cross-validation with a linear-SVM in the case of gender, and ridge-regression in the case of depression. The obtained results reveal a consistent pattern: in gender, age, and depression prediction all the features that take context into account outperform $P(1p)$ which is the vastly reported measure of self-reference in the literature. This suggests that there is more information to be gained by utilization syntactic and semantic context. In other words, we can achieve a more meaningful, deeper insight into the relationship of subject and object position of the first person in different contexts, revealing a more complex, and more insightful set of relations.

We achieve the best performance by utilizing verb categories. We first observe that integrating sentiment helps in nearly all verb categorization approaches. Next, we see that while both CCA and Levin verb clusters yield improvement in prediction accuracy, our performance gains using the data-driven CCA-based verb clustering are not as large as that from Levin's linguistically-driven classes.

While we believe our features can improve prediction accuracy, that is not the primary application of social science research. Rather, it is correlating the behavior of referencing the self with psychological conditions, like depression, in order to gain human insights. In the case of correlating behavior with a psychological measure, Pearson coefficients above .1 are considered noteworthy and above .3 are considered approaching a "correlational upperbound" (Meyer et al., 2001).

Tables 2, 3, and 4 show the most predictive features, using the best performing clustering method (i.e. Levin & Sentiment). Note that in the case of age and gender, we see that not only does the magnitude of the relationship change, but it's possible that the direction can completely change.

For example, while males are less likely to

Verb Clusters	r
1st person singular pronoun use	.06
1st person singular nominal subject	
cry, worry, suffer, fear, bother, ache, mourn, anger	.11
scare, annoy, confuse, depress, upset, disappoint	.11
1st person singular direct object	
kill, murder, slay, slaughter, butcher	.09
scare, annoy, confuse, depress, upset, disappoint	.07

Table 4: Linear regression coefficient of depression score and 1st person singular pronoun use in different verb clusters.

use first-person singular pronouns overall, they are much more likely to use them as the subject of aggressive physical contact verbs like “kick", “shoot", “slap", and “smash", suggesting men are more likely to express themselves as agents of aggressive contact. On the other hand, women use first-person singulars in the social sphere, particularly in an affiliative context. They assert themselves as agents of empowering and encouraging others (e.g. “love", “enjoy", “cherish", “admire") and faith in others (e.g. “trust", “value", “support", “respect").

5 Conclusion

We have shown that the well-studied link between the first-person singular pronoun and human psycho-demographics is largely dependent on its syntactic and semantic context. Many theories and conclusions are built on such relationships, but here we show these relationships depend on verbal context; correlations can shrink, grow, and even change directions depending on the verbs governing the pronoun. For example, while the usage of 1st person singular pronoun decreases over age, it increases if it is used as the subject of verbs such as “thank", and “celebrate", or as the object of verbs such as “join". Similarly, while females tend to use 1st person singular pronouns more than males, they use them less often as the subject of “destroy” verbs or as the object of “hit” and “kick” verbs.

By integrating syntactic dependency relationships along with semantic classes of verbs, we can capture more nuanced linguistic relationships with human factors. Beyond pronouns, we ultimately aim to expand the regimen of *open-vocabulary* techniques available for the analysis of psychologically-relevant outcomes.

Acknowledgments

The authors acknowledge the support from Templeton Religion Trust, grant TRT-0048.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Shlomo Argamon, Moshe Koppel, James W Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: Age, gender and the varieties of self-expression. *First Monday*, 12(9).

Wilma Bucci and Norbert Freedman. 1981. The language of depression. *Bulletin of the Menninger Clinic*, 45(4):334.

Angela L Carey, Melanie S Brucks, Albrecht CP Küfner, Nicholas S Holtzman, Mitja D Back, M Brent Donnellan, James W Pennebaker, Matthias R Mehl, et al. 2015. Narcissism and the use of personal pronouns revisited. *Journal of personality and social psychology*, 109(3):e1.

Cindy Chung and James W Pennebaker. 2007. The psychological functions of function words. *Social communication*, pages 343–359.

Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24.

Lewis R Goldberg. 1999. A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. *Personality psychology in Europe*, 7(1):7–28.

Lori Kendall. 1998. Meaning and identity in âĂIJcyberspaceâĂİ: The performance of gender, class, and race online. *Symbolic interaction*, 21(2):129–153.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Gregory J Meyer, Stephen E Finn, Lorraine D Eyde, Gary G Kay, Kevin L Moreland, Robert R Dies, Elena J Eisman, Tom W Kubiszyn, and Geoffrey M Reed. 2001. Psychological testing and psychological assessment: A review of evidence and issues. *American psychologist*, 56(2):128.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Shigehiro Oishi, Jesse Graham, Selin Kesebir, and Iolanda Costa Galinha. 2013. Concepts of happiness across time and cultures. *Personality and Social Psychology Bulletin*, 39(5):559–577.

James W Pennebaker and Lori D Stone. 2003. Words of wisdom: language use over the life span. *Journal of personality and social psychology*, 85(2):291.

James W Pennebaker. 2011. The secret life of pronouns. *New Scientist*, 211(2828):42–45.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

H Andrew Schwartz, Johannes Eichstaedt, Margaret L Kern, Gregory Park, Maarten Sap, David Stillwell, Michal Kosinski, and Lyle Ungar. 2014. Towards assessing changes in degree of depression through facebook. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 118–125. Citeseer.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465.

Jean M Twenge, W Keith Campbell, and Brittany Gentile. 2012. Male and female pronoun use in us books reflects womenâĂŹs status, 1900–2008. *Sex roles*, 67(9-10):488–493.

Walter Weintraub. 1981. *Verbal behavior: Adaptation and psychopathology*. Springer Publishing Company.

Johannes Zimmermann, Markus Wolf, Astrid Bock, Doris Peham, and Cord Benecke. 2013. The way we refer to ourselves reflects how we relate to others: Associations between first-person pronoun use and interpersonal problems. *Journal of research in personality*, 47(3):218–225.

Learning to Identify Metaphors from a Corpus of Proverbs

Gözde Özbal[†] and **Carlo Strapparava**[†] and **Serra Sinem Tekiroğlu**[†] and **Daniele Pighin**[‡]
[†]FBK-Irst, Trento, Italy, [‡]Google - Zürich, Switzerland
gozbalde@gmail.com, {strappa, tekiroglu}@fbk.eu, biondo@google.com

Abstract

In this paper, we experiment with a resource consisting of metaphorically annotated proverbs on the task of word-level metaphor recognition. We observe that existing feature sets do not perform well on this data. We design a novel set of features to better capture the peculiar nature of proverbs and we demonstrate that these new features are significantly more effective on the metaphorically dense proverb data.

1 Introduction

Recent years have seen a growing attention towards attempts to understand figurative language in text (Steen et al., 2010, Shutova and Teufel, 2010, Turney et al., 2011, Neuman et al., 2013, Klebanov et al., 2015). Recently, Özbal et al. (2016) published a resource consisting of 1,054 proverbs annotated with metaphors at the word and sentence level, making it possible for the first time to test existing models for metaphor detection on such data. More than in other genres, such as news, fiction and essays, in proverbs metaphors can resolve a significant amount of the figurative meaning (Faycel, 2012). The richness of proverbs in terms of metaphors is very fascinating from a linguistic and cultural point of view. Due to this richness, proverbs constitute a challenging benchmark for existing computational models of metaphoricity.

In this paper, we devise novel feature sets especially tailored to cope with the peculiarities of proverbs, which are generally short and figuratively rich. To the best of our knowledge, this is the first attempt to design a word-level metaphor recognizer specifically tailored to such metaphorically rich data. Even though some of the resources that we use (e.g., imageability and concreteness) have been used for this task before, we propose new ways of encoding this information, especially with respect to the density of the feature space and the way that the context of each word is modeled. On the proverb data, the novel features result in compact models that significantly outperform existing features designed for word-level metaphor detection in other genres (Klebanov et al., 2014), such as news and essays. By also testing the new features on these other genres, we show that their generalization power is not limited to proverbs.

2 Background

In this section we provide a brief overview of the efforts of the NLP community to build metaphor datasets and utilize them to develop computational techniques for metaphor processing. Steen et al. (2010) construct the Amsterdam Metaphor Corpus (VUAMC) by annotating a subset of BNC Baby[1]. Linguistic metaphors in VUAMC are annotated by utilizing the Metaphor Annotation Procedure (MIP) proposed by Group (2007). VUAMC contains 200,000 words in sentences sampled from various genres (news, fiction, academic, and conversations) and 13.6% of the words are annotated as metaphoric (Shutova, 2010). Another metaphor annotation study following the MIP procedure is conducted by Shutova and Teufel (2010). A subset of

[1] http://www.natcorp.ox.ac.uk/corpus/babyinfo.html

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2060–2065,
Austin, Texas, November 1-5, 2016.

the British National Corpus (BNC) (Burnard, 2000) is annotated to reveal word-level verb metaphors and to determine the conceptual mappings of the metaphorical verbs.

Turney et al. (2011) introduce an algorithm to classify word-level metaphors expressed by an adjective or a verb based on their concreteness levels in association with the nouns they collocate. Similarly, Neuman et al. (2013) extend the concreteness model with a selectional preference approach to detect metaphors formed of concrete concepts. They focus on three types of metaphors: i) IS-A, ii) verb-noun, iii) adjective-noun. Rather than restricting the identification task to a particular POS or metaphoric structure, Hovy et al. (2013) aim to recognize any word-level metaphors given an unrestricted text, and they create a corpus containing sentences where one target token for each sentence is annotated as metaphorical or literal. They use SVM and CRF models with dependency tree-kernels to capture the anomalies in semantic patterns. Klebanov et al. (2014) propose a supervised approach to predict the metaphoricity of all content words in a running text. Their model combines unigram, topic model, POS and concreteness features and it is evaluated on VUAMC and a set of essays written for a large-scale assessment of college graduates. Following this study, Klebanov et al. (2015) improve their model by re-weighting the training examples and redesigning the concreteness features.

The experiments in this paper are carried out on PROMETHEUS (Özbal et al., 2016), a dataset consisting of 1,054 English proverbs and their equivalents in Italian. Proverbs are annotated with word-level metaphors, overall metaphoricity, meaning and century of first appearance. For our experiments, we only use the word-level annotations on the English data.

3 Word-level metaphor detection

Similarly to Klebanov et al. (2014), we classify each content word (i.e., adjective, noun, verb or adverb) appearing in a proverb as being used metaphorically or not. Out of 1,054 proverbs in PROMETHEUS, we randomly sample 800 for training, 127 for development and 127 for testing. We carry out the development of new features on the development set; then we compare the performance of different feature sets using 10-fold cross validation on the combination of the development and training data. Finally, we test the most meaningful configurations on the held-out test data. As a baseline, we use a set of features very similar to the one proposed by Klebanov et al. (2014). To obtain results more easily comparable with Klebanov et al. (2014), we use the same classifier, i.e., logistic regression, in the implementation bundled with the *scikit-learn* package (Pedregosa et al., 2011). For all the experiments, we adjust the weight of the examples proportionally to the inverse of the class frequency.

3.1 Baseline features (B)

Unigrams (u_B): Klebanov et al. (2014) use all content word forms as features without stemming or lemmatization. To reduce sparsity, we consider lemmas along with their POS tag.

Part-of-speech (p_B): The coarse-grained part-of-speech (i.e., noun, adjective, verb or adverb) of content words[2].

Concreteness (c_B): We extract the concreteness features from the resource compiled by Brysbaert et al. (2014). Similarly to Klebanov et al. (2014), the mean concreteness ratings, ranging from 1 to 5, are binned in 0.25 increments. We also add a binary feature which encodes the information about whether the lemma is found in the resource.

Topic models (t_B): We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) using Gibbs sampling for parameter estimation and inference (Griffiths, 2002). We run LDA on the full British National Corpus (Consortium and others, 2001) to estimate 100 topics, using 2000 Gibbs sampling iterations, and keeping the first 1000 words for each topic. As topic model features for a lemma, we use the conditional probability of the topic given the lemma for each of the 100 topics generated by LDA. Besides, we use a binary feature that encodes whether the lemma exists in the LDA model.

3.2 Novel features (N)

We introduce five feature sets that capture other aspects of the data which we consider to be meaningful for the peculiar characteristics of proverbs.

[2]Klebanov et al. (2014) consider the Penn Treebank tagset generated by Stanford POS tagger.

Imageability (i) and Concreteness (c): Imageability and concreteness of the metaphor constituents were found to be highly effective in metaphor identification by several studies in the literature (Turney et al., 2011, Broadwell et al., 2013, Neuman et al., 2013, Tsvetkov et al., 2014). We obtain the imageability and concreteness scores of each lemma from the resource constructed by Tsvetkov et al. (2014), as it accounts for both dimensions. The imageability (concreteness) feature set contains the following four features:

- ***Has score*:** A binary feature that indicates whether the lemma exists in the relevant resource.
- ***Score value*:** The imageability (concreteness) score of the lemma.
- ***Average sentence score*:** The average imageability (concreteness) score of the other lemmas in the sentence.
- ***Score difference*:** The difference between *Average sentence score* and *Score value*.

The last two features take the context of the target lemma into account and encode the intuition that metaphorical lemmas often have higher imageability (concreteness) than the rest of the sentence (Broadwell et al., 2013).

Metaphor counts (m): This feature set consists of three features. The first two features encode the number of times a lemma-POS pair is used as a metaphor and a non-metaphor in the data. The third feature evaluates to the difference between these counts[3].

Standard domains (d_s) and normalized domains (d_n): These features reflect our intuition that there is a strong prior for some domains to be used as a source for metaphors. This notion is backed by the analysis of PROMETHEUS carried out by Özbal et al. (2016). We also expect that words which are clearly out of context with respect to the rest of the sentence are more likely to be used as metaphors. The correlation between word and sentence domains described below aims to model such phenomenon. For each lemma-POS pair, we collect the domain information from WordNet Domains[4] (Magnini et al., 2002, Bentivogli et al., 2004) for the standard domains feature set, which consists of 167 features (1 real valued, 166 binary). It includes a binary indicator set to 1 if the lemma is found in WordNet Domains. A domain vector consisting of 164 binary indicators mark the domains to which the lemma belongs. Then, we compute a sentence domain vector by summing the vectors for all the other lemmas in the sentence, and we encode the Pearson correlation coefficient between the two vectors (lemma and sentence) as a real valued feature. Finally, a binary feature accounts for the cases in which no other lemma in the sentence has associated domain information.

The same process is repeated for the normalized domains. For normalization, we use a reduced set of domains (43 distinct domains) by considering the middle level of the WordNet Domains hierarchy. For instance, VOLLEY or BASKETBALL domains are mapped to the SPORT domain. Normalization already proved to be beneficial in tasks such as word sense disambiguation (Gliozzo et al., 2004). It allows for a good level of abstraction without losing relevant information and it helps to overcome data sparsity. The set of normalized domain features (d_n) consists of 46 features (45 binary, 1 real valued).

Dense signals (s): This set includes three binary features which summarize the concreteness, imageability and metaphor count feature sets. The first (second) feature is set to 1 if the imageability (concreteness) of the lemma is higher than the average

Feature sets	*C*	*P*	*R*	*F*
$B^{\#}$	0.9	0.666	0.832	0.738
N^{*}	0.6	0.785	0.884	0.833
$B \cup N^{*}$	0.6	0.798	0.875	0.834
$N \setminus i^{*}$	0.6	0.788	0.886	0.833
$N \setminus c^{*}$	0.6	0.782	0.888	0.831
$N \setminus m^{*\#}$	0.6	0.780	0.824	0.799
$N \setminus d_s^{*}$	1.0	0.787	0.842	0.815
$N \setminus d_n^{*}$	1.0	0.789	0.884	0.832
$N \setminus (d_s \cup d_n)^{\#}$	1.0	0.746	0.704	0.724
$N \setminus s^{*}$	1.0	0.776	0.909	0.836
$(N \setminus (d_s \cup d_n)) \cup t_B^{\#}$	0.6	0.751	0.705	0.724

Table 1: Cross-validation performance on the proverb training and development data. The meta-parameter C is the inverse of the regularization strength. *: significantly different from B with $p < .001$; $^{\#}$: s.d. from N with $p < .001$.

[3] Counts are estimated on training folds. To reduce overfitting, lemmas are randomly sampled with a probability of 2/3.

[4] We always select the first sense of the lemma-POS.

Features	*P*	*R*	*F*
$B^{\#}$	0.75	0.70	0.73
N^{*}	0.86	0.83	0.85
$N \setminus s^{*}$	0.82	0.87	0.85
$B \cup N^{*}$	0.87	0.85	0.86

Table 2: Performance on the proverb test data. *: significantly different from B with $p < .001$. $^{\#}$: significantly different from N with $p < .001$.

Genre	*Features*	*C*	*P*	*R*	*F*
News	B	1.0	0.475	0.742	0.576
	N	1.0	0.576	0.479	0.522
	$B \cup N$	1.0	0.615	0.539	0.574
Academic	B	0.6	0.489	0.733	0.568
	N	0.6	0.572	0.494	0.511
	$B \cup N$	1.0	0.539	0.648	0.569
Conversation	B	0.6	0.292	0.799	0.416
	N	0.6	0.304	0.626	0.393
	$B \cup N$	1.0	0.299	0.731	0.406
Fiction	B	0.6	0.349	0.695	0.460
	N	0.6	0.430	0.418	0.421
	$B \cup N$	0.6	0.409	0.551	0.465

Table 3: Cross-validation performance on VUAMC. B is always significantly different from N ($p < .001$), and $B \cup N$ is always significantly different from both B and N ($p < .001$).

imageability (concreteness) of the rest of the sentence. The third feature is set to 1 if the lemma was observed more frequently as a metaphor than not, as estimated on training data.

3.3 Results

Table 1 shows the results of the 10-fold cross validation on the English proverb data. The value reported in the column labeled C is the optimal inverse of regularization strength, determined via grid-search in the interval $[0.1, 1.0]$ with a step of 0.1. Using only baseline features (B) we measure an average F1 score of 0.738. The performance goes up to 0.833 when the novel features are used in isolation (N) (statistically significant with $p < 0.001$). We believe that the difference in performance is at least in part due to the sparser B features requiring more data to be able to generalize. But most importantly, unlike B, N accounts for the context and the peculiarity of the target word with respect to the rest of the sentence. The combination of the two feature sets ($B \cup N$) very slightly improves over N (0.834), but the difference is not significant. The second block of rows in Table 1 presents a summary of the ablation tests that we conducted to assess the contribution of the different feature groups. Each lowercase letter indicates one of the feature sets introduced in the previous section. All configurations reported, except $N \setminus (d_s \cup d_n)$, significantly outperform B. In two cases, $N \setminus m$ and $N \setminus (d_s \cup d_n)$, there is a significant loss of performance with respect to N. The worst performance is observed when all the domain features are removed (i.e., $N \setminus (d_s \cup d_n)$). These results suggest that the prior knowledge about the domain of a word and the frequency of its metaphorical use are indeed strong predictors of a word metaphoricity in context. The fact that $N \setminus d_n$ and $N \setminus d_s$ do not result in the same loss of performance as $N \setminus (d_s \cup d_n)$ indicates that both d_n and d_s are adequately expressive to model the figuratively rich proverb data. In one case (i.e., $N \setminus s$), the F1 measure is slightly higher than N, even though the difference does not appear to be statistically significant. Our intuition is that each of the three binary indicators is a very good predictor of metaphoricity *per se*, and due to the relatively small size of the data the classifier may tend to over-fit on these features. As another configuration, the last row shows the results obtained by replacing our domain features d_s and d_n with the topic features t from B. With this experiment, we aim to understand the extent to which the two features are interchangeable. The results are significantly worse than N, which is a further confirmation of the suitability of the domain features to model the proverbs dataset.

We then evaluated the best configuration from the cross-fold validation ($N \setminus s$) and the three feature sets B, N and $B \cup N$ on the held-out test data. The results of this experiment reported in Table 2 are similar to the cross-fold evaluation, and in this case the contribution of N features is even more accentuated. Indeed, the absolute F_1 of N and $B \cup N$ is slightly higher on test data, while the f-measure of B decreases slightly. This might be explained by the low-dimensionality of N, which makes it less prone to overfitting the training data. On test data, $N \setminus s$ is not found to outperform N. Interestingly, $N \setminus s$ is the only configuration having higher recall than precision. As shown by the feature ablation experi-

ments, one of the main reasons for the performance difference between N and B is the ability of the former to model domain information. This finding can be further confirmed by inspecting the cases where B misclassifies metaphors that are correctly detected by N. Among these, we can find several examples including words that belong to domains often used as a metaphor source, such as "grist" (domain: "gastronomy") in "All is grist that comes to the mill", or "horse" (domain: "animals") in "You can take a horse to the water , but you can't make him drink".

Finally, Table 3 shows the effect of the different feature sets on VUAMC used by Klebanov et al. (2014). We use the same 12-fold data split as Klebanov et al. (2014), and also in this case we perform a grid-search to optimize the meta-parameter C of the logistic regression classifier. The best value of C identified for each genre and feature set is shown in the column labeled C. On this data, N features alone are significantly outperformed by B ($p < 0.01$). On the other hand, for the genres "academic" and "fiction", combining N and B features improves classification performance over B, and the difference is always statistically significant. Besides, the addition of N always leads to more balanced models, by compensating for the relatively lower precision of B. Due to the lack of a separate test set, as in the original setup by Klebanov et al. (2014), and to the high dimensionality of B's lexicalized features, we cannot rule out over-fitting as an explanation for the relatively good performance of B on this benchmark. It should also be noted that the results reported in (Klebanov et al., 2014) are not the same, due to the mentioned differences in the implementation of the features and possibly other differences in the experimental setup (e.g., data filtering, pre-processing and meta-parameter optimization). In particular, our implementation of the B features performs better than reported by Klebanov et al. (2014) on all four genres, namely: 0.52 vs. 0.51 for "news", 0.51 vs. 0.28 for "academic", 0.39 vs. 0.28 for "conversation" and 0.42 vs. 0.33 for "fiction".

Even though the evidence is not conclusive, these results suggest that the insights derived from the analysis of PROMETHEUS and captured by the feature set N can also be applied to model word-level metaphor detection across very different genres. In particular, we believe that our initial attempt to encode context and domain information for metaphor detection deserves further investigation.

4 Conclusion

We designed a novel set of features inspired by the analysis of PROMETHEUS, and used it to train and test models for word-level metaphor detection. The comparison against a strong set of baseline features demonstrates the effectiveness of the novel features at capturing the metaphoricity of words for proverbs. In addition, the novel features show a positive contribution for metaphor detection on "fiction" and "academic" genres. The experimental results also highlight the peculiarities of PROMETHEUS, which stands out as an especially dense, metaphorically rich resource for the investigation of the linguistic and computational aspects of figurative language.

References

Luisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. 2004. Revising the Wordnet Domains Hierarchy: Semantics, Coverage and Balancing. In *Proceedings of the Workshop on Multilingual Linguistic Ressources*, pages 101–108. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022.

George Aaron Broadwell, Umit Boz, Ignacio Cases, Tomek Strzalkowski, Laurie Feldman, Sarah Taylor, Samira Shaikh, Ting Liu, Kit Cho, and Nick Webb. 2013. Using Imageability and Topic Chaining to Locate Metaphors in Linguistic Corpora. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 102–110. Springer.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness Ratings for 40 Thousand Generally Known English Word Lemmas. *Behavior Research Methods*, 46(3):904–911.

Lou Burnard. 2000. Reference guide for the British National Corpus (World Edition).

BNC Consortium et al. 2001. The British National Corpus, version 2 (BNC World). *Distributed by Oxford University Computing Services*.

Dahklaoui Faycel. 2012. *Food Metaphors in Tunisian Arabic Proverbs*. Rice Working Papers in Linguistics 3/1.

Alfio Gliozzo, Carlo Strapparava, and Ido Dagan. 2004. Unsupervised and Supervised Exploitation of Semantic Domains in Lexical Disambiguation. *Computer Speech & Language*, 18(3):275–299.

Tom Griffiths. 2002. Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation.

Pragglejaz Group. 2007. MIP: A Method for Identifying Metaphorically Used Words in Discourse. *Metaphor and Symbol*, 22(1):1–39.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying Metaphorical Word Use with Tree Kernels. *Meta4NLP 2013*, page 52.

Beata Beigman Klebanov, Chee Wee Leong, Michael Heilman, and Michael Flor. 2014. Different Texts, Same Metaphors: Unigrams and Beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 11–17.

Beata Beigman Klebanov, Chee Wee Leong, and Michael Flor. 2015. Supervised Word-Level Metaphor Detection: Experiments with Concreteness and Reweighting of Examples. *NAACL HLT 2015*, page 11.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2002. The Role of Domain Information in Word Sense Disambiguation. *Natural Language Engineering*, 8(04):359–373.

Yair Neuman, Dan Assaf, Yohai Cohen, Mark Last, Shlomo Argamon, Newton Howard, and Ophir Frieder. 2013. Metaphor Identification in Large Texts Corpora. *PloS one*, 8(4):e62343.

Gözde Özbal, Carlo Strapparava, and Serra Sinem Tekiroğlu. 2016. PROMETHEUS: A Corpus of Proverbs Annotated with Metaphors. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor Corpus Annotated for Source-Target Domain Mappings. In *LREC*, volume 2, pages 2–2.

Ekaterina Shutova. 2010. Automatic Metaphor Interpretation as a Paraphrasing Task. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1029–1037. Association for Computational Linguistics.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna A Kaal, and Tina Krennmayr. 2010. Metaphor in Usage. *Cognitive Linguistics*, 21(4):765–796.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor Detection with Cross-Lingual Model Transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 248–258. Association for Computational Linguistics.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and Metaphorical Sense Identification through Concrete and Abstract Context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, pages 680–690.

An Embedding Model for Predicting Roll-Call Votes

Peter E. Kraft Hirsh Jain Alexander M. Rush
School of Engineering and Applied Science, Harvard University
{pkraft, hirshjain}@college.harvard.edu, srush@seas.harvard.edu

Abstract

We develop a novel embedding-based model for predicting legislative roll-call votes from bill text. The model introduces multidimensional *ideal vectors* for legislators as an alternative to single dimensional ideal point models for quantitatively analyzing roll-call data. These vectors are learned to correspond with pre-trained word embeddings which allows us to analyze which features in a bill text are most predictive of political support. Our model is quite simple, while at the same time allowing us to successfully predict legislator votes on specific bills with higher accuracy than past methods.

1 Introduction

Quantitative analysis of political data can contribute to our understanding of governments. One important source of such data is *roll-call votes*, records of how legislators vote on bills. Analysis of roll-call data can reveal interesting information about legislators (such as political leanings and ideological clusters) and can also allow prediction of future votes (Clinton, 2012).

Previous work on analyzing roll-call votes has chiefly involved positioning congresspeople on ideal point models. Ideal point models assume all legislators and bills can be plotted as single points in one-dimensional "political space." The closer a particular bill's position is to a particular congressperson's, the more utility the congressperson is expected to derive from the bill. Initial work on ideal point models focused on using them to test theories about legislative behavior, such as predicting that the relative differences between ideal points of congresspeople of different parties, and thus party polarization, would increase over time (McCarty, 2001). Ideal point models are often created using Bayesian techniques over large amounts of roll-call data (Clinton et al., 2004; Jackman, 2001). However, these models are not used to make predictions. They are trained using the complete vote matrix for the bill, which indicates how each congressperson voted on each bill. Therefore, they cannot say anything about how congresspeople will vote on a new bill, as until some congresspeople have voted on the bill its ideal point is not known.

We target this vote prediction problem: given the text of a bill and a congressperson, can we independently predict how each congressperson will vote on the bill? The first prior attempt at this task was made by Gerrish and Blei (2011) who create an *ideal point topic model* which integrates a topic model similar to LDA for the bill text with an ideal point model for the congresspeople. They use variational inference to approximate the posterior distribution of the topics and ideal points, predicting with a linear model. Gerrish and Blei (2012) further extend this work with an *issue-adjusted model*, a similar model that modifies congressperson ideal points based on topics identified with labeled LDA, but which cannot be used for predictions. Further work in a similar vein includes Wang et al. (2013), who introduced temporal information to a graphical model for predicting Congressional votes, and Kim et al. (2014), who used sparse factor analysis to estimate Senatorial ideal points from bill text and the votes of party leadership.

In this work we revisit this task with a simple bilinear model that learns multidimensional embeddings for both legislators and bills, combining them

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2066–2070,
Austin, Texas, November 1-5, 2016.

to make vote predictions. We represent a bill as the average of its word embeddings. We represent legislators as *ideal vectors*, trained end-to-end for vote prediction. These ideal vectors serve as a useful, easy-to-train, multidimensional representation of legislator ideology that does not rely on elaborate statistical models or any further assumptions about legislator behavior. Finally, we train our model by optimizing a cross-entropy objective instead of the posterior of a topic model. The final model achieves high accuracy at predicting roll-call votes.

2 Model

Our goal is to predict roll-call votes by learning from the texts of bills and from past votes. Our input consists of a congressperson c and the set $\mathcal{B}$ of unique words in a bill. Our output y is whether that the congressperson voted `yea` or `nay` on the bill. We train on the full set of congressional votes on a number of bills. At test time, we supply entirely new bills and predict how each congressperson will vote on each new bill.

We propose a simple bilinear model that uses low-dimensional embeddings to model each word in our dictionary and each congressperson. We represent each bill using its word embeddings in order to capture the multivariate relationships between words and their meanings (Collobert et al., 2011; Mikolov et al., 2013). The model is trained to synthesize information about each congressperson's voting record into a multidimensional ideal vector. At test time, the model combines the embedding representation of a new bill with the trained ideal vector of a congressperson and generates a prediction for how the congressperson will vote on the bill.

Let $\mathbf{e}_w \in \mathbb{R}^{d_{word}}$ be the pretrained embedding for a word w. We initialize to the GloVe embeddings with $d_{word} = 50$ (Pennington et al., 2014), then jointly train them with the model. To represent a bill, we average over the embeddings of the set $\mathcal{B}$ of words in the bill.

To represent a congressperson, we introduce another set of embeddings $\mathbf{v}_c \in \mathbb{R}^{d_{emb}}$ for each congressperson c. The embeddings act as the ideal vector for each legislator. Unlike the word embeddings, we initialize these randomly.

The full model takes in a bill and a congressperson. It applies an affine transformation, represented by a matrix $\mathbf{W} \in \mathbb{R}^{d_{emb} \times d_{word}}$ and bias $\mathbf{b} \in \mathbb{R}^{d_{emb}}$, to map the bill representation into the space of the ideal vectors, and then uses a dot-product to provide a `yea`/`nay` score.

$$p(y = yea|\mathcal{B}, c) = \sigma((\mathbf{W}\left(\sum_{w \in \mathcal{B}} \mathbf{e}_w/|\mathcal{B}|\right) + \mathbf{b}) \cdot \mathbf{v}_c)$$

The full model is simply trained to minimize the negative log-likelihood of the training set, and requires no additional meta-information (such as party affiliation) or additional preprocessing of the bills during training- or test-time.

Congress	# Bills	House	Senate	Pres
106	557	R	R	Clinton
107	505	R	D[2]	Bush
108	607	R	R	Bush
109	579	R	R	Bush
110	854	D	D	Bush
111	965	D	D	Obama

Table 1: Dataset details for 106-111th Congress.

3 Experimental Setup

Data Following past work, our dataset is derived from the Govtrack database.[1] Specifically, our dataset consists of all votes on the full-text (not amendments) of bills or resolutions from the 106th-111th Congress, six of the most recent Congresses for which bill texts are readily available. Details of each these congresses are shown in Table 1.

To create our dataset, we first find a list of all votes on the full text of bills, and create a matrix of how each congressperson voted on each bill, which will be used in training and in testing. In accordance with previous work, we only consider yes-or-no votes and omit abstentions and "present" votes (Gerrish and Blei, 2011). We then simply collect the set of words used in each bill. Overall, our dataset consists of 4067 bills and over a million unique yes-or-no votes.

[1] https://www.govtrack.us/

[2] Mostly. Republicans controlled the 107th Senate for five months between the inauguration of Dick Cheney as vice-president in January of 2001 and the defection of Jim Jeffords in June.

Congress	YEA	GB	IDP	EMB
106	83.0	-	79.5	84.9
107	85.9	-	85.8	89.7
108	87.1	-	85.0	91.9
109	83.5	-	81.5	88.4
110	82.7	-	80.8	92.1
111	86.0	-	85.7	93.4
Avg	84.5	89.0	83.1	90.6

Table 2: Main results comparing predictive accuracy of our model EMB with a several baselines (described in the text) on the 106th-111th Congress.

Congress	EMB
106	0.524
107	0.546
108	0.595
109	0.628
110	0.728
111	0.737
Avg	0.645

Table 3: Minority class F1 Scores of our model EMB on the 106th-111th Congress.

Model We tested prediction accuracy of the average-of-embeddings model, EMB, by running it for ten epochs at a learning rate of $\eta = 0.1$ and d_{emb} set to 10. Hyperparameters were tuned on a held-out section of the 107th Congress. We ran on each of the 106th to 111th Congresses individually using five-fold cross-validation.

Baselines We compare our results to three different baselines. The first, YEA, is a majority class baseline which assumes all legislators vote `yea`. The second, IDP, is our model with d_{emb} set to 1 to simulate a simple ideal point model. The third, GB, is Gerrish and Blei's reported predictive accuracy of 89 % on average from the 106th to 111th Congresses, which is to the extent of our knowledge the best predictive accuracy on roll-call votes yet achieved in the literature. Gerrish and Blei report on the same data set using cross-validation and like us train and test on each congress individually, but do not split out results into individual congresses.

4 Experiments and Analysis

Predictive Results The main predictive experimental results are shown in Table 2. We see that EMB performs substantially better than YEA on all six Congresses. It has a weighted average of 90.6% on an 84.5% baseline, compared to Gerrish and Blei's 89% on an identical dataset. IDP, however, actually does worse than the baseline, demonstrating that the bulk of our gain in prediction accuracy comes from using ideal vectors instead of ideal points. To further test this hypothesis, we experimented with replacing word embeddings with LDA and obtained an accuracy of 89.5%, in between GB and EMB. This indicates that the word embeddings are also responsible for part, but not all, of the accuracy improvement. We also report minority class F1 scores for EMB in Table 3, finding an overall average F1 score of 0.645.

Ideal Vectors Beyond predictive accuracy, one of the most interesting features of the model is that it produces ideal vectors as its complete representation of congresspeople. These vectors are much easier to compute than standard ideal points, which require relatively complex and computationally intensive statistical models (Jackman, 2001). Additionally unlike ideal point models, which tend to contain many assumptions about legislative behavior, ideal vectors arise naturally from raw data and bill text (Clinton et al., 2004).

In Figure 1, we show the ideal vectors for the 111th Congress. We use PCA to project the vectors down to two dimensions. This graph displays several interesting patterns in agreement with theories of legislative behavior. For example, political scientists theorize that the majority party in a legislature will display more unity in roll-call votes because they decide what gets voted on and only allow a vote on a bill if they can unify behind it and pass it, while that bill may divide the other party (Carrubba et al., 2006; Carrubba et al., 2008). On this graph, in accordance with that prediction, the majority Democrats are more clustered than the minority Republicans. We observe similar trends in the ideal vectors of the other Congresses. Moreover, the model lets us examine the positions of individual congresspeople. In the figure, the 34 Democrats who

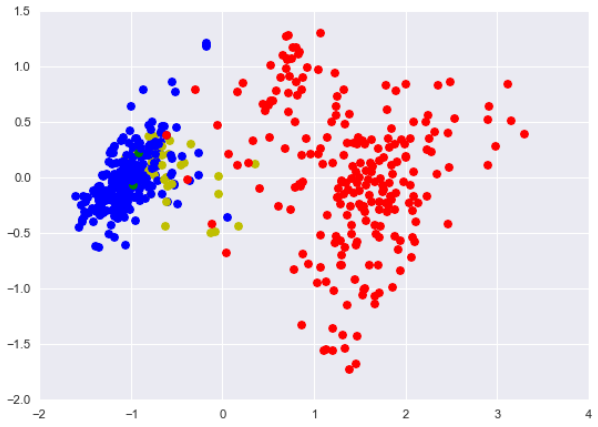


Figure 1: PCA projection of the ideal vectors for 111th Congress, both House and Senate. Republicans shown in red, Democrats who voted for Affordable Care Act (ACA) in blue, Democrats who voted against ACA in yellow, and independents in green.

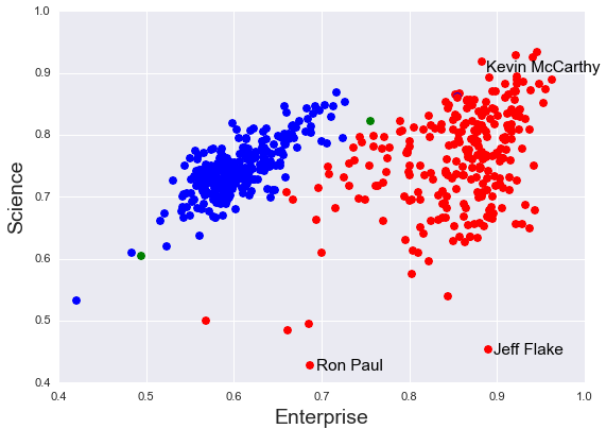


Figure 2: Relative likelihood of congresspeople in the 110th Congress voting for a bill containing only the word "Enterprise" versus only the word "Science." Coordinates are sigmoids of dot products of congressperson vectors with normalized word vectors.

voted against the Affordable Care Act (ACA, better known as Obamacare) are shown in yellow. The ACA was a major Democratic priority and point of difference between the two parties. The Democrats who voted against it tended to be relatively conservative and closer to the Republicans. The model picks up on this distinction.

Furthermore, since our model maps individual words and congresspeople to the same vector space, we can use it to determine how words (and by proxy issues) unite or divide congresspeople and parties. In Figure 2, we show the scaled probabilities that congresspeople will vote for a bill containing only the word "enterprise" versus one containing only the word "science" in the 110th Congress. The word "enterprise," denoting pro-business legislation, neatly divides the parties. Both are for it, but Republicans favor it more. More interestingly, the word "science" creates division *within* the parties, as neither was at the time more for science funding than the other but both contained congresspeople with varying levels of support for it. An ideal point model would likely capture the "enterprise" dimension, but not the "science" one, and would not be able to distinguish between libertarians like Ron Paul (R-TX) who are against both "corporate welfare" and government science funding, conservative budget hawks like Jeff Flake (R-AZ) who favor business but are skeptical of government funding of science, and establishment Republicans like Kevin McCarthy (R-CA) who support both. Indeed, ideal point models are known to perform poorly at describing ideologically idiosyncratic figures like Ron Paul (Gerrish and Blei, 2011). Providing the ability to explore multiple dimensions of difference between legislators will be extremely helpful for political scientists analyzing the dynamics of legislatures.

Lexical Properties Finally, as with topic modeling approaches, we can use our model to analyze the relationships between congresspeople or parties and individual words in bills. For example, Table 4 shows the ten words closest by cosine similarity to each party's average congressperson (stop words omitted) for the 110th Congress. The Democratic list mostly contains words relating to governing and regulating, such as "consumer," "state," and "educational," likely because the Democrats were at the time the majority party with the responsibility for passing large governmental and regulatory bills like budgets. The Republican list is largely concerned with the military, with words like "veterans," "service," and "executive," probably because of the importance at the time of the wars in Iraq and Afghanistan, started by a Republican president.

Democrats	Republicans
economic	veterans
exchange	head
state	opportunities
carrying	provided
government	promote
higher	service
congress	identified
consumer	information
educational	record
special	executive

Table 4: Top ten words by cosine similarity for each party in the 110th Congress with stop words removed.

5 Conclusion

We have developed a novel model for predicting Congressional roll-call votes. This new model provides a new and interesting way of analyzing the behavior of parties and legislatures. It achieves predictive accuracies around 90.6% on average and outperforms any prior model of roll-call voting. We also introduce the idea of ideal vectors as a fast, simple, and multidimensional alternative to ideal point models for analyzing the actions of individual legislators and testing theories about their behavior. Our code and datasets are available online at `https://github.com/kraftp/roll_call_predictor`.

References

Clifford J Carrubba, Matthew Gabel, Lacey Murrah, Ryan Clough, Elizabeth Montgomery, and Rebecca Schambach. 2006. Off the record: Unrecorded legislative votes, selection bias and roll-call vote analysis. *British Journal of Political Science*, 36(04):691–704.

Clifford Carrubba, Matthew Gabel, and Simon Hug. 2008. Legislative voting behavior, seen and unseen: A theory of roll-call vote selection. *Legislative Studies Quarterly*, 33(4):543–572.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. The statistical analysis of roll call data. *American Political Science Review*, 98(02):355–370.

Joshua D Clinton. 2012. Using roll call estimates to test models of politics. *Annual Review of Political Science*, 15:79–99.

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

Sean Gerrish and David M Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th international conference on machine learning (icml-11)*, pages 489–496.

Sean Gerrish and David M Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems*, pages 2753–2761.

Simon Jackman. 2001. Multidimensional analysis of roll call data via bayesian simulation: Identification, estimation, inference, and model checking. *Political Analysis*, 9(3):227–241.

In Song Kim, John Londregan, and Marc Ratkovic. 2014. Voting, speechmaking, and the dimensions of conflict in the us senate. In *Annual Meeting of the Midwest Political Science Association*.

Nolan McCarty. 2001. The hunt for party discipline in congress. In *American Political Science Association*, volume 95, pages 673–687. Cambridge Univ Press.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Eric Wang, Esther Salazar, David Dunson, Lawrence Carin, et al. 2013. Spatio-temporal modeling of legislation and votes. *Bayesian Analysis*, 8(1):233–268.

Natural Language Model Re-usability for Scaling to Different Domains

Young-Bum Kim, Alexandre Rochette and **Ruhi Sarikaya**
Microsoft Corporation, Redmond, WA
ybkim,alrochet,ruhi.sarikaya@microsoft.com

Abstract

Natural language understanding is the core of the human computer interactions. However, building new domains and tasks that need a separate set of models is a bottleneck for scaling to a large number of domains and experiences. In this paper, we propose a practical technique that addresses this issue in a web-scale language understanding system: Microsoft's personal digital assistant Cortana. The proposed technique uses a constrained decoding method with a universal slot tagging model sharing the same schema as the collection of slot taggers built for each domain. The proposed approach allows reusing of slots across different domains and tasks while achieving virtually the same performance as those slot taggers trained per domain fashion.

1 Introduction

Recently there has been tremendous investment into the personal digital assistants by big technology companies (Sarikaya, 2015; Sarikaya et al., 2016). Apple's SIRI, Google Now, Microsoft's Cortana and Amazon's Alexa are examples of such systems. Natural language understanding (Gupta et al., 2006; Tur and De Mori, 2011)is at the core of these systems providing natural communication between the user and the agent. These systems support a number of scenarios including creating reminders, setting up alarms, note taking, scheduling meetings, finding and consuming entertainment (i.e. movie, music, games), finding places of interest and getting driving directions to them. The domains supported by these systems are on the order of tens (not in hundreds) and adding new domains and experiences is a scaling problem that has not been solved yet (Tur, 2006; Jeong and Lee, 2009; El-Kahky et al., 2014; Kim et al., 2015d).

The primary reason behind this is that each domain requires potentially a new schema, intents and slots extracted from the natural language query. That, in turn requires collecting and annotating new data, which is the most expensive step in terms of time and money, and building a new set of domain specific models to support the new domains and scenarios. Slot modeling in particular is the most demanding component in terms of the difficulty of annotation and modeling.

In this study, we propose a new approach that reduces the cost of scaling natural language understanding to a large number of domains and experiences without significantly sacrificing the accuracy. The approach consists of a universal slot tagging method and a runtime technique called *constrained decoding* that performs the decoding according a specific schema. The proposed approach, heavily enables reusing existing slot modeling data that are collected and annotated for potentially different domains and applications, for building new domains and applications. The new domains can be expressed in terms of existing semantic schema.

The rest of the paper is organized as follows. In the next section, we talk about universal slot modeling. In section 3, we present the constrained decoding technique. We describe the experimental set up, results and analysis in section 4 followed by the conclusions and future directions in section 5.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2071–2076,
Austin, Texas, November 1-5, 2016.

2 Universal Slot Tagging

The first step is to build a universal slot tagger, a single model that can handle all the domains an agent (e.g. Cortana) can support. In Table 1, we show a subset of the domains that are supported by Cortana for experimentation purposes.

2.1 Universal Slot Tagger Training

To train the universal slot tagger, we consider two simple and intuitive approaches: *Binary* and *All-in-one*.

Suppose we have a combined k slots across domains and also have access to the labeled data, *Binary* approach trains k binary classifier one for each slot type. For each binary slot tagger targeting a specific slot type, the labeled data is programatically mapped to create a new labeled data set, where only the target label is kept while all the other labels are mapped "other" label. *All-in-one* approach simply trains a single model by aggregating queries across all domains.

2.2 Slot Tagging Ambiguity

Universal slot tagging model has an advantage, which can share schema across all domains used for training time. In spite of the advantage, there are ambiguity problems caused by combining all domains (and the underlying data) into a single model. The problems can be grouped into two categories:

- *Imbalanced training data distribution*: The amount of training data varies across domains. Universal slot model may have bias towards predicting the slots in domains with larger training data. For example, slots with less training data (e.g. *app_name* in *MEDIACONTROL* domain could be overwhelmed by slots with large training data (e.g. *place_name* in *PLACES* domain).

- *Domain-specific schema:* In practice, the domains the system handles are not constructed at the same time. They are designed for different application back-ends, requirements and scenarios at different points in time. In other words, the semantic schema for a domain is designed without considering other domains. In *ALARM* domain, the slot indicating time is subdivided into sub-slots such as *start_time* representing starting time, *duration* representing duration for an alarm. In contrast, in *PLACES* domain, there is only a single slot indicating time (*time*).

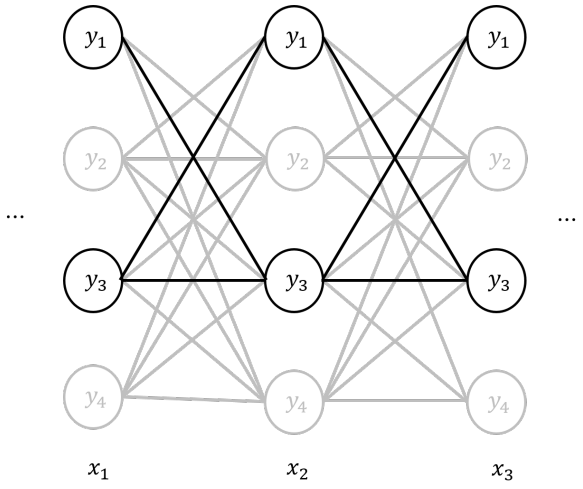


Figure 1: Constrained Lattice: Disabling nodes and transition while decoding the lattice to honor given constraints of domain schema.

3 Constrained Decoding

Slot tagging is considered as a sequence learning problem (Deoras and Sarikaya, 2013; Li et al., 2009; Xu and Sarikaya, 2014; Celikyilmaz et al., 2015; Kim et al., 2015b; Kim et al., 2015c; Kim et al., 2015a). In sequence learning, given a sample query $x_1 \dots x_n$, the decoding problem is to find the most likely slot sequence among all the possible sequences, $y_1 \dots y_n$:

$$f(x_1 \dots x_n) = \arg\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

Here, we assume that output space in training is same as those in test time.

However, in our problem, output (slot) space in test time can be different from those in training time. At test time, we may observe different slot sequences than what is observed in the training time.

This is not an issue for the *Binary* approach, since we can use the output of the selected taggers needed for the new domain. We simply use general decoding approach with each of the selected taggers. Note

that a given query is run as many times as the numbers of slot types covered in a given domain.

For *All-in-One* technique, we consider two possible approaches: *Post-Filter* and *Constrained Decoding*. With *Post-Filter*, we simply provide the best hypothesis generated by the slot tagger that meets the domain schema constraints, by computing the full n-best of slots and filtering out the slot types that do not meet the target domain schema. With *Constrained Decoding*, given a schema $\tilde{y} \subset y$ for the target domain, we first define a constrained lattice *lattice* $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}) \times \ldots \times \mathcal{Y}(x_n, \tilde{y})$, as shown in Figure 1. Then, we perform the decoding in the constrained lattice:

$$f(x_1 \ldots x_n, \tilde{y}) = \arg\max_{\mathcal{Y}(x,\tilde{y})} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

4 Experiments

In this section, we conducted a series of experiments to evaluate the proposed techniques on datasets obtained from real usage.

4.1 Experimental Setup

To test the effectiveness of the proposed approach, we apply it to a suite of 16 Cortana domains for slot tagging tasks. The data statistics and short descriptions are shown in Table 1. As the table indicates, the domains have different granularity and diverse semantics. Note that we keep domain-specific slots such as alarm_state, but there are enough shared labels across domains. For example, *ALARM* domain, there are 6 shared slots among 8 slots. There are 62 slots appearing more than one domain. Especially, some basic slots such as *time*, *date*,*place_name*,*person_name*,*location* and *product* appear in most domains.

4.2 Slot Taggers

In all our experiments, we trained Conditional Random Fields (CRFs)(Lafferty et al., 2001) and used n-gram features up to n = 3, regular expression, lexicon features, and Brown Clusters (Brown et al., 1992). With these features, we compare the following methods for slot tagging[1]:

- *In-domain*: Train a domain specific model using the domain specific data covering the slots supported in that domain.
- *Binary*: Train a binary classifier for each slot type, combine the slots needed for a given domain schema.
- *Post*: Train a single model with all domain data, take the one-best parse of the tagger and filter-out slots outside the domain schema.
- *Const*: Train a single model with all domain data and then perform constrained decoding using a domain specific schema.

4.3 Results

For the first scenario, we assume that test domain semantic schema is a subset of training domain schema. The results of this scenario are shown in Table 2. We consider *In-domain* as a plausible upper bound on the performance, yielding 94.16% of F1 on average. First, *Binary* has the lowest performance of 75.85%. We believe that when we train a binary classifier for each slot type, the other slots that provide valuable contextual information for the slot sequence are ignored. This leads to degradation in tagging accuracy. *Post* improves F1 scores across domains, resulting into 86.50% F1 on average. Note that this technique does not handle ambiguities and data distribution mismatch due to combining multiple domain specific data with different data sizes. Finally, *Const* lead to consistent gains across all domains, achieving 93.36%, which almost matches the *In-domain* performance. The reason why *Const* performs better than *Binary* is that *Const* constrains the best path search to the target domain schema. It does not consider the schema elements that are outside the target domain schema. By doing so, it addresses the training data distribution issue as well as overlap on various schema elements.

For the second scenario, we consider a new set of test domains not covered in the training set, as shown in Table 3. The amount of training data for the test domains are limited ($< 5K$). These domains lack training data for the *location* and *app_name* slots. When we use universal slot tagger with constrained decoding *Const* yields 94.30%. On average, *Const* increases F1-score by 1.41 percentage points,

[1]For parameter estimation, we used L-BFGS (Liu and Nocedal, 1989) with 100 as the maximum iteration count and 1.0 for the L2 regularization parameter.

	#slots	#shared slots	#train	#test	Description
ALARM	8	6	160K	16K	Set alarms
CALENDAR	21	17	100K	10K	Set meeting in calendar
COMM.	21	14	700K	70K	Make a call&send msg
MYSTUFF	20	16	24K	2.5K	find&open a document
ONDEVICE	10	8	227K	24k	Set up a phone
PLACES	31	22	478K	45K	Find location & info
REMIND	17	13	153K	14K	Remind to-do list
WEATHER	9	5	281K	26K	Ask weather
TRANSIT	16	16	0	2k	Ask bus schedule & info
MEDIACONT.	15	15	0	10k	Set up a music player
ENTERTAIN.	18	12	130k	13k	Find&play movie&music
ORDERFOOD	15	15	2.5k	2k	Order food
RESERVATIONS	21	19	3k	2k	Reserve restaurant
TAXI	17	17	0	2k	Book a cab
EVENTS	7	7	2k	1k	Book an event ticket
SHOWTIMES	15	15	2k	1k	Book a movie ticket

Table 1: The overview of data we used and descriptions.

Domain	In-domain	Binary	Post	Const
ALARM	**96.24**	76.49	91.86	95.33
CALENDAR	**91.79**	75.62	80.58	90.19
COMM.	**95.06**	84.17	88.19	94.76
ENTER.	**96.05**	85.39	90.42	95.84
MYSTUFF	**88.34**	51.3	80.6	87.51
ONDEVICE	**97.65**	70.16	77.8	96.43
PLACES	**92.39**	75.27	87.63	91.36
REMIND	**91.53**	72.67	88.98	91.1
WEATHER	**98.37**	91.56	92.45	97.73
Average	**94.16**	75.85	86.50	93.36

Table 2: Performance for universal models.

Domain	In-domain	Const
ORDERFOOD	93.62	**95.63**
RESERVATIONS	93.03	**94.58**
EVENTS	92.82	**94.28**
SHOWTIMES	92.07	**92.69**
Average	92.89	**94.30**

Table 3: Performance for prototype domains.

	TAXI	TRANSIT	MEDIAC.	AVG.
Const	90.86	99.5	93.08	94.48

Table 4: Results across new domains.

corresponding a 20% decrease in relative error. We believe that universal slot tagger learns to tag these slots from data available in *PLACES* and *ENTERTAINMENT* domains.

For the last scenario shown in Table 4, we assume that we do not have training data for the test domains. The *Const* performs reasonably well, yielding 94.48% on average. Interestingly, for the *TRANSIT* domain, we can get almost perfect tagging performance of 99.5%. Note that all tags in *TRANSIT* and *TAXI* domains are fully covered by our universal models, but the *MEDIACONTROL* domain is partially covered.

4.4 Discussion

By using the proposed technique, we maximize the reuse of existing data labeled for different domains and applications. The proposed technique allows mixing and matching of slots across different domains to create new domains. For example, we can tag the slots in the *SHOWTIMES* domain, which involves finding a movie to watch by using *movie_titles, actor_names* from the *ENTERTAINMENT* domain, and the location of the theater by using *location, place_name* slots from the *PLACES* domain. If the new domain needs some new slots that are not covered by the universal tagger, then some examples queries could be annotated and added to the universal slot tagger training data to retrain the models. Instead of maintaining a separate

slot tagger for each domain, one needs to maintain a single slot tagger. The new slots added can be used by future domains and applications.

5 Conclusions

We proposed a solution for scaling domains and experiences potentially to a large number of use cases by reusing existing data labeled for different domains and applications. The universal slot tagging coupled with constrained decoding achieves almost as good a performance as those slot taggers built in a domain specific fashion. This approach enables creation of virtual domains through any combination of slot types covered in the universal slot tagger schema, reducing the need to collect and annotate the same slot types multiple times for different domains. One of the future directions of research is to extend the same idea to the intent modeling, where we can re-use intent data built for different applications and domains for a new domain. Also, we plan to extend the constrained decoding idea to slot tagging with neural networks (Kim et al., 2016), which achieved gains over CRFs.

References

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Asli Celikyilmaz, Dilek Hakkani-Tur, Panupong Pasupat, and Ruhi Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. AAAI - Association for the Advancement of Artificial Intelligence, January.

Anoop Deoras and Ruhi Sarikaya. 2013. Deep belief network markov model sequence classification spoken language understanding. In *Interspeech*.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4067–4071. IEEE.

Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. 2006. The at&t spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):213–222.

Minwoo Jeong and Gary Geunbae Lee. 2009. Multi-domain spoken language understanding with transfer learning. *Speech Communication*, 51(5):412–424.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 84–92. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proceedings of Association for Computational Linguistics (ACL)*, pages 806–811. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 192–198. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. In *ACL*. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Minjoon Seo, and Ruhi Sarikaya. 2016. Domainless adaptation by constrained decoding on a schema lattice. In *Proceedings of the International Conference on Computational Linguistics (Coling)*. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Ruhi Sarikaya, Paul Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiuahu Liu, Daniel Boies, Tasos Anastasakos, Zhalleh Feizollahi, Nikhil Ramesh, Hisami Suzuki, Roman Holenstein, Elizabeth Krawczyk, and Vasiliy Radoste. 2016. An overview of end-to-end language understanding and dialog management for personal digital

assistants. In *IEEE Workshop on Spoken Language Technology*.

Ruhi Sarikaya. 2015. *The technology powering personal digital assistants*. Keynote at Interspeech, Dresden, Germany.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*, Toulouse, France.

Puyang Xu and Ruhi Sarikaya. 2014. Targeted feature dropout for robust slot filling in natural language understanding. In *ISCA - International Speech Communication Association*, September.

Leveraging Sentence-level Information with Encoder LSTM for Semantic Slot Filling

Gakuto Kurata
IBM Research
gakuto@jp.ibm.com

Bing Xiang
IBM Watson
bingxia@us.ibm.com

Bowen Zhou
IBM Watson
zhou@us.ibm.com

Mo Yu
IBM Watson
yum@us.ibm.com

Abstract

Recurrent Neural Network (RNN) and one of its specific architectures, Long Short-Term Memory (LSTM), have been widely used for sequence labeling. Explicitly modeling output label dependencies on top of RNN/LSTM is a widely-studied and effective extension. We propose another extension to incorporate the global information spanning over the whole input sequence. The proposed method, *encoder-labeler LSTM*, first encodes the whole input sequence into a fixed length vector with the encoder LSTM, and then uses this encoded vector as the initial state of another LSTM for sequence labeling. With this method, we can predict the label sequence while taking the whole input sequence information into consideration. In the experiments of a slot filling task, which is an essential component of natural language understanding, with using the standard ATIS corpus, we achieved the state-of-the-art F_1-score of 95.66%.

1 Introduction

Natural language understanding (NLU) is an essential component of natural human computer interaction and typically consists of identifying the intent of the users (intent classification) and extracting the associated semantic slots (slot filling) (De Mori et al., 2008). We focus on the latter semantic slot filling task in this paper.

Slot filling can be framed as a sequential labeling problem in which the most probable semantic slot labels are estimated for each word of the given word sequence. Slot filling is a traditional task and tremendous efforts have been done, especially since the 1980s when the Defense Advanced Research Program Agency (DARPA) Airline Travel Information System (ATIS) projects started (Price, 1990). Following the success of deep learning (Hinton et al., 2006; Bengio, 2009), Recurrent Neural Network (RNN) (Elman, 1990; Jordan, 1997) and one of its specific architectures, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), have been widely used since they can capture temporal dependencies (Yao et al., 2013; Yao et al., 2014a; Mesnil et al., 2015). The RNN/LSTM-based slot filling has been extended to be combined with explicit modeling of label dependencies (Yao et al., 2014b; Liu and Lane, 2015).

In this paper, we extend the LSTM-based slot filling to consider sentence-level information. In the field of machine translation, an encoder-decoder LSTM has been gaining attention (Sutskever et al., 2014), where the encoder LSTM encodes the global information spanning over the whole input sentence in its last hidden state. Inspired by this idea, we propose an *encoder-labeler LSTM* that leverages the encoder LSTM for slot filling. First, we encode the input sentence into a fixed length vector by the encoder LSTM. Then, we predict the slot label sequence by the labeler LSTM whose hidden state is initialized with the encoded vector by the encoder LSTM. With this encoder-labeler LSTM, we can predict the label sequence while taking the sentence-level information into consideration.

The main contributions of this paper are twofolds:

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2077–2083,
Austin, Texas, November 1-5, 2016.

1. Proposed an encoder-labeler LSTM to leverage sentence-level information for slot filling.

2. Achieved the state-of-the-art F_1-score of 95.66% in the slot filling task of the standard ATIS corpus.

2 Proposed Method

We first revisit the LSTM for slot filling and enhance this to explicitly model label dependencies. Then we explain the proposed encoder-labeler LSTM.

2.1 LSTM for Slot Filling

Figure 1(a) shows a typical LSTM for slot filling and we call this as *labeler LSTM(W)* where words are fed to the LSTM (Yao et al., 2014a).

Slot filling is a sequential labeling task to map a sequence of T words x_1^T to a sequence of T slot labels y_1^T. Each word x_t is represented with a V dimensional one-hot-vector where V is the vocabulary size and is transferred to d_e dimensional continuous space by the word embedding matrix $E \in \mathbb{R}^{d_e \times V}$ as Ex_t. Instead of simply feeding Ex_t into the LSTM, *Context Window* is a widely used technique to jointly consider k preceding and succeeding words as $Ex_{t-k}^{t+k} \in \mathbb{R}^{d_e(2k+1)}$. The LSTM has the architecture based on Jozefowicz et al. (2015) that does not have peephole connections and yields the hidden state sequence h_1^T. For each time step t, the posterior probabilities for each slot label are calculated by the softmax layer over the hidden state h_t. The word embedding matrix E, LSTM parameters, and softmax layer parameters are estimated to minimize the negative log likelihood over the correct label sequences with Back-Propagation Through Time (BPTT) (Williams and Peng, 1990).

2.2 Explicit Modeling of Label Dependency

A shortcoming of the labeler LSTM(W) is that it does not consider label dependencies. To explicitly model label dependencies, we introduce a new architecture, *labeler LSTM (W+L)*, as shown in Figure 1(b), where the output label of previous time step is fed to the hidden state of current time step, jointly with words, as Mesnil et al. (2015) and Liu and Lane (2015) tried with RNN. For model training, one-hot-vector of ground truth label of previous time step is fed to the hidden state of current time step and for evaluation, left-to-right beam search is used.

2.3 Encoder-labeler LSTM for Slot Filling

We propose two types of the encoder-labeler LSTM that uses the labeler LSTM(W) and the labeler LSTM(W+L). Figure 1(d) shows the *encoder-labeler LSTM(W)*. The encoder LSTM, to the left of the dotted line, reads through the input sentence backward. Its last hidden state contains the encoded information of the input sentence. The labeler LSTM(W), to the right of the dotted line, is the same with the labeler LSTM(W) explained in Section 2.1, *except that its hidden state is initialized with the last hidden state of the encoder LSTM*. The labeler LSTM(W) predicts the slot label conditioned on the encoded information by the encoder LSTM, which means that slot filling is conducted with taking sentence-level information into consideration. Figure 1(e) shows the *encoder-labeler LSTM(W+L)*, which uses the labeler LSTM(W+L) and predicts the slot label considering sentence-level information and label dependencies jointly.

Model training is basically the same as with the baseline labeler LSTM(W), as shown in Section 2.1, except that the error in the labeler LSTM is propagated to the encoder LSTM with BPTT.

This encoder-labeler LSTM is motivated by the encoder-decoder LSTM that has been applied to machine translation (Sutskever et al., 2014), grapheme-to-phoneme conversion (Yao and Zweig, 2015), text summarization (Nallapati et al., 2016) and so on. The difference is that the proposed encoder-labeler LSTM accepts the same input sequence twice while the usual encoder-decoder LSTM accepts the input sequence once in the encoder. Note that the LSTMs for encoding and labeling are different in the encoder-labeler LSTM, but the same word embedding matrix is used both for the encoder and labeler since the same input sequence is fed twice.

2.4 Related Work on Considering Sentence-level Information

Bi-directional RNN/LSTM have been proposed to capture sentence-level information (Mesnil et al., 2015; Zhou and Xu, 2015; Vu et al., 2016). While the bi-directional RNN/LSTM model the preceding and succeeding contexts at a specific word and

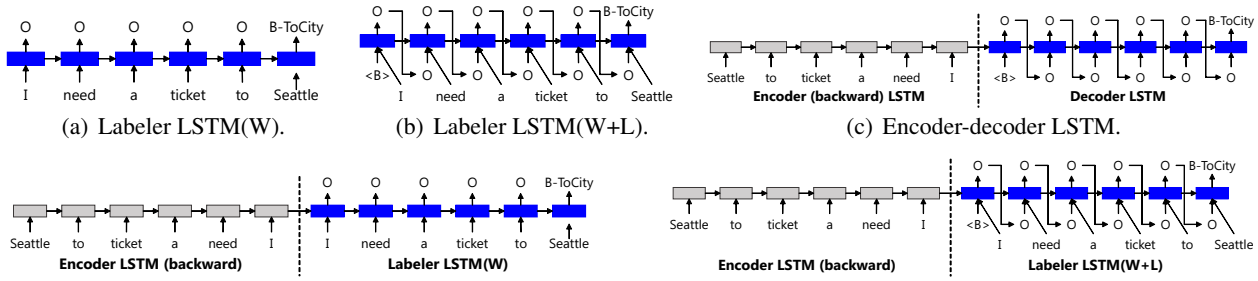


Figure 1: Neural network architectures for slot filling. Input sentence is "I need a ticket to Seattle". "B-ToCity" is slot label for specific meaning and "O"is slot label without specific meaning. "<B>" is beginning symbol for slot sequence.

Sentence	show	flights	from	Boston	to	New	York	today
Slots	O	O	O	B-FromCity	O	B-ToCity	I-ToCity	B-Date

Figure 2: Example of ATIS sentence and annotated slots.

don't explicitly encode the whole sentence, our proposed encoder-labeler LSTM explicitly encodes whole sentence and predicts slots conditioned on the encoded information.

Another method to consider the sentence-level information for slot filling is the attention-based approach (Simonnet et al., 2015). The attention-based approach is novel in aligning two sequences of different length. However, in the slot filling task where the input and output sequences have the same length and the input word and the output label has strong relations, the effect of introducing "soft" attention might become smaller. Instead, we directly fed the input word into the labeler part with using context window method as explained in Section 2.3.

3 Experiments

We report two sets of experiments. First we use the standard ATIS corpus to confirm the improvement by the proposed encoder-labeler LSTM and compare our results with the published results while discussing the related works. Then we use a large-scale data set to confirm the effect of the proposed method in a realistic use-case.

3.1 ATIS Experiment

3.1.1 Experimental Setup

We used the ATIS corpus, which has been widely used as the benchmark for NLU (Price, 1990; Dahl et al., 1994; Wang et al., 2006; Tur et al., 2010). Figure 2 shows an example sentence and its semantic slot labels in In-Out-Begin (IOB) representation. The slot filling task was to predict the slot label sequences from input word sequences.

The performance was measured by the F_1-score: $F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$, where precision is the ratio of the correct labels in the system's output and recall is the ratio of the correct labels in the ground truth of the evaluation data (van Rijsbergen, 1979).

The ATIS corpus contains the training data of 4,978 sentences and evaluation data of 893 sentences. The unique number of slot labels is 127 and the vocabulary size is 572. In the following experiments, we randomly selected 80% of the original training data to train the model and used the remaining 20% as the heldout data (Mesnil et al., 2015). We reported the F_1-score on the evaluation data with hyper-parameters that achieved the best F_1-score on the heldout data.

For training, we randomly initialized parameters in accordance with the normalized initialization (Glorot and Bengio, 2010). We used *ADAM* for learning rate control (Kingma and Ba, 2014) and *dropout* for generalization with a dropout rate of 0.5 (Srivastava et al., 2014; Zaremba et al., 2014).

3.1.2 Improvement by Encoder-labeler LSTM

We conducted experiments to compare the labeler LSTM(W) (Section 2.1), the labeler LSTM(W+L) (Section 2.2), and the encoder-labeler LSTM (Section 2.3). As for yet another baseline, we tried the encoder-decoder LSTM as shown in Figure 1(c)[1].

For all architectures, we set the initial learning rate to 0.001 (Kingma and Ba, 2014) and

[1]Length of the output label sequence is equal to that of the input word sequence in a slot filling task. Therefore, ending symbol for slot sequence is not necessary.

the dimension of word embeddings to $d_e = 30$. We changed the number of hidden units in the LSTM, $d_h \in \{100, 200, 300\}$[2], and the size of the context window, $k \in \{0, 1, 2\}$[3]. We used backward encoding for the encoder-decoder LSTM and the encoder-labeler LSTM as suggested in Sutskever et al. (2014). For the encoder-decoder LSTM, labeler LSTM(W+L), and encoder-labeler LSTM(W+L), we used the left-to-right beam search decoder (Sutskever et al., 2014) with beam sizes of 1, 2, 4, and 8 for evaluation where the best F_1-score was reported. During 100 training epochs, we reported the F_1-score on the evaluation data with the epoch when the F_1-score for the heldout data was maximized. Table 1 shows the results.

The proposed encoder-labeler LSTM(W) and encoder-labeler LSTM(W+L) both outperformed the labeler LSTM(W) and labeler LSTM(W+L), which confirms the novelty of considering sentence-level information with the encoder LSTM by our proposed method.

Contrary to expectations, F_1-score by the encoder-labeler LSTM(W+L) was not improved from that by the encoder-labeler LSTM(W). A possible reason for this is the propagation of label prediction errors. We compared the label prediction accuracy for the words after the first label prediction error in the evaluation sentences and confirmed that the accuracy deteriorated from 84.0% to 82.6% by using pthe label dependencies.

For the encoder-labeler LSTM(W) which was better than the encoder-labeler LSTM(W+L), we tried the deep architecture of 2 LSTM layers (*Encoder-labeler deep LSTM(W)*). We also trained the corresponding *labeler deep LSTM(W)*. As in Table 1, we obtained improvement from 94.91% to 95.47% by the proposed encoder-labeler deep LSTM(W), which was statistically significant at the 90% level.

Lastly, F_1-score by the encoder-decoder LSTM was worse than other methods as shown in the first row of Table 1. Since the slot label is closely related with the input word, the encoder-decoder LSTM was not an appropriate approach for the slot filling task.

[2]When using deep architecture later in this section, d_h was tuned for each layer.

[3]In our preliminary experiments with using the labeler LSTM(W), F_1-scores deteriorated with $k \geq 3$.

		F_1-score
(c)	Encoder-decoder LSTM	80.11
(a)	Labeler LSTM(W)	94.80
(d)	**Encoder-labeler LSTM(W)**	**95.29**
(b)	Labeler LSTM(W+L)	94.91
(e)	**Encoder-labeler LSTM(W+L)**	**95.19**
	Labeler Deep LSTM(W)	94.91
	Encoder-labeler Deep LSTM(W)	**95.47**

Table 1: Experimental results on ATIS slot filling task. Leftmost column corresponds to Figure 1. Lines with bold fonts use proposed encoder-labeler LSTM. [%]

3.1.3 Comparison with Published Results

Table 2 summarizes the recently published results on the ATIS slot filling task and compares them with the results from the proposed methods.

Recent research has been focusing on RNN and its extensions. Yao et al. (2013) used RNN and outperformed methods that did not use neural networks, such as SVM (Raymond and Riccardi, 2007) and CRF (Deng et al., 2012). Mesnil et al. (2015) tried bi-directional RNN, but reported degradation comparing with their single-directional RNN (94.98%). Yao et al. (2014a) introduced LSTM and deep LSTM and obtained improvement over RNN. Peng and Yao (2015) proposed RNN-EM that used an external memory architecture to improve the memory capability of RNN.

Many studies have been also conducted to explicitly model label dependencies. Xu and Sarikaya (2013) proposed CNN-CRF that explicitly models the dependencies of the output from CNN. Mesnil et al. (2015) used hybrid RNN that combined Elman-type and Jordan-type RNNs. Liu and Lane (2015) used the output label for the previous word to model label dependencies (RNN-SOP).

Vu et al. (2016) recently proposed to use ranking loss function over bi-directional RNNs with achieving 95.47% (R-biRNN) and reported 95.56% by ensemble (5×R-biRNN).

By comparing with these methods, the main difference of our proposed encoder-labeler LSTM is the use of encoder LSTM to leverage sentence-level information [4].

[4]Since Simonnet et al. (2015) did not report the experimental results on ATIS, we could not experimentally compare our result with their attention-based approach. Theoretical comparison is available in Section 2.4.

	F_1-score
RNN (Yao et al., 2013)	94.11
CNN-CRF (Xu and Sarikaya, 2013)	94.35
Bi-directional RNN (Mesnil et al., 2015)	94.73
LSTM (Yao et al., 2014a)	94.85
RNN-SOP (Liu and Lane, 2015)	94.89
Hybrid RNN (Mesnil et al., 2015)	95.06
Deep LSTM (Yao et al., 2014a)	95.08
RNN-EM (Peng and Yao, 2015)	95.25
R-biRNN (Vu et al., 2016)	95.47
5×R-biRNN (Vu et al., 2016)	95.56
Encoder-labeler LSTM(W)	**95.40**
Encoder-labeler Deep LSTM(W)	**95.66**

Table 2: Comparison with published results on ATIS slot filling task. F_1-scores by proposed method are improved from Table 1 due to sophisticated hyper-parameters. [%]

For our encoder-labeler LSTM(W) and encoder-labeler deep LSTM(W), we further conducted hyper-parameter search with a random search strategy (Bergstra and Bengio, 2012). We tuned the dimension of word embeddings, $d_e \in \{30, 50, 75\}$, number of hidden states in each layer, $d_h \in \{100, 150, 200, 250, 300\}$, size of context window, $k \in \{0, 1, 2\}$, and initial learning rate sampled from uniform distribution in range $[0.0001, 0.01]$. To the best of our knowledge, the previously published best F_1-score was 95.56%[5] (Vu et al., 2016). Our encoder-labeler deep LSTM(W) achieved 95.66% F_1-score, outperforming the previously published F_1-score as shown in Table 2.

Note some of the previous results used whole training data for model training while others used randomly selected 80% of data for model training and the remaining 20% for hyper-parameter tuning. Our results are based on the latter setup.

3.2 Large-scale Experiment

We prepared a large-scale data set by merging the MIT Restaurant Corpus and MIT Movie Corpus (Liu et al., 2013a; Liu et al., 2013b; Spoken Laungage Systems Group, 2013) with the ATIS corpus. Since users of the NLU system may provide queries without explicitly specifying their domain, building one NLU model for multiple domains is necessary. The merged data set contains 30,229 training and 6,810 evaluation sentences. The unique number of slot labels is 191 and the vocabulary size is 16,049. With this merged data set, we compared the labeler LSTM(W) and the proposed encoder-labeler LSTM(W) according to the experimental procedure explained in Section 3.1.2. The labeler LSTM(W) achieved the F_1-score of 72.80% and the encoder-labeler LSTM(W) improved it to 74.41%, which confirmed the effect of the proposed method in large and realistic data set [6].

4 Conclusion

We proposed an encoder-labeler LSTM that can conduct slot filling conditioned on the encoded sentence-level information. We applied this method to the standard ATIS corpus and obtained the state-of-the-art F_1-score in a slot filling task. We also tried to explicitly model label dependencies, but it was not beneficial in our experiments, which should be further investigated in our future work.

In this paper, we focused on the slot labeling in this paper. Previous papers reported that jointly training the models for slot filling and intent classification boosted the accuracy of both tasks (Xu and Sarikaya, 2013; Shi et al., 2015; Liu et al., 2015). Leveraging our encoder-labeler LSTM approach in joint training should be worth trying.

Acknowledgments

We are grateful to Dr. Yuta Tsuboi, Dr. Ryuki Tachibana, and Mr. Nobuyasu Itoh of IBM Research - Tokyo for the fruitful discussion and their comments on this and earlier versions of the paper. We thank Dr. Ramesh M. Nallapati and Dr. Cicero Nogueira dos Santos of IBM Watson for their valuable suggestions. We thank the anonymous reviewers for their valuable comments.

[5] There are other published results that achieved better F_1-scores by using other information on top of word features. Vukotic et al. (2015) achieved 96.16% F_1-score by using the named entity (NE) database when estimating word embeddings. Yao et al. (2013) and Yao et al. (2014a) used NE features in addition to word features and obtained improvement with both the RNN and LSTM upto 96.60% F_1-score. Mesnil et al. (2015) also used NE features and reported F_1-score of 96.29% with RNN and 96.46% with Recurrent CRF.

[6] The purpose of this experiment is to confirm the effect of the proposed method. The absolute F_1-scores can not be compared with the numbers in Liu et al. (2013b) since the capitalization policy and the data size of the training data were different.

References

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proc. HLT*, pages 43–48.

Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding. *IEEE Signal Processing Magazine*, 3(25):50–58.

Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tur. 2012. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *Proc. SLT*, pages 210–215.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

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

Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. ICML*, pages 2342–2350.

Diederik Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and James Glass. 2013a. Asgard: A portable architecture for multilingual dialogue systems. In *Proc. ICASSP*, pages 8386–8390.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and James Glass. 2013b. Query understanding enhanced by hierarchical parsing structures. In *Proc. ASRU*, pages 72–77.

Chunxi Liu, Puyang Xu, and Ruhi Sarikaya. 2015. Deep contextual language understanding in spoken dialogue systems. In *Proc. INTERSPEECH*, pages 120–124.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Ramesh Nallapati, Bowen Zhou, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. CoNLL*.

Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

Patti Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proc. DARPA Speech and Natural Language Workshop*, pages 91–95.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proc. INTERSPEECH*, pages 1605–1608.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. Contextual spoken language understanding using recurrent neural networks. In *Proc. ICASSP*, pages 5271–5275.

Edwin Simonnet, Camelin Nathalie, Deléglise Paul, and Estève Yannick. 2015. Exploring the use of attention-based recurrent neural networks for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Spoken Laungage Systems Group. 2013. The MIT Restaurant Corpus and The MIT Movie Corpus. `https://groups.csail.mit.edu/sls/downloads/`, MIT Computer Science and Artificial Intelligence Laboratory.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112.

Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2010. What is left to be understood in ATIS? In *Proc. SLT*, pages 19–24.

Cornelis Joost van Rijsbergen. 1979. *Information Retrieval*. Butterworth.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Proc. ICASSP*, pages 6060–6064.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2015. Is it time to switch to word embedding and recurrent neural networks for spoken language understanding? In *Proc. INTERSPEECH*, pages 130–134.

Ye-Yi Wang, Alex Acero, Milind Mahajan, and John Lee. 2006. Combining statistical and knowledge-based spoken language understanding in conditional models. In *Proc. COLING-ACL*, pages 882–889.

Ronald J Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *Proc. ASRU*, pages 78–83.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *Proc. INTERSPEECH*, pages 3330–3334.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Proc. INTERSPEECH*, pages 2524–2528.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014a. Spoken language understanding using long short-term memory neural networks. In *Proc. SLT*, pages 189–194.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014b. Recurrent conditional random field for language understanding. In *Proc. ICASSP*, pages 4077–4081.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. ACL*, pages 1127–1137.

AMR-to-text generation as a Traveling Salesman Problem

Linfeng Song[1], **Yue Zhang**[3], **Xiaochang Peng**[1], **Zhiguo Wang**[2] and **Daniel Gildea**[1]
[1]Department of Computer Science, University of Rochester, Rochester, NY 14627
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
[3]Singapore University of Technology and Design

Abstract

The task of AMR-to-text generation is to generate grammatical text that sustains the semantic meaning for a given AMR graph. We attack the task by first partitioning the AMR graph into smaller fragments, and then generating the translation for each fragment, before finally deciding the order by solving an asymmetric generalized traveling salesman problem (AGTSP). A Maximum Entropy classifier is trained to estimate the traveling costs, and a TSP solver is used to find the optimized solution. The final model reports a BLEU score of 22.44 on the SemEval-2016 Task8 dataset.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism encoding the meaning of a sentence as a rooted, directed graph. Shown in Figure 1, the nodes of an AMR graph (e.g. "boy", "go-01" and "want-01") represent concepts, and the edges (e.g. "ARG0" and "ARG1") represent relations between concepts. AMR jointly encodes a set of different semantic phenomena, which makes it useful in applications like question answering and semantics-based machine translation. AMR has served as an intermediate representation for various text-to-text NLP applications, such as statistical machine translation (SMT) (Jones et al., 2012).

The task of AMR-to-text generation is to generate grammatical text containing the same semantic meaning as a given AMR graph. This task is important yet also challenging since each AMR graph usually has multiple corresponding sentences, and syntactic structure and function words are abstracted away when transforming a sentence into AMR (Banarescu et al., 2013). There has been work dealing with text-to-AMR parsing (Flanigan et al., 2014; Wang et al., 2015; Peng et al., 2015; Vanderwende et al., 2015; Pust et al., 2015; Artzi et al., 2015). On the other hand, relatively little work has been done on AMR-to-text generation. One recent exception is Flanigan et al. (2016), who first generate a spanning tree for the input AMR graph, and then apply a tree transducer to generate the sentence. Here, we directly generate the sentence from an input AMR by treating AMR-to-text generation as a variant of the traveling salesman problem (TSP).

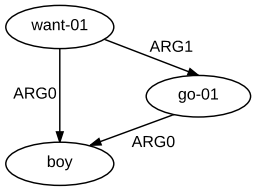


Figure 1: AMR graph for "The boy wants to go".

Given an AMR as input, our method first cuts the graph into several rooted and connected fragments (sub-graphs), and then finds the translation for each fragment, before finally generating the sentence for the whole AMR by ordering the translations. To cut the AMR and translate each fragment, we match the input AMR with rules, each consisting of a rooted, connected AMR fragment and a corresponding translation. These rules serve in a similar way to rules in SMT models. We learn the rules by a modified version of the sampling algorithm of Peng

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2084–2089,
Austin, Texas, November 1-5, 2016.

et al. (2015), and use the rule matching algorithm of Cai and Knight (2013).

For decoding the fragments and synthesizing the output, we define a *cut* to be a subset of matched rules without overlap that covers the AMR, and an *ordered cut* to be a cut with the rules being ordered. To generate a sentence for the whole AMR, we search for an ordered cut, and concatenate translations of all rules in the cut. TSP is used to traverse different cuts and determine the best order. Intuitively, our method is similar to phrase-based SMT, which first cuts the input sentence into phrases, then obtains the translation for each source phrase, before finally generating the target sentence by ordering the translations. Although the computational cost of our method is low, the initial experiment is promising, yielding a BLEU score of 22.44 on a standard benchmark.

2 Method

We reformulate the problem of AMR-to-text generation as an asymmetric generalized traveling salesman problem (AGTSP), a variant of TSP.

2.1 TSP and its variants

Given a *non-directed* graph G_N with n cities, supposing that there is a traveling cost between each pair of cities, TSP tries to find a tour of the minimal total cost visiting each city exactly once. In contrast, the asymmetric traveling salesman problem (ATSP) tries to find a tour of the minimal total cost on a *directed* graph, where the traveling costs between two nodes are different in each direction. Given a directed graph G_D with n nodes, which are clustered into m groups, the asymmetric generalized traveling salesman problem (AGTSP) tries to find a tour of the minimal total cost visiting each *group* exactly once.

2.2 AMR-to-text Generation as AGTSP

Given an input AMR A, each node in the AGTSP graph can be represented as (c, r), where c is a concept in A and $r = (A_{sub}, T_{sub})$ is a rule that consists of an AMR fragment containing c and a translation of the fragment. We put all nodes containing the same concept into one group, thereby translating each concept in the AMR exactly once.

To show a brief example, consider the AMR in Figure 1 and the following rules,

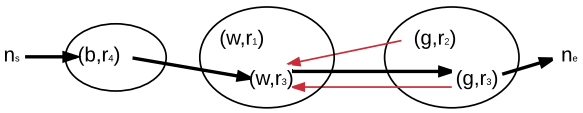


Figure 2: An example AGTSP graph

r_1	(w/want-01) ||| wants
r_2	(g/go-01) ||| to go
r_3	(w/want-01 :ARG1 g/go-01) ||| wants to go
r_4	(b/boy) ||| The boy

We build an AGTSP graph in Figure 2, where each circle represents a group and each tuple (such as (b, r_4)) represents a node in the AGTSP graph. We add two nodes n_s and n_e representing the start and end nodes respectively. Each belongs to a specific group that only contains that node, and a tour always starts with n_s and ends with n_e. Legal moves are shown in black arrows, while illegal moves are shown in red. One legal tour is $n_s \rightarrow (b, r_4) \rightarrow (w, r_3) \rightarrow (g, r_3) \rightarrow n_e$. The order in which nodes within a rule are visited is arbitrary; for a rule with N concepts, the number of visiting orders is $O(N!)$. To reduce the search space, we enforce the breadth first order by setting costs to zero or infinity. In our example, the traveling cost from (w, r_3) to (g, r_3) is 0, while the traveling cost from (g, r_3) to (w, r_3) is infinity. Traveling from (g, r_2) to (w, r_3) also has infinite cost, since there is overlap on the concept "w/want-01" between them.

The traveling cost is calculated by Algorithm 1. We first add n_s and n_e serving the same function as Figure 2. The traveling cost from n_s directly to n_e is infinite, since a tour has to go through other nodes before going to the end. On the other hand, the traveling cost from n_e to n_s is 0 (Lines 3-4), as a tour always goes back to the start after reaching the end. The traveling cost from n_s to $n_i = (c_i, r_i)$ is the model score only if c_i is the first node of the AMR fragment of r_i, otherwise the traveling cost is infinite (Lines 6-9). Similarly, the traveling cost from n_i to n_e is the model score only if c_i is the last node of the fragment of r_i. Otherwise, it is infinite (Lines 10-13). The traveling cost from $n_i = (c_i, r_i)$ to $n_j = (c_j, r_j)$ is 0 if r_i and r_j are the same rule and c_j is the next node of c_i in the AMR fragment of r_i (Lines 16-17).

A tour has to travel through an AMR fragment be-

Data: Nodes in AGTSP graph G
Result: Traveling Cost Matrix T

1 $n_s \leftarrow$ (“<s>”,“<s>”);
2 $n_e \leftarrow$ (“</s>”,“</s>”);
3 T[n_s][n_e] $\leftarrow \infty$;
4 T[n_e][n_s] $\leftarrow 0$;
5 **for** $n_i \leftarrow (c_i, r_i)$ ***in*** G **do**
6 **if** c_i = r_i.*frag.first* **then**
7 T[n_s][n_i] $\leftarrow$ ModelScore(n_s,n_i);
8 **else**
9 T[n_s][n_i] $\leftarrow \infty$;
10 **if** c_i = r_i.*frag.last* **then**
11 T[n_i][n_e] $\leftarrow$ ModelScore(n_i,n_e);
12 **else**
13 T[n_i][n_e] $\leftarrow \infty$;
14 **for** $n_i \leftarrow (c_i, r_i)$ ***in*** G **do**
15 **for** $n_j \leftarrow (c_j, r_j)$ ***in*** G **do**
16 **if** r_i = r_j *and* r_i.*frag.next*(c_i) = c_j **then**
17 T[n_i][n_j] $\leftarrow 0$
18 **else if** r_i.*frag* $\cap$ r_j.*frag* = $\emptyset$ *and* c_i = r_i.*frag.last and* c_j = r_j.*frag.first* **then**
19 T[n_i][n_j] $\leftarrow$ ModelScore(n_i,n_j)
20 **else**
21 T[n_i][n_j] $\leftarrow \infty$

Algorithm 1: Traveling cost algorithm

fore jumping to another fragment. We choose the breadth-first order of nodes within the same rule, which is guaranteed to exist, as each AMR fragment is rooted and connected. Costs along the breadth-first order within a rule r_i are set to 0, while other costs with a rule are infinite.

If r_i is not equal to r_j, then the traveling cost is the model score if there is no overlap between r_i and r_j's AMR fragment and it moves from r_i's last node to r_j's first node (Lines 18-19), otherwise the traveling cost is infinite (Lines 20-21). All other cases are illegal and we assign infinite traveling cost. We do not allow traveling between overlapping nodes, whose AMR fragments share common concepts. Otherwise the traveling cost is evaluated by a maximum entropy model, which will be discussed in detail in Section 2.4.

2.3 Rule Acquisition

We extract rules from a corpus of (sentence, AMR) pairs using the method of Peng et al. (2015). Given an aligned (sentence, AMR) pair, a *phrase-fragment pair* is a pair $([i, j], f)$, where $[i, j]$ is a span of the sentence and f represents a connected and rooted AMR fragment. A *fragment decomposition forest* consists of all possible phrase-fragment pairs that satisfy the alignment agreement for phrase-based MT (Koehn et al., 2003). The rules that we use for generation are the result of applying an MCMC procedure to learn a set of likely phrase-fragment pairs from the forests containing all possible pairs. One difference from the work of Peng et al. (2015) is that, while they require the string side to be tight (does not include unaligned words on both sides), we expand the tight phrases to incorporate unaligned words on both sides. The intuition is that they do text-to-AMR parsing, which often involves discarding function words, while our task is AMR-to-text generation, and we need to be able to fill in these unaligned words. Since incorporating unaligned words will introduce noise, we rank the translation candidates for each AMR fragment by their counts in the training data, and select the top N candidates.[1]

We also generate *concept rules* which directly use a morphological string of the concept for translation. For example, for concept “w/want-01” in Figure 1, we generate concept rules such as “(w/want-01) ||| want”, “(w/want-01) ||| wants”, “(w/want-01) ||| wanted” and “(w/want-01) ||| wanting”. The algorithm (described in section 2.2) will choose the most suitable one from the rule set. It is similar to most MT systems in creating a translation candidate for each word, besides normal translation rules. It is easy to guarantee that the rule set can fully cover every input AMR graph.

Some concepts (such as “have-rel-role-91”) in an AMR graph do not contribute to the final translation, and we skip them when generating concept rules. Besides that, we use a verbalization list[2] for concept rule generation. For rule “VERBALIZE peacekeeping TO keep-01 :ARG1 peace”, we will create a concept rule “(k/keep-01 :ARG1 (p/peace)) ||| peacekeeping” if the left-hand-side fragment appears in the target graph.

[1] Our code for grammar induction can be downloaded from https://github.com/xiaochang13/AMR-generation

[2] http://amr.isi.edu/download/lists/verbalization-list-v1.06.txt

2.4 Traveling cost

Considering an AGTSP graph whose nodes are clustered into m groups, we define the traveling cost for a tour T in Equation 1:

$$\text{cost}(n_s, n_e) = -\sum_{i=0}^{m} \log p(\text{“yes”}|n_{T_i}, n_{T_{i+1}}) \quad (1)$$

where $n_{T_0} = n_s$, $n_{T_{m+1}} = n_e$ and each n_{T_i} ($i \in [1 \dots m]$) belongs to a group that is different from all others. Here $p(\text{“yes”}|n_j, n_i)$ represents a learned score for a move from n_j to n_i. The choices before n_{T_i} are independent from choosing $n_{T_{i+1}}$ given n_{T_i} because of the Markovian property of the TSP problem. Previous methods (Zaslavskiy et al., 2009) evaluate traveling costs $p(n_{T_{i+1}}|n_{T_i})$ by using a language model. Inevitably some rules may only cover one translation word, making only bigram language models naturally applicable. Zaslavskiy et al. (2009) introduces a method for incorporating a trigram language model. However, as a result, the number of nodes in the AGTSP graph grows exponentially.

To tackle the problem, we treat it as a *local* binary (“yes” or “no”) classification problem whether we should move to n_j from n_i. We train a maximum entropy model, where $p(\text{“yes”}|n_i, n_j)$ is defined as:

$$p(\text{“yes”}|n_i, n_j) = \frac{1}{Z(n_i, n_j)} \exp\Big[\sum_{i=1}^{k} \lambda_i f_i(\text{“yes”}, n_i, n_j)\Big] \quad (2)$$

The model uses 3 real-valued features: a language model score, the word count of the concatenated translation from n_i to n_j, and the length of the shortest path from n_i's root to n_j's root in the input AMR. If either n_i or n_j is the start or end node, we set the path length to 0. Using this model, we can use whatever N-gram we have at each time. Although language models favor shorter translations, word count will balance the effect, which is similar to MT systems. The length of the shortest path is used as a feature because the concepts whose translations are adjacent usually have lower path length than others.

3 Experiments

3.1 Setup

We use the dataset of SemEval-2016 Task8 (Meaning Representation Parsing), which contains 16833 training instances, 1368 dev instances and 1371 test instances. Each instance consists of an AMR graph and a sentence representing the same meaning. Rules are extracted from the training data, and hyperparameters are tuned on the dev set. For tuning and testing, we filter out sentences that have more than 30 words, resulting in 1103 dev instances and 1055 test instances. We train a 4-gram language model (LM) with gigaword (LDC2011T07), and use BLEU (Papineni et al., 2002) as the evaluation metric. To solve the AGTSP, we use Or-tool[3].

System	Dev	Test
PBMT	13.13	16.94
OnlyConceptRule	13.15	14.93
OnlyInducedRule	17.68	18.09
OnlyBigramLM	17.19	17.75
All	21.12	22.44
JAMR-gen	**23.00**	**23.00**

Table 1: Main results.

Our graph-to-string rules are reminiscent of phrase-to-string rules in phrase-based MT (PBMT). We compare our system to a baseline (*PBMT*) that first linearizes the input AMR graph by breadth first traversal, and then adopts the PBMT system from Moses[4] to translate the linearized AMR into a sentence. To traverse the children of an AMR concept, we use the original order in the text file. The MT system is trained with the default setting on the same dataset and LM. We also compare with *JAMR-gen*[5] (Flanigan et al., 2016), which is trained on the same dataset but with a 5-gram LM from gigaword (LDC2011T07).

To evaluate the importance of each module in our system, we develop the following baselines: *OnlyConceptRule* uses only the concept rules, *OnlyInducedRule* uses only the rules induced from the fragment decomposition forest, *OnlyBigramLM* uses both types of rules, but the traveling cost is evaluated by a bigram LM trained with gigaword.

3.2 Results

The results are shown in Table 1. Our method (*All*) significantly outperforms the baseline (*PBMT*)

[3]https://developers.google.com/optimization/

[4]http://www.statmt.org/moses/

[5]https://github.com/jflanigan/jamr/tree/Generator

(w / want-01 :ARG0 (b / boy) :ARG1 (b2 / believe-01 :ARG0 (g / girl) :ARG1 b))
Ref: the boy wants the girl to believe him
All: a girl wanted to believe him
JAMR-gen: boys want the girl to believe

Table 2: Case study.

on both the dev and test sets. *PBMT* does not outperform *OnlyBigramLM* and *OnlyInducedRule*, demonstrating that our rule induction algorithm is effective. We consider rooted and connected fragments from the AMR graph, and the TSP solver finds better solutions than beam search, as consistent with Zaslavskiy et al. (2009). In addition, *OnlyInducedRule* is significantly better than *OnlyConceptRule*, showing the importance of induced rules on performance. This also confirms the reason that *All* outperforms *PBMT*. This result confirms our expectation that concept rules, which are used for fulfilling the coverage of an input AMR graph in case of OOV, are generally not of high quality. Moreover, *All* outperforms *OnlyBigramLM* showing that our maximum entropy model is stronger than a bigram language model. Finally, *JAMR-gen* outperforms *All*, while *JAMR-gen* uses a higher order language model than *All* (5-gram VS 4-gram).

For rule coverage, around 31% AMR graphs and 84% concepts in the development set are covered by our induced rules extracted from the training set.

3.3 Analysis and Discussions

We further analyze *All* and *JAMR-gen* with an example AMR and show the AMR graph, the reference, and results in Table 2. First of all, both *All* and *JAMR-gen* outputs a reasonable translation containing most of the meaning from the AMR. On the other hand, *All* fails to recognize "boy" as the subject. The reason is that the feature set does not include edge labels, such as "ARG0" and "ARG1". Finally, neither *All* and *JAMR-gen* can handle the situation when a re-entrance node (such as "b/boy" in example graph of Table 2) need to be translated twice. This limitation exists for both works.

4 Related Work

Our work is related to prior work on AMR (Banarescu et al., 2013). There has been a list of work on AMR parsing (Flanigan et al., 2014; Wang et al., 2015; Peng et al., 2015; Vanderwende et al., 2015; Pust et al., 2015; Artzi et al., 2015), which predicts the AMR structures for a given sentence. On the reverse direction, Flanigan et al. (2016) and our work here study sentence generation from a given AMR graph. Different from Flanigan et al. (2016) who map a input AMR graph into a tree before linearization, we apply synchronous rules consisting of AMR graph fragments and text to directly transfer a AMR graph into a sentence. In addition to AMR parsing and generation, there has also been work using AMR as a semantic representation in machine translation (Jones et al., 2012).

Our work also belongs to the task of text generation (Reiter and Dale, 1997). There has been work on generating natural language text from a bag of words (Wan et al., 2009; Zhang and Clark, 2015), surface syntactic trees (Zhang, 2013; Song et al., 2014), deep semantic graphs (Bohnet et al., 2010) and logical forms (White, 2004; White and Rajkumar, 2009). We are among the first to investigate generation from AMR, which is a different type of semantic representation.

5 Conclusion

In conclusion, we showed that a TSP solver with a few real-valued features can be useful for AMR-to-text generation. Our method is based on a set of graph to string rules, yet significantly better than a PBMT-based baseline. This shows that our rule induction algorithm is effective and that the TSP solver finds better solutions than beam search.

Acknowledgments

We are grateful for the help of Jeffrey Flanigan, Lin Zhao, and Yifan He. This work was funded by NSF IIS-1446996, and a Google Faculty Research Award. Yue Zhang is funded by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 1699–1710.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Bernd Bohnet, Leo Wanner, Simon Mill, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 98–106.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 748–752.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 1426–1436.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-16)*, pages 731–739.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the International Conference on Computational Linguistics (COLING-12)*, pages 1359–1376.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 48–54.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, pages 311–318.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL-15)*, pages 731–739.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 1143–1154.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-14)*, pages 1522–1528.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*, pages 26–30.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL-09)*, pages 852–860.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*, pages 366–375.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 410–419.

Michael White. 2004. Reining in CCG chart realization. In *International Conference on Natural Language Generation (INLG-04)*, pages 182–191.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-09)*, pages 333–341.

Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.

Yue Zhang. 2013. Partial-tree linearization: Generalized word ordering for text synthesis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2232–2238.

Learning to Capitalize with Character-Level Recurrent Neural Networks: An Empirical Study

Raymond Hendy Susanto[†] and **Hai Leong Chieu**[‡] and **Wei Lu**[†]
[†]Singapore University of Technology and Design
[‡]DSO National Laboratories
{raymond_susanto,luwei}@sutd.edu.sg
chaileon@dso.org.sg

Abstract

In this paper, we investigate case restoration for text without case information. Previous such work operates at the word level. We propose an approach using character-level recurrent neural networks (RNN), which performs competitively compared to language modeling and conditional random fields (CRF) approaches. We further provide quantitative and qualitative analysis on how RNN helps improve truecasing.

1 Introduction

Natural language texts (e.g., automatic speech transcripts or social media data) often come in non-standard forms, and normalization would typically improve the performance of downstream natural language processing (NLP) applications. This paper investigates a particular sub-task in text normalization: case restoration or *truecasing*. Truecasing refers to the task of restoring case information (uppercase or lowercase) of characters in a text corpus. Case information is important for certain NLP tasks. For example, Chieu and Ng (2002) used unlabeled mixed case text to improve named entity recognition (NER) on uppercase text.

The task often presents ambiguity: consider the word "apple" in the sentences "he bought an apple" and "he works at apple". While the former refers to a fruit (hence, it should be in lowercase), the latter refers to a company name (hence, it should be capitalized). Moreover, we often need to recover the case information for words that are previously unseen by the system.

In this paper, we propose the use of character-level recurrent neural networks for truecasing. Previous approaches for truecasing are based on word level approaches which assign to each word one of the following labels: all lowercase, all uppercase, initial capital, and mixed case. For mixed case words, an additional effort has to be made to decipher exactly how the case is mixed (e.g., *MacKenzie*). In our approach, we propose a generative, character-based recurrent neural network (RNN) model, allowing us to predict exactly how cases are mixed in such words.

Our main contributions are: (i) we show that character-level approaches are viable compared to word-level approaches, (ii) we show that character-level RNN has a competitive performance compared to character-level CRF, and (iii) we provide our quantitative and qualitative analysis on how RNN helps improve truecasing.

2 Related Work

Word-based truecasing The most widely used approach works at the word level. The simplest approach converts each word to its most frequently seen form in the training data. One popular approach uses HMM-based tagging with an N-gram language model, such as in (Lita et al., 2003; Nebhi et al., 2015). Others used a discriminative tagger, such as MEMM (Chelba and Acero, 2006) or CRF (Wang et al., 2006). Another approach uses statistical machine translation to translate uncased text into a cased one. Interestingly, no previous work operated at the character level. Nebhi et al. (2015) investigated truecasing in tweets, where truecased cor-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2090–2095,
Austin, Texas, November 1-5, 2016.

pora are less available.

Recurrent neural networks Recent years have shown a resurgence of interest in RNN, particularly variants with long short-term memory (Hochreiter and Schmidhuber, 1997) or gated recurrent units (Cho et al., 2014). RNN has shown an impressive performance in various NLP tasks, such as machine translation (Cho et al., 2014; Luong et al., 2015), language modeling (Mikolov et al., 2010; Kim et al., 2016), and constituency parsing (Vinyals et al., 2015). Nonetheless, understanding the mechanism behind the successful applications of RNN is rarely studied. In this work, we take a closer look at our trained model to interpret its internal mechanism.

3 The Truecasing Systems

In this section, we describe the truecasing systems that we develop for our empirical study.

3.1 Word-Level Approach

A word-level approach truecases one word at a time. The first system is a tagger based on HMM (Stolcke, 2002) that translates an uncased sequence of words to a corresponding cased sequence. An N-gram language model trained on a cased corpus is used for scoring candidate sequences. For decoding, the Viterbi algorithm (Rabiner, 1989) computes the highest scoring sequence.

The second approach is a discriminative classifier based on linear chain CRF (Lafferty et al., 2001). In this approach, truecasing is treated as a sequence labeling task, labelling each word with one of the following labels: all lowercase, all uppercase, initial capital, and mixed case. For our experiments, we used the truecaser in Stanford's NLP pipeline (Manning et al., 2014). Their model includes a rich set of features (Finkel et al., 2005), such as surrounding words, character N-grams, word shape, etc.

Dealing with mixed case Both approaches require a separate treatment for mixed case words. In particular, we need a gazetteer that maps each word to its mixed case form – either manually created or statistically collected from training data. The character-level approach is motivated by this: Instead of treating them as a special case, we train our model to capitalize a word character by character.

3.2 Character-Level Approach

A character-level approach converts each character to either uppercase or lowercase. In this approach, mixed case forms are naturally taken care of, and moreover, such models would generalize better to unseen words. Our third system is a linear chain CRF that makes character-level predictions. Similar to the word-based CRF, it includes surrounding words and character N-grams as features.

Finally, we propose a character-level approach using an RNN language model. RNN is particularly useful for modeling sequential data. At each time step t, it takes an input vector x_t and previous hidden state h_{t-1}, and produces the next hidden state h_t. Different recurrence formulations lead to different RNN models, which we will describe below.

Long short-term memory (LSTM) is an architecture proposed by Hochreiter and Schmidhuber (1997). It augments an RNN with a memory cell vector c_t in order to address learning long range dependencies. The content of the memory cell is updated additively, mitigating the *vanishing gradient problem* in vanilla RNNs (Bengio et al., 1994). Read, write, and reset operations to the memory cell are controlled by input gate i, output gate o, and forget gate f. The hidden state is computed as:

$$
\begin{aligned}
i_t &= \sigma(W_i h_{t-1} + U_i x_t) && (1)\\
o_t &= \sigma(W_o h_{t-1} + U_o x_t) && (2)\\
f_t &= \sigma(W_f h_{t-1} + U_f x_t) && (3)\\
g_t &= \tanh(W_g h_{t-1} + U_g x_t) && (4)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t && (5)\\
h_t &= o_t \odot \tanh(c_t) && (6)
\end{aligned}
$$

where σ and tanh are element-wise sigmoid and hyperbolic tangent functions, and W_j and U_j are parameters of the LSTM for $j \in \{i, o, f, g\}$.

Gated recurrent unit (GRU) is a gating mechanism in RNN that was introduced by Cho et al. (2014). They proposed a hidden state computation with reset and update gates, resulting in a simpler LSTM variant:

$$
\begin{aligned}
r_t &= \sigma(W_r h_{t-1} + U_r x_t) && (7)\\
z_t &= \sigma(W_z h_{t-1} + U_z x_t) && (8)\\
\widetilde{h}_t &= \tanh(W_h(r_t \odot h_{t-1}) + U_h x_t) && (9)\\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t && (10)
\end{aligned}
$$

	EN-Wikipedia				EN-WSJ				EN-Reuters				DE-ECI			
	Acc.	P	R	F_1	Acc.	P	R	F_1	Acc.	P	R	F_1	Acc.	P	R	F_1
Word-based Approach																
LM ($N = 3$)	94.94	89.34	84.61	86.91	95.59	91.56	78.79	84.70	94.57	93.49	79.43	85.89	95.67	97.84	87.74	92.51
LM ($N = 5$)	94.93	89.42	84.41	86.84	95.62	91.72	78.79	84.77	94.66	93.92	79.47	86.09	95.68	97.91	87.70	92.53
CRF-WORD	96.60	94.96	87.16	90.89	97.64	93.12	90.41	91.75	96.58	93.91	87.19	90.42	96.09	98.41	88.73	93.32
Chelba and Acero (2006)		n/a			97.10	-	-	-		n/a				n/a		
Character-based Approach																
CRF-CHAR	96.99	94.60	89.27	91.86	97.00	94.17	84.46	89.05	97.06	94.63	89.12	91.80	98.26	96.95	96.59	96.77
LSTM-SMALL	96.95	93.05	90.59	91.80	97.83	93.99	90.92	***92.43***	97.37	93.08	92.63	92.86	98.70	97.52	97.39	97.46
LSTM-LARGE	97.41	93.72	92.67	***93.19***	97.72	93.41	90.56	91.96	97.76	94.08	93.50	***93.79***	99.00	98.04	97.98	***98.01***
GRU-SMALL	96.46	92.10	89.10	90.58	97.36	92.28	88.60	90.40	97.01	92.85	90.84	91.83	98.51	97.15	96.96	97.06
GRU-LARGE	96.95	92.75	90.93	91.83	97.27	90.86	90.20	90.52	97.12	92.02	92.07	92.05	98.35	96.86	96.79	96.82

Table 2: Truecasing performance in terms of precision (P), recall (R), and F_1. All improvements of the best performing character-based systems (**bold**) over the best performing word-based systems (underlined) are statistically significant using sign test ($p < 0.01$). All improvements of the best performing RNN systems (*italicized*) over CRF-CHAR are statistically significant using sign test ($p < 0.01$).

At each time step, the conditional probability distribution over next characters is computed by linear projection of h_t followed by a softmax:

$$P(x_t = k|x_{1:t-1}) = \frac{\exp(w_k h_t)}{\sum_{j=1}^{|V|} \exp(w_j h_t)} \quad (11)$$

where w_k is the k-th row vector of a weight matrix W. The probability of a sequence of characters $x_{1:T}$ is defined as:

$$P(x_{1:T}) = \prod_{t=1}^{T} P(x_t|x_{1:t-1}) \quad (12)$$

Similar to the N-gram language modeling approach we described previously, we need to maximize Equation 12 in order to decode the most probable cased sequence. Instead of Viterbi decoding, we approximate this using a beam search.

4 Experiments and Results

4.1 Datasets and Tools

Our approach is evaluated on English and German datasets. For English, we use a Wikipedia corpus from (Coster and Kauchak, 2011), WSJ corpus (Paul and Baker, 1992), and the Reuters corpus from the CoNLL-2003 shared task on named entity recognition (Tjong Kim Sang and De Meulder, 2003). For German, we use the ECI Multilingual Text Corpus from the same shared task. Each corpus is tokenized.[1] The input test data is lowercased. Table 1 shows the statistics of each corpus split into training, development, and test sets.

We use SRILM (Stolcke, 2002) for N-gram language model training ($N \in \{3, 5\}$) and HMM decoding. The word-based CRF models are trained using the CRF implementation in Stanford's CoreNLP 3.6.0 (Finkel et al., 2005). We use a recommended configuration for training the truecaser.We use CRFSuite version 0.12 (Okazaki, 2007) to train the character-based CRF model. Our feature set includes character N-grams ($N \in \{1, 2, 3\}$) and word N-grams ($N \in \{1, 2\}$) surrounding the current character. We tune the ℓ_2 regularization parameter λ using a grid search where $\lambda \in \{0.01, 0.1, 1, 10\}$.

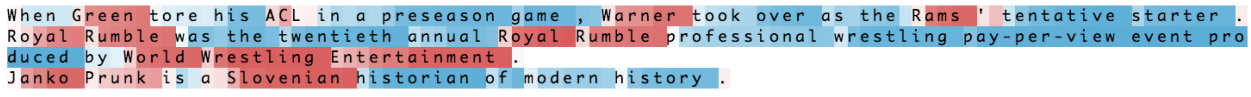
Corpus	Split	#words	#chars
EN-Wiki	train	2.9M	16.1M
	dev	294K	1.6M
	test	32K	176K
EN-WSJ	train	1.9M	10.5M
	dev	101K	555K
	test	9K	48K
EN-Reuters	train	3.1M	16.8M
	dev	49K	264K
	test	44K	231K
DE-ECI	train	2.8M	18M
	dev	51K	329K
	test	52K	327K

Table 1: Statistics of the data.

We use an open-source character RNN implementation.[2] We train a SMALL model with 2 layers and 300 hidden nodes, and a LARGE model with 3 layers and 700 hidden nodes. We also vary the hidden unit type (LSTM/GRU). The network is trained using truncated backpropagation for 50 time steps. We use a mini-batch stochastic gradient descent with batch size 100 and RMSprop update (Tieleman and Hinton, 2012). We use dropout regularization (Srivastava et al., 2014) with 0.25 probability. We choose the model with the smallest validation loss after 30 epochs. For decoding, we set beam size to 10. The experimental settings are reported in more depth in the supplementary materials. Our system and code are publicly available at `http://statnlp.org/research/ta/`.

[1] News headlines, which are all in uppercase, are discarded.

[2] `https://github.com/karpathy/char-rnn`



(a) Samples from EN-Wiki

(b) Samples from DE-ECI

Figure 1: Cells that are sensitive to lowercased and capitalized words. Text color represents activations ($-1 \leq \tanh(c_t) \leq 1$): positive is blue, negative is red. Darker color corresponds to greater magnitude.

4.2 Results

Table 2 shows the experiment results in terms of precision, recall, and F_1. Most previous work did not evaluate their approaches on the same dataset. We compare our work to (Chelba and Acero, 2006) using the same WSJ sections for training and evaluation on 2M word training data. Chelba and Acero only reported error rate, and all our RNN and CRF approaches outperform their results in terms of error rate.

First, the word-based CRF approach gives up to 8% relative F_1 increase over the LM approach. Other than WSJ, moving to character level further improves CRF by 1.1-3.7%, most notably on the German dataset. Long compound nouns are common in the German language, which generates many out-of-vocabulary words. Thus, we hypothesize that character-based approach improves generalization. Finally, the best F_1 score for each dataset is achieved by the RNN variants: 93.19% on EN-Wiki, 92.43% on EN-WSJ, 93.79% on EN-Reuters, and 98.01% on DE-ECI.

We highlight that different features are used in CRF-WORD and CRF-CHAR. CRF-CHAR only includes simple features, namely character and word N-grams and sentence boundary indicators. In contrast, CRF-WORD contains a richer feature set that is predefined in Stanford's truecaser. For instance, it includes word shape, in addition to neighboring words and character N-grams. It also includes more feature combinations, such as the concatenation of the word shape, current label, and previous label. Nonetheless, CRF-CHAR generally performs better than CRF-WORD. Potentially, CRF-CHAR can be improved further by using larger N-grams. The decision to use simple features is for optimizing the training speed. Consequently, we are able to dedicate more time for tuning the regularization weight.

Training a larger RNN model generally improves performance, but it is not always the case due to possible overfitting. LSTM seems to work better than GRU in this task. The GRU models have 25% less parameters. In terms of training time, it took 12 hours to train the largest RNN model on a single Titan X GPU. For comparison, the longest training time for a single CRF-CHAR model is 16 hours. Training LM and CRF-WORD is much faster: 30 seconds and 5.5 hours, respectively, so there is a speed-accuracy trade-off.

5 Analysis

5.1 Visualizing LSTM Cells

An interesting component of LSTM is its memory cells, which is supposed to store long range dependency information. Many of these memory cells are not human-interpretable, but after introspecting our trained model, we find a few memory cells that are sensitive to case information. In Figure 1, we plot the memory cell activations at each time step (i.e., $\tanh(c_t)$). We can see that these cells activate differently depending on the case information of a word (towards -1 for uppercase and +1 for lowercase).

5.2 Case Category and OOV Performance

Corpus	Lower	Cap.	Upper	Mixed	OOV
EN-Wiki	79.91	18.67	0.91	0.51	2.40
EN-WSJ	84.28	13.06	2.63	0.03	3.11
EN-Reuters	78.36	19.80	1.53	0.31	5.37
DE-ECI	68.62	29.15	1.02	1.21	4.01

Table 3: Percentage distribution of the case categories and OOV words

In this section, we analyze the system performance on each case category. First, we report the percentage distribution of the case categories in each test set in Table 3. For both languages, the most frequent case category is lowercase, followed by capitalization, which generally applies to the first word

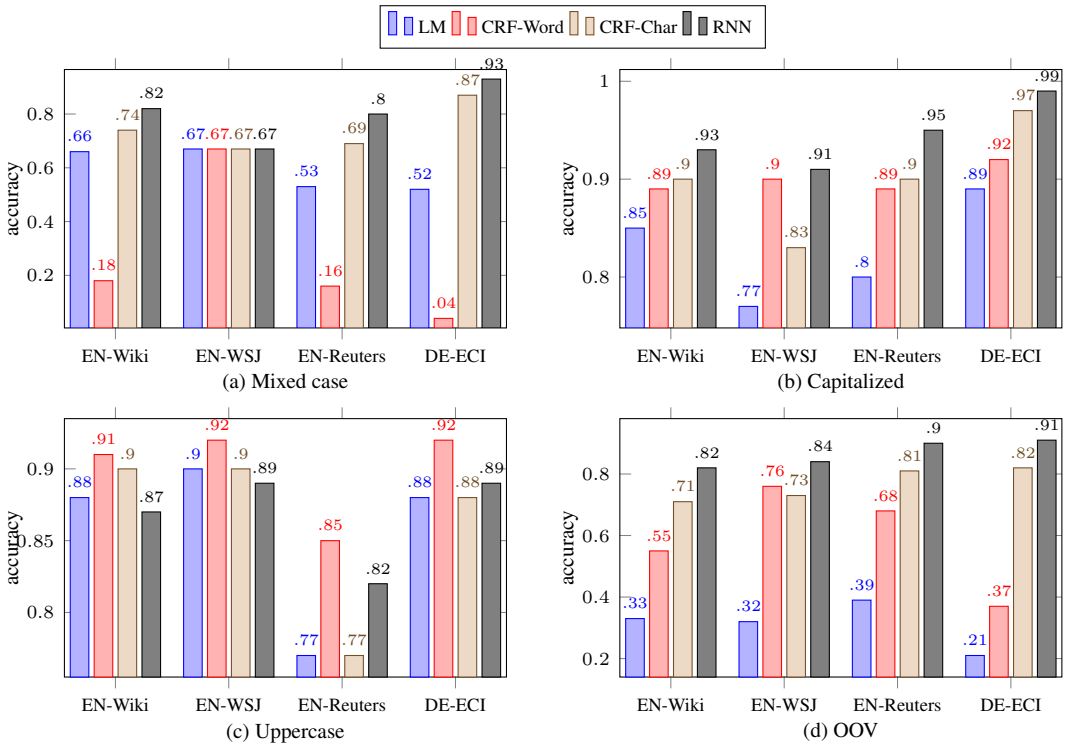


Figure 2: Accuracy on mixed case (a), capitalized (b), uppercase (c), and OOV words (d).

in the sentence and proper nouns. The uppercase form, which is often found in abbreviations, occurs more frequently than mixed case for English, but the other way around for German.

Figure 2 (a) shows system accuracy on mixed case words. We choose the best performing LM and RNN for each dataset. Character-based approaches have a better performance on mixed case words than word-based approaches, and RNN generally performs better than CRF. In CRF-WORD, surface forms are generated after label prediction. This is more rigid compared to LM, where the surface forms are considered during decoding.

In addition, we report system accuracy on capitalized words (first letter uppercase) and uppercase words in Figure 2 (b) and (c), respectively. RNN performs the best on capitalized words. On the other hand, CRF-WORD performs the best on uppercase. We believe this is related to the rare occurrences of uppercase words during training, as shown in Table 3. Although mixed case occurs more rarely in general, there are important clues, such as character prefix. CRF-CHAR and RNN have comparable performance on uppercase. For instance, there are only 2 uppercase words in WSJ that were predicted differently between CRF-CHAR and RNN. All systems perform equally well (∼99% accuracy) on lowercase. Overall, RNN has the best performance.

Last, we present results on out-of-vocabulary (OOV) words with respect to the training set. The statistics of OOV words is given in Table 3. The system performance across datasets is reported in Figure 2 (d). We observe that RNN consistently performs better than the other systems, which shows that it generalizes better to unseen words.

6 Conclusion

In this work, we conduct an empirical investigation of truecasing approaches. We have shown that character-level approaches work well for truecasing, and that RNN performs competitively compared to language modeling and CRF. Future work includes applications in informal texts, such as tweets and short messages (Muis and Lu, 2016).

Acknowledgments

We would also like to thank the anonymous reviewers for their helpful comments. This work is supported by MOE Tier 1 grant SUTDT12015008.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Ciprian Chelba and Alex Acero. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Hai Leong Chieu and Hwee Tou Ng. 2002. Teaching a weaker classifier: Named entity recognition on upper case text. In *Proceedings of ACL*, pages 481–488.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of ACL-HLT*, pages 665–669.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of ACL*, pages 363–370.

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

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *Proceedings of ICLR*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *Proceedings of ACL*, pages 152–159.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL System Demonstrations*, pages 55–60.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, pages 1045–1048.

Aldrian Obaja Muis and Wei Lu. 2016. Weak semi-Markov CRFs for noun phrase chunking in informal text. In *Proceedings of NAACL*.

Kamel Nebhi, Kalina Bontcheva, and Genevieve Gorrell. 2015. Restoring capitalization in #tweets. In *Proceedings of WWW Companion*, pages 1111–1115.

Naoaki Okazaki. 2007. CRFsuite: A fast implementation of conditional random fields (CRFs).

Douglas B Paul and Janet M Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the Workshop on Speech and Natural Language*, pages 357–362.

Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*, pages 142–147.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of NIPS*, pages 2755–2763.

Wei Wang, Kevin Knight, and Daniel Marcu. 2006. Capitalizing machine translation. In *Proceedings of NAACL-HLT*, pages 1–8.

The Effects of the Content of FOMC Communications on US Treasury Rates

Christopher Rohlfs[1] and **Sunandan Chakraborty**[2] and **Lakshminarayanan Subramanian**[2]
[1]Morgan Stanley
[2]New York University
New York, USA
Christopher.Rohlfs@morganstanley.com, Sunandan@cims.nyu.edu, lakshmi@cims.nyu.edu

Abstract

This study measures the effects of Federal Open Market Committee text content on the direction of short- and medium-term interest rate movements. Because the words relevant to short- and medium-term interest rates differ, we apply a supervised approach to learn distinct sets of topics for each dependent variable being examined. We generate predictions with and without controlling for factors relevant to interest rate movements, and our prediction results average across multiple training-test splits. Using data from 1999-2016, we achieve 93% and 64% accuracy in predicting Target and Effective Federal Funds Rate movements and 38%-40% accuracy in predicting longer term Treasury Rate movements. We obtain lower but comparable accuracies after controlling for other macroeconomic and market factors.

1 Introduction

This study uses the verbal content of Federal Open Market Committee (FOMC) public communications to predict the directions of interest rate movements on the days those communications are released. The FOMC, who determines government policies relevant to interest rates, meets roughly eight times a year and releases a statement after each meeting. The FOMC is known to be an important mover of markets, and economic research has found that equity and interest rate markets tend to move when FOMC communications are released (Farka and Fleissig, 2012; Gürkaynak *et al.*, 2005; Mueller, 2015; Rosa, 2011) that the policy actions alone do not explain these responses (and thus the content of the text must be responsible) (Gürkaynak *et al.*, 2005), and that the directions of market movements coincide with a human-coded measure of the sentiment expressed in the texts (Rosa, 2011). Writers in the finance industry and in the popular press have also examined word clouds of FOMC minutes (Cofnas, 2010; Durden, 2011) and have discussed the market implications of the total number of words included in FOMC minutes (Fitz-gerald, 2014; Kennedy, 2014; Wynne, 2013).

A growing body of research applies NLP methods to understand the market effects from the contents of these texts. Researchers have applied Latent Semantic Analysis (LSA) to describe the key topics covered in FOMC minutes, obtaining insights into the FOMC's deliberation process (Hansen *et al.*, 2015; Fligstein *et al.*, 2014; Schonhardt, 2013). Additionally, researchers have used NLP-derived measures of the content of FOMC minutes to predict equity and interest rate volatilities; (Boukus and Rosenberg, 2006) use LSA-defined topics in a regression context, and (Zadeh and Zollman, 2009) apply a dependency-based measure of text content to an expert-classified set of financially relevant words and then use both regression and SVM to predict volatility. Papers have found temporary effects of the sentiments from company-specific news articles and message board postings on stock prices and volatility, company earnings, and trading volumes, using dictionary-based sentiment measures (Davis *et al.*, 2012; Tetlock, 2007; Tetlock *et al.*, 2007; Tetlock, 2011) as well as sentiment measures that are trained on a human-classified subsample (Antweiler

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2096–2102,
Austin, Texas, November 1-5, 2016.

and Frank, 2004, 2006; Das and Chen, 2007).[1] Studies have found temporary effects even when information is "stale" (Tetlock, 2011) and also that short-sales precede negative news (Fox *et al.*, 2009/2010). Researchers also find that the readability of corporate filings is positively associated with earnings and the precision of analysts' forecasts about the company (Li, 2008; Lehavy *et al.*, 2011).

The current study builds upon this literature by examining a somewhat different question than previous researchers do and by applying a different set of techniques that are particularly well-suited for measuring the market effects of texts. Rather than examine the texts' effects on *volatility*, which increases in response to both positive and negative sentiments, we predict the *direction* in which interest rates move, which is the focus of market participants as well as the FOMC texts themselves.[2] The question we ask is also somewhat different than that examined in the literature because we analyze the relatively short FOMC statements that are released immediately following the meetings—and contain the key market-moving content (Gürkaynak *et al.*, 2005; Mueller, 2015)—rather than on the lengthier minutes that have more text to analyze but are only released after the key information from the statements has been available for three weeks.

In addition to making contributions specific to our application, this study highlights methods that are particularly useful for measuring the market effects of text content. FOMC communications are known to provide distinct information about short- versus medium- or long-term policies (Gürkaynak *et al.*, 2005). We consequently use MedLDA (Zhu *et al*, 2009), a supervised topic model, to learn separately the sets of words that are most predictive of movements in short- and medium-term interest rates[3] Through this supervised topic model, we generate topics, based upon context (which words appear together) as well as co-movement with the outcome variables being studied. Hence, the varies depending upon which dependent variable is being considered. Second, we address possible bias from one important set of omitted variables—releases of macroeconomic data, as discussed by (Rosa, 2011)—by estimating specifications in which we control for those factors separately and predict whether interest rates moved more or less than would be expected based upon the latest data on the macroeconomic environment. By examining an immediate market response to the publication of text and controlling for potential confounding factors, this study demonstrates one way in which NLP approaches, in addition to their value in classifying text content, can be applied to estimate statements' causal effects. We control for the effects of macroeconomic data and time-specific factors like day-of-week effects and time trends using only observations from non-FOMC dates, so that we do not lose degrees of freedom in our estimation. Third, unlike Boukus and Rosenberg (2006) and Hendry and Madeley (2010) but similarly to Zadeh and Zollman (2009), we split the sample into training and test sets in order to limit overfitting in our predicted values. Zadeh and Zollman (2009) use data from 1967-2000 as a training set, and then they test their model on data from 2001-2008. Given the importance of context in predicting interest rate movements, we instead restrict our sample to observations from meetings from May 1999 to May 2016[4]. Because autocorrelation in our dependent variables is relatively limited, we treat the observations as independent and, among observations in our sample, average our test performance across multiple training-test splits.

2 Market Effects of Text Content

2.1 Overview of Text Content

FOMC statements contain information about many aspects of the economy, including interest rates, the money supply, inflation, unemployment, and economic growth. These communications are highly repetitive, often containing nearly identical sentences and sentence structures from previous meet-

[1] While not examining market data, (Chua *et al.*, 2009) also examines the problem of classifying sentiment in message board postings.

[2] A related study has applied LDA to measure the impacts on returns and volatility of communications from the Bank of Canada (Hendry and Madeley, 2010).

[3] Other classification methods that we attempted but found to be less effective include regression of rate movements on word count, logit estimation on the frequencies of the most common words, and k-nearest neighbor estimation using a word2vec similarity measure (Mikolov, 2013).

[4] May 1999 was the date of the last major redesign of the FOMC statements

ings. Slight changes in the wordings are known to have major effects on markets (Gürkaynak *et al.*, 2005).

Pre-processing of text: In order to convert the text into a format that can be easily processed, we perform several cleaning operations to the texts. Non-alphabetic characters are removed, and the texts are converted to lower case. Each document is separated into a bag of words, and common words (*e.g.,* mr and federal) and stop words are deleted using the stopwords list from nltk.corpus in Python. Words are stemmed using the Porter stemming algorithm (stem from stemming.porter2 in Python), and one-letter words are dropped.

2.2 MedLDA

LDA (Latent Dirichlet Allocation) (Blei *et al.*, 2003) is an unsupervised model, whereas supervised topic model (sLDA) (Blei and McAuliffe, 2007) introduces a response variable to LDA for each document. Max-Entropy Discrimination LDA (MedLDA) (Zhu *et al*, 2009) is max-margin variant of the supervised topic models. MedLDA can be built for both regression and classification prediction tasks. In this study we employed the model built for classification task. For classification, the response variables y are discrete having values $\{1, 0, -1\}$ denoting the movements of the interest rates. Hence, we consider the multi-class classification version of the MedLDA. It is defined based on a Support Vector Machine (SVM), which integrates the max-margin principle with an underlying LDA model for topics. Formally, the probabiltiies associated with max-entropy discrimination topic models (MedTM) can be generally defined as:

$$min_d \mathcal{L}(q(H)) + KL(q(\Gamma)||p_p(\Gamma)) + U(\xi) \quad (1)$$

where H are hidden variables (e.g., (θ, z) in LDA); are the parameters of the model pertaining to the prediction task (e.g., η in sLDA); Γ are the parameters of the underlying topic model (e.g., the Dirichlet parameter α); and L is a variational upper bound of the negative log likelihood associated with the underlying topic model. U is a convex function over slack variables. For the general MedTM model, we can develop a similar variational EM-algorithm as for the MedLDA.

We apply the MedLDA model on the FOMC documents and considering the interest rates as the response variables (y) to compute topics that are closely related to variations in the interest rates. Eventually these topics are used to classify changes in the rates using the max-margin classifier embedded in the MedLDA model.

2.3 Controlling for Macroeconomic Information

In addition to these text-based data, we supply our classifier with "control" variables describing the latest releases of macroeconomic variables. The macroeconomic data considered in this analysis are three of the most important measures of US economic health: the Consumer Price Index (CPI) used to measure inflation, Unemployment, and real annualized growth in the US Gross Domestic Product (GDP). The values for all three of these statistics are publicly released on a monthly basis. The CPI and Unemployment numbers are measured on a monthly basis and are typically not updated from their initially released values. The CPI data are typically released between 15 and 20 days after the end of the month, and the Unemployment data are typically released 6 to 10 days after the end of the month. GDP is measured on a quarterly basis, and three estimates are provided: "advance," "preliminary" or "second," and "final" or "third," which are released about one, two, and three months after the end of the quarter, respectively. The final GDP numbers are occasionally revised in later releases. Our release date data and some of the macroeconomic statistics were obtained from direct requests to the U.S. Bureau of Economic Analysis (B. of Econ. An. (a), 2015; B. of Econ. An. (b), 2015) and the U.S. Bureau of Labor Stats (B. of Lab. Stat. (a), 2015; B. of Lab. Stat. (d), 2009). Additional data on the GDP and unemployment numbers released were obtained from public sources (Econ. Anal. (c), 1989; Fed. Res. (a), 15).

If macroeconomic information is released on the same day as an FOMC communication, it is possible that this release could influence both the content of the FOMC statement as well as the interest rate movements that day. To avoid that possibility, we implement a modified MedLDA approach using a dependent variable that is "purged" of these potentially confounding influences. In some of our speci-

Table 1: Accuracy of Medlda Classifier after purging out of control for statements between 1999 and May, 2016 [K (topics) = 20]

Outcome variable	MedLDA			Baseline (Random Chance)[5]		
	None	Linear	Interactions	None	Linear	Interactions
Target Fed Funds Rate	0.9321	0.9160	0.8954	0.6849	0.6849	0.6849
Effective Fed Funds Rate	0.6421	0.4479	0.5112	0.4589	0.4658	0.4658
Median Treasury Rate	0.4209	0.3803	0.4012	0.4589	0.4247	0.4247
Average Treasury Rate	0.3803	0.4611	0.3924	0.4726	0.4041	0.4041

fications, we first regress the interest rate movements of interest on these macroeconomic indicators. Our main set of controls includes the latest values for the most recent two values of the unemployment rate, GDP growth rate, and CPI inflation rate and their changes, a daily time trend, and year, month, and day-of-the-week dummies. Some specifications use this set, and others add the full set of two-way interactions across these different variables. For both the main and the interacted set, we regress the change in the rate of interest on the full set of controls for the full set of non-FOMC dates from May 1999 through May 2016. Hence, we estimate the relationship between interest rate movements and the releases of macroeconomic data using dates in which FOMC statements or minutes were not released. Using the coefficients from these regressions, we generate residuals of interest rate movements for the FOMC dates and then create indicators for whether the residual was positive or negative for that interest rate movement on that FOMC date.

3 Empirical Results

We randomly split the data, containing 146 data points (FOMC statements and corresponding movements in the interest rates from May 1999 through May 2016) into a a 80-20% train-test set split to compute the accuracy of the model to predict the movement. For each experiment, we varied the number of topics (K) to see which value of K is giving the best accuracy. In most cases, the best accuracy is given by $K = 20$. The results presented in Table 1 shows the average accuracies of predicting the movements of the interest rates after purging the outcome variables out of control. The presented accuracies are the results of 20 fold validation. When no controls are used, our accuracy is 93% and 64% for the Target and Effective Federal Funds Rates (both better than random chance) and 42% and 38% for the Median and Average Treasury Rates[6] The specifications with control variables have similar but somewhat lower accuracy rates. Hence, our text-mining approach is comparable in effectiveness at measuring whether interest rates moved more or less than expected, after controlling for the economic environment, than it is at predicting the raw directions of movement. MedLDA model is compared against a simple baseline. The baseline is the accuracy, if the interest rate movements are randomly guessed from the prior distribution of each of the interest rates under the different controls. For the Target and Effective rates, the MedLDA model outperforms the baseline with a great margin and for the Median and Average rate, the performance is slightly poorer.

The high target rate prediction accuracy suggests that the MedLDA model can effectively associate the text contents of the meetings with the movements in the rate, even though the numeric values are dropped from the text. Similar arguments can be applied to the effective rate prediction. On the other hand, treasury rates are not directly connected to the text of the FOMC statements, so the factors influencing these rates are not present or mentioned in the text. Thus, to have a better prediction accuracies for these variables information from other sources are necessary which is beyond the scope of this paper. However, the present FOMC meeting might give an indication to future FOMC plans and thus, to the

[5]Our random chance baseline is a classifier that always selects the most likely of the three outcomes (increase, decrease, or no change) based upon their frequencies in the full dataset.

[6]Median Treasury Rate is the median of the -1, 0, and 1 classifications among the movements of the 3m, 1y, 3y, 5y, and 10y Treasury Rates. Average Treasury Rate is the -1/0/1 classification of the average of those rates.

treasury rates. Hence, the prediction accuracies are not much worse than the baseline.

4 Conclusion

This study measures the effects of text-based information released by the FOMC on daily interest rate movements. We used the medLDA model on a set of 146 docs and obtain accuracies of 93% and 64% in predicting the Federal Funds Target Rate and the Effective Rate.

References

Werner Antweiler and Murray Z. Frank. Is all that talk just noise? The information content of internet stock message boards. *J Fin 59*(3). Jun 2004.

Werner Antweiler and Murray Z. Frank. Do US stock markets typically overreact to corporate news stories? Unpublished manuscript. Aug 2006. Available at SSRN: http://ssrn.com/abstract=878091

David M. Blei and Jon D. McAuliffe. Supervised topic models. In *NIPS*, 2007.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

Bloomberg. Daily historical time-series of Federal Funds Target Rate and Futures prices for Federal Funds Rate and US Treasuries. March 25, 2016.

Ellyn Boukus and Joshua V. Rosenberg. The information content of FOMC minutes. 2006 http://ssrn.com/abstract=922312

Christopher C. Chua, Maria Milosavljevic, and James R. Curran. A sentiment detection engine for internet stock message boards. Proceedings of the Australasian Language Technology Association Workshop. Dec 2009.

Abe Cofnas. *Sentiment indicators: renko, price break, kagi, point & figure —what they are and how to use them to trade*. Hoboken, NJ: Wiley. 2010.

Deborah J. Danker and Matthew M. Luecke. Background on FOMC meeting minutes. *Federal Reserve Bulletin 91*(2). Spr 2005.

Sanjiv R. Das and Mike Y. Chen. Yahoo! for Amazon: sentiment extraction from small talk on the web. *Management Science 53*(9). Sep 2007.

Angela K. Davis, Jeremy M. Piger, and Lisa M. Sedor. Beyond the numbers: measuring the information content of earnings press release language. *Contemporary Accounting Research, 29*(3). Fall 2012.

Tyler Durden. FOMC minutes word cloud and key word count. *Zero Hedge*. Aug 30, 2011. http://www.zerohedge.com/news/fomc-minutes-word-cloud-and-key-word-count

Mira Farka and Adrian R. Fleissig. The effect of FOMC statements on asset prices. *Intl Rev Appl Econ 26*(3). May 2012.

Keith Fitz-gerald. The huge economic indicator everyone misses. *Money Morning*. Mar 25, 2014. Available at: http://moneymorning.com/2014/03/25/huge-economic-indicator-everyone-misses/

Neil Fligstein, Jonah Stuart Brundage, and Michael Schultz. "Why the Federal Reserve failed to see the financial crisis of 2008: the role of 'macroeconomics' as a sense making and cultural frame." University of California, Berkeley Institute for Research on Labor and Employment (IRLE) Working Paper 111-14. September 2014.

Merritt B. Fox, Lawrence R. Glosten, and Paul C. Tetlock. Short selling and the news: a preliminary report on an empirical study. *NY Law Sch Rev 54*. 2009/2010.

Refet S. Gürkaynak, Brian Sack, and Eric T. Swanson. Do actions speak louder than words? The response of asset prices to monetary policy actions and statements. *Intl J Central Banking 1*(1). May 2005.

Stephen Hansen, Michael McMahon, and Andrea Prat. Transparency and deliberation within the FOMC: a computational linguistics approach. Unpublished manuscript. Feb 2015.

Scott Hendry and Alison Madeley. Text mining and the information content of Bank of Canada communications. Bank of Canada Working Paper 2010-31. Nov 2010.

Simon Kennedy. Word inflation accelerating at Fed justifies investor confusion. *Bloomberg*. Sep 18, 2014.

Reuven Lehavy, Feng Li, and Kenneth Merkley. The effect of annual report readability on analyst following and the properties of their earnings forecasts. *Accounting Rev 86*(3). May 2011.

Feng Li. Annual report readability, current earnings, and earnings persistence. *J Accounting and Econ 45*(2-3). Aug 2008.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality." *Advances in Neural Information Processing Systems*, 2013, pp. 3111-9.

Philippe Mueller, Alireza Tahbaz-Salehi, and Andrea Vedolin. "Exchange rates and monetary policy uncertainty." Unpublished manuscript, December 2015.

Akin Oyedele. This is how often the Fed talks about employment and inflation. *Business Insider*. Feb 18, 2015. http://www.businessinsider.com/fomc-minutes-on-unemployment-and-inflation-2015-2

Prattle Analytics. prattle-analytics.com. 2014.

Carlo Rosa. Words that shake traders: the stock markets reaction to central bank communication in real time. *J Empirical Finance 18*(5). Dec 2011.

Cheryl Schonhardt-Bailey. *Deliberating American monetary policy: a textual analysis*. Cambridge, MA: MIT Press. Nov 2013.

Paul C. Tetlock. Giving content to investor sentiment: the role of media in the stock market. *J Fin 62*(3):1139-68. Jun 2007.

Paul C. Tetlock. All the news that's fit to reprint: do investors react to stale information? *Rev Fin Stud 24*(5). May 2011.

Paul C. Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. More than words: quantifying language to measure firms' fundamentals. *J Fin 63*(3) Jun 2008.

United States. Bureau of Economic Analysis, 2015. "GDP releases 1968 forward."

United States. Bureau of Economic Analysis, 2015. "GDP-GDI vintage history."

United States. Bureau of Economic Analysis, 1989-2015. Survey of Current Business (all months).

United States. Bureau of Labor Statistics, 2015. "CPI release dates 2009-2014."

United States. Bureau of Labor Statistics, 2015. "Release day and time for the Employment Situation news release 1966-present."

United States. Bureau of Labor Statistics, 2015. "Seasonally adjusted unemployment rate as published, 1957-present."

United States. Bureau of Labor Statistics, 2009. "CPI release dates 1953-2008."

.S. Department of the Treasury. Daily Treasury yield curve rates, 1990-2015. Accessed on 3 Apr 2015. Available at: http://www.treasury.gov/resource-center/data-chart-center/interest-rates/

United States. Federal Reserve Bank of St. Louis, 2015. "Consumer Price Index for all urban consumers: all items, index 1982-1984=100, monthly, seasonally adjusted."

U.S. Federal Reserve Bank. About the FOMC. http://www.federalreserve.gov/monetarypolicy.fomc.htm. 2015.

U.S. Federal Reserve Bank of St. Louis. Federal Reserve Economic Data. Accessed on 3 Apr 2015. Available at: http://research.stlouisfed.org/fred2/

U.S. Federal Reserve System Board of Governors. Federal Open Market Committee. Accessed Apr 2015. Available at: http://www.federalreserve.gov/monetarypolicy/fomc.htm

U.S. Federal Reserve System Board of Governors. The Federal Reserve System purposes and functions. Washington, DC: Board of Governors of the Federal Reserve System. Jun 2005.

Mark A. Wynne. A short history of FOMC communication. *Economic Letter 8*(8), Federal Reserve Bank of Dallas. Sep 2013.

Reza Bosagh Zadeh and Andreas Zollman. Predicting market volatility from Federal Reserve Board meeting minutes. Unpublished manuscript. 2009.

Jun Zhu, Amr Ahmed, and Eric P. Xing. Medlda: Maximum margin supervised topic models for regression and classification. ICML '09:1257-64, 2009.

United States. Bureau of Economic Analysis, 2015. "GDP releases 1968 forward."

United States. Bureau of Economic Analysis, 2015. "GDP-GDI vintage history."

United States. Bureau of Economic Analysis, 1989-2015. Survey of Current Business (all months).

United States. Bureau of Labor Statistics, 2015. "CPI release dates 2009-2014."

United States. Bureau of Labor Statistics, 2015. "Release day and time for the Employment Situation news release 1966-present."

United States. Bureau of Labor Statistics, 2015. "Seasonally adjusted unemployment rate as published, 1957-present."

United States. Bureau of Labor Statistics, 2009. "CPI release dates 1953-2008."

United States. Federal Reserve Bank of St. Louis, 2015. "Consumer Price Index for all urban consumers: all items, index 1982-1984=100, monthly, seasonally adjusted."

U.S. Department of the Treasury. Daily Treasury yield curve rates, 1990-2015. Accessed on 3 Apr 2015. Available at: http://www.treasury.gov/resource-center/data-chart-center/interest-rates/

U.S. Federal Reserve Bank. About the FOMC. http://www.federalreserve.gov/monetarypolicy.fomc.htm. 2015.

U.S. Federal Reserve Bank of St. Louis. Federal Reserve Economic Data. Accessed on 3 Apr 2015. Available at: http://research.stlouisfed.org/fred2/

U.S. Federal Reserve System Board of Governors. Federal Open Market Committee. Accessed Apr 2015. Available at: http://www.federalreserve.gov/monetarypolicy/fomc.htm

Learning to refine text based recommendations

Youyang Gu and **Tao Lei** and **Regina Barzilay** and **Tommi Jaakkola**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{yygu,taolei,regina,tommi}@csail.mit.edu

Abstract

We propose a text-based recommendation engine that utilizes recurrent neural networks to flexibly map textual input into continuous vector representations tailored to the recommendation task. Here, the text objects are documents such as Wikipedia articles or question and answer pairs. As neural models require substantial training time, we introduce a sequential component so as to quickly adjust the learned metric over objects as additional evidence accrues. We evaluate the approach on recommending Wikipedia descriptions of ingredients to their associated product categories. We also exemplify the sequential metric adjustment on retrieving similar Stack Exchange AskUbuntu questions. [1]

1 Introduction

Modern recommender problems involve complex objects, often described in textual form. In order to learn to predict how disparate objects may go together, it is helpful to first map them into a common representation where they are easily compared, regardless of their origin. Neural models are particularly well-suited for this task as continuous vector representations of objects can be tailored in a flexible way to the desired task. While these models have been shown to be effective across NLP tasks (Sutskever et al., 2014; Andreas et al., 2016; Hermann et al., 2015), they take considerable time to learn and are therefore ill-suited to be adjusted rapidly as additional evidence accumulates.

We cast our text-to-text recommendation problem in two phases. In the first phase, flexible neural text-to-vector mappings are learned from currently available data. Such mappings are optimized to function well in a collaborative filtering setting. For example, in the context of recommending food product categories for ingredients based on their Wikipedia pages, the continuous vectors are adjusted so that their inner product directly reflects the degree of association between the objects. Once learned, the mapping can be applied to any previously unseen text to yield the corresponding vector representation, and therefore also used for predicting associations. In the second phase, we no longer adjust text-to-vector mappings but rather parameterize and learn how the vectors are compared. For example, we can optimize the metric separately for each new ingredient based on a few category observations for that ingredient. The goal of this second phase is to specifically boost the accuracy when the neural baseline (unaware of the new evidence) would otherwise not perform well.

Our approach builds on the recent work on recurrent convolutional models to obtain text-to-vector mappings (Lei et al., 2015; Lei et al., 2016). This architecture is particularly well suited for noisy Wikipedia pages as it can learn to omit and highlight different parts of the text, as needed. The additional sequential component is a regularized logistic regression model (for ingredient-product prediction) or a ranking model (for question retrieval). We demonstrate the accuracy of the baseline neural recommender and the gains from the second sequential phase in both of these tasks.

[1] The code/data is available at https://github.com/youyanggu/rcnn.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2103–2108,
Austin, Texas, November 1-5, 2016.

2 Related Work

A great deal of recent effort has gone into developing flexible neural models for text and their use across variety of NLP tasks. This includes building vector representations for sentences and documents (Le and Mikolov, 2014), convolutional neural network models of text (Collobert and Weston, 2008; Zhang and LeCun, 2015), non-consecutive variants of CNNs (Lei et al., 2015), and compositional architectures (Socher et al., 2013), among many others. Our work is most closely related to the use of such models for question retrieval (Lei et al., 2016) but differs, in particular, in terms of our two-phase collaborative filtering formulation and the ingredient mapping task from Wikipedia pages (cf.(Sutskever et al., 2011; Song and Roth, 2015)).

3 Recommender Problems

We explore two recommender problems in this work. In the first problem, we are given a food ingredient, and our goal is to predict which product categories it could appear in. Both ingredients and product categories are provided in terms of natural language descriptions via their associated Wikipedia pages. For example, if given "tomato", we would predict "canned foods" as one likely category for the ingredient. A small number of categories appear as targets for each ingredient.

We also consider the task of predicting questions that are similar to the one provided as a query. The purpose is to facilitate effective question answering by retrieving related past questions (and the associated answers that are available). For this we use Stack Exchange's AskUbuntu question retrieval dataset used in recent work (dos Santos et al., 2015; Lei et al., 2016)

4 Approach

We explain our approach in terms of the first task: predicting product categories from ingredients. Collaborative predictions are made by mapping each ingredient into a vector representation and comparing that representation with an analogous one for product categories. We train these vectors in an end-to-end manner to function well as part of the collaborative task. The vector representations are based on Wikipedia pages that are available for most ingredients and categories in our problem. Rather than derive the vector from the entire article (which can be long), we only use the top summary section. For the AskUbuntu question-answering dataset, we make use of both the title and the question body.

We use a recurrent neural network (RNN) model to map each text description into a vector representation. Our model builds on the recurrent convolutional neural network model of (Lei et al., 2016) used to train the AskUbuntu question representations. We describe below a modified version used for ingredient-product category prediction.

Let $v_\theta(x) \in \mathbb{R}^d$ be the parameterized RNN mapping of text x into a vector representation, where d is the dimension of the hidden representation. Let x_i and z_p be the Wikipedia pages for ingredient $i \in I$ and product category $p \in P$, respectively. We use the same parameters θ to generate the representations for both ingredients and product categories due to their overall similarity. Thus $v_\theta(x_i)$ is the vector representation for ingredient i and $v_\theta(z_p)$ is the vector representation for product category p for an RNN model with parameters θ. We train the RNN model to predict each association $Y_{ip} = 1$ as a binary prediction task, i.e.,

$$P(Y_{ip} = 1|\theta) = \sigma(v_\theta(z_p) \cdot v_\theta(x_i)), \qquad (1)$$

where σ is the sigmoid function $\sigma(t) = (1 + \exp(-t))^{-1}$. The formulation is akin to a binary collaborative filtering task where user/item feature vectors are produced by the RNN. The parameters θ can be learned by back-propagating log-likelihood of the binary 0/1 predictions back to θ.

4.1 Sequential learning

Our RNN model, once trained, will be able to map any new ingredient and product category (their text descriptions) into vectors, and make a binary prediction of whether the two go together. However, training the model takes considerable time and cannot be easily adjusted in the face of new evidence, e.g., a few positive and negative categories for a previously unseen ingredient. Since RNN features are global (affecting the mapping from text to features for all ingredients/products), it is not clear how the adjustments made in light of additional information about

a specific new ingredient will impact predictions for other ingredients. We propose a sequential approach that is instead local, tailored to the new ingredient.

In order to sequentially adjust the model predictions with new evidence, we introduce parameters $w = [w_1, \ldots, w_d], w_j \in \mathbb{R}^+$ that modify the comparison of ingredient and category vectors. Specifically, the association is predicted by

$$P(Y_{ip} = 1|\theta, w) = \sigma\{v_\theta(z_p)^T \text{diag}(w) v_\theta(x_i)\}, \quad (2)$$

where $\text{diag}(w)$ is a diagonal matrix with the entries specified by w. We assume that, at this stage, the RNN parameters θ and therefore the vector representations $v_\theta(z_p)$ and $v_\theta(x_i)$ are nonadjustable. We will only update weights w in response to each new observation, separately for each ingredient. The observations can both be positive ($Y = 1$) and negative ($Y = 0$).

Because we expect a new input may only have a small number of observations, it is important to properly regularize the weights as to avoid overfitting. We append the log-likelihood objective with a regularizer

$$reg(w) = \frac{\lambda}{2} \sum_{j=1}^{d} (w_j - 1)^2 \quad (3)$$

where λ is the overall regularization parameter. Note that for large values of λ, the regularizer keeps the parameters at the default values $w_j = 1$ corresponding to the baseline RNN collaborative predictions, unmodified by the new evidence.

In the context of predicting similar questions, we use a modified binary formulation where the goal is to classify each triplet of questions (x, z_1, z_2) in terms of whether z_1 is closer to the query than z_2. In this ranking model, the probability that z_1 is closer is given by

$$\sigma\Big((v_\theta(z_1) - v_\theta(z_2))^T \text{diag}(w) v_\theta(x)\Big), \quad (4)$$

The parameters w are again trained from observed additional triplet relations in the AskUbuntu dataset. The parameters w are regularized as in the ingredient-product category setup.

The sequential part can therefore be viewed as a content recommendation task which is tailored to the specific query (e.g., ingredient) using features from previously trained RNNs. It assumes additional feedback in order to adjust the feature comparison using the introduced weights w.

5 Experimental Setup and Results

Ingredients: We use the *FoodEssentials LabelAPI*[2] and *Rapid Alert System for Food and Feed (RASFF)*[3] databases to extract 5439 ingredients and the product categories they appear in. On average, each ingredient appears in 16.3 product categories (out of 131 categories). We leverage Mechanical Turk to link each ingredient to the appropriate Wikipedia article. From the 5439 ingredients, there are 1680 unique Wikipedia articles. Each ingredient summary description has a median of 169 tokens.

AskUbuntu: The dataset consists of 167k questions and 16k user-marked similar question pairs taking from a 2014 dump of AskUbuntu website.

5.1 Training, development, and test sets

Ingredients: We take the set of unique Wikipedia articles and randomly split them into training, development, and test sets (60/20/20). We then assign the ingredients to the appropriate data set based on their Wikipedia articles. This is to ensure that the articles of the ingredients used in the development and test sets are not seen in training.

AskUbuntu: We take 8000 human annotated question pairs as our development and test sets. There are 200 query questions in each set. Each query question is paired with 20 candidate questions which are annotated as similar or non-similar. We evaluate by ranking these candidate questions.

5.2 Sequential scenario

Ingredients: Let n be the total number of labeled positive categories for the ingredient. We provide $min(20, n/2)$ positive categories for the sequential model to train. We also include k negative categories, where k is selected using the validation set. We evaluate the performance on the remaining $n - min(20, n/2)$ positive categories as well as on the negative categories not included in training.

[2] http://developer.foodessentials.com/

[3] http://ec.europa.eu/food/safety/rasff/index_en.htm

Ingredient	*Wikipedia article*	*Prediction 1*	*Prediction 2*	*Prediction 3*
oatmeal	Oatmeal	cereal (0.564)	snack, energy & granola bars (0.196)	breads & buns (0.039)
watermelon juice	Watermelon	fruit & vegetable juice (0.352)	ice cream & frozen yogurt (0.205)	yogurt (0.064)
jasmine rice	Jasmine rice	flavored rice dishes (0.294)	rice (0.237)	herbs & spices (0.062)
shrimp extract	Shrimp (food)	fish & seafood (0.491)	frozen dinners (0.128)	frozen appetizers (0.113)
meatball	Meatball	pizza (0.180)	breakfast sandwiches (0.128)	frozen dinners (0.120)
polysorbate 80	Polysorbate 80	chewing gum & mints (0.531)	candy (0.092)	baking decorations (0.049)
ketchup	Ketchup	ketchup (0.461)	salad dressing & mayonnaise (0.049)	other cooking sauces (0.044)
benzoic acid	Benzoic acid	powdered drinks (0.062)	fruit & vegetable juice (0.051)	candy (0.045)

Table 1: The three most likely food product category predictions generated by the baseline RNN model on eight unseen ingredients. The number in parenthesis represents the probability provided by the model.

AskUbuntu: We use the difference vectors in Equation 4 to compute the loss and sequentially update the feature weights w. Let n be the total number of labeled positive examples (similar questions). We select up to $n/2$ positive and negative examples. From the $n^2/4$ possible pairs, we select the 20 most informative pairs for training.

While we use the loss function commonly used for binary classification during training, we ultimately want to frame our question as a ranking problem. Therefore, after iterating through the initial observations, we compute the mean average precision (MAP) over the remaining (unseen) ingredients/questions and compare it to the MAP of the baseline RNN model on the same unseen examples.

5.3 Hyperparameters

RNN: We use Adam (Kingma and Ba, 2015) as the optimization method with the default setting suggested by the authors. We use a hidden dimension of $d = 50$ for the ingredients and $d = 400$ for the AskUbuntu questions. Additional parameters such as dropout (Hinton et al., 2012), hidden layers, regularization, stopping criteria, batch size, and learning rate is tuned on the development set.

Word Vectors: For the ingredient/product prediction task, we used the GloVe pre-trained vectors (Common Crawl, 42 billion tokens, 300-dimensional) (Pennington et al., 2014). The word vectors for the AskUbuntu vectors are pre-trained using the AskUbuntu and Wikipedia corpora.

Sequential: We utilize the bounded limited-memory BFGS algorithm (L-BFGS-B) (Byrd et al., 1995) to solve for the optimal feature weights with bounds $w_j \in [0.01, 2]$. We tuned the the constraint bounds and the regularization parameter λ on the development set.

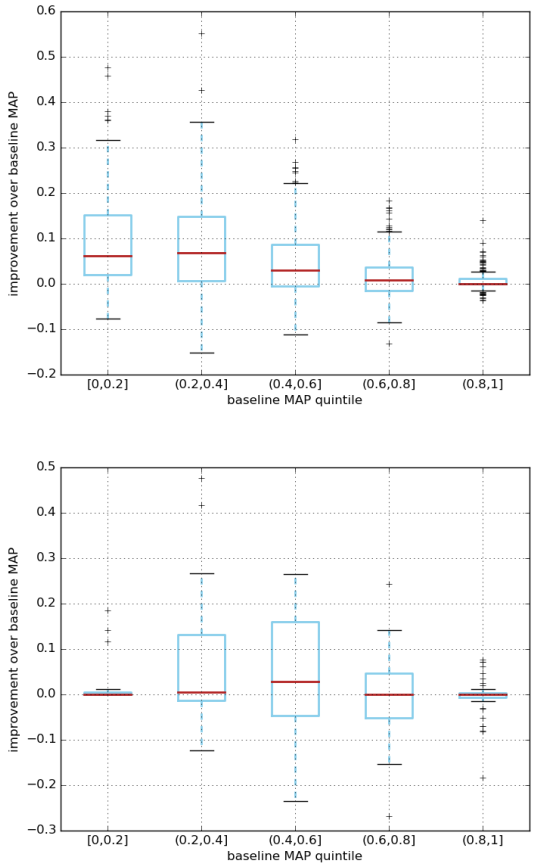


Figure 1: Box plot of the mean absolute mean average precision (MAP) improvement of the sequential model on the ingredients dataset (top) and AskUbuntu questions (bottom). They are divided into five quintiles based on the baseline RNN MAP score. The model shows gains in cases where the baseline RNN model's performance is poor or mediocre. The number of data points in each of the five quintiles of the ingredients dataset are, respectively: 131, 210, 240, 135, 191. For the AskUbuntu dataset, they are: 15, 26, 32, 40, 41.

	Ing / Dev	Ing / Test	AskUbuntu / Dev	AskUbuntu / Test
Mean MAP gain (percent)	0.0525 (30.9%)	0.0492 (26.5%)	0.0246 (8.2%)	0.0224 (7.5%)
Mean # positive observations	8.6	9.1	3.2	2.9

Table 2: We show the mean absolute improvement in the mean average precision (MAP) over the unobserved data points for each ingredient/question. The percent improvement shown is an average percent improvement across the ingredients/questions. They are the average of 100 runs per ingredient and 20 runs per AskUbuntu question.

Model	Validation set	Test set
Random	0.150 / 0.120	0.158 / 0.129
Baseline	0.320 / 0.291	0.331 / 0.300
MLP	0.432 / 0.390	0.459 / 0.416
RNN	**0.476 / 0.422**	**0.478 / 0.426**

Table 3: Results of the RNN model on the ingredient dataset, averaged across 5 runs. The two metrics shown are the mean average precision (MAP) and precision at N (P@N), where N is the total number of positive examples. The random model generates a random ranking of food categories for each ingredient. The baseline model uses the mean occurrence distribution of the food categories for all ingredients to rank the predictions. The multilayer perceptron model (MLP) is a three-layer neural network trained on the hierarchical properties of the input ingredients (extracted from the UMLS Metathesaurus). The RNN model outperforms all other baselines.

5.4 Results

Table 1 and 3 shows our results from using RNN to predict likely food product categories from Wikipedia text descriptions of ingredients.

We show the gains of the sequential update model in Table 2. We are able to generate consistent improvements in the MAP after seeing half of the observations. Box plots of the test set MAP improvements can be seen in Figure 1. For the ingredients prediction task, the sequential model offers the greatest improvements when the baseline RNN has low MAP. In the AskUbuntu questions, on the other hand, the positive effect is greatest when the baseline MAP is around 0.5.

There are three possible reasons for the difference in performance between the two tasks:

- The mean number of positive observations in the AskUbuntu task is 2.9, compared to 9.1 observations in the ingredients task (Table 2). This is a key factor in determining the sequential model's ability to tune for the optimal parameters. Having access to more annotated data would likely result in an increase in performance.

- Owing to the complexity of information encoded, the vectors for the AskUbuntu task are of dimension of 400 as opposed to 50 in the ingredients task. As a result, the sequential model would require more feedback to find near optimal weights w.

- We hypothesize that the sequential model leads to the most increased performance when the baseline model is mediocre. This is especially highlighted in the AskUbuntu task, as extremely poor performance indicate a complete mismatch of questions, while an exceptional performance leaves little room for additional improvement.

6 Conclusion

We demonstrated a text-based neural recommender approach to predict likely food products from a given ingredient as well as other similar questions from a given AskUbuntu question. We then extended this model to an online stream of new data, which improves over the off-line trained version for both of the two tasks tested. This sequential process improves model performance while requiring minimal additional training time and resources.

7 Acknowledgments

We thank the MIT NLP group and the reviewers for their helpful comments. The work was partially supported by the U.S. Food & Drug Administration, and by Google Faculty Award (Barzilay and Jaakkola). Any opinions, findings, conclusions, or recommendations expressed in the paper are those of the authors alone, and do not necessarily reflect the views of the funding organizations.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *International Conference on Machine Learning (ICML 2008)*.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grenfenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NIPS 2015)*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representation (ICLR 2015)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *International Conference on Machine Learning (ICML 2014)*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2015)*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. 2016. Semi-supervised question retrieval with recurrent convolutions. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Yangqiu Song and Dan Roth. 2015. Unsupervised sparse vector densification for short text similarity. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2015)*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural network. *Proceedings of the International Conference on Machine Learning (ICML 2011)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS 2014)*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction

Courtney Napoles,[1] **Keisuke Sakaguchi**,[1] and **Joel Tetreault**[2]
[1]Center for Language and Speech Processing, Johns Hopkins University
[2]Grammarly
{napoles,keisuke}@cs.jhu.edu, joel.tetreault@grammarly.com

Abstract

Current methods for automatically evaluating grammatical error correction (GEC) systems rely on gold-standard references. However, these methods suffer from penalizing grammatical edits that are correct but not in the gold standard. We show that reference-less grammaticality metrics correlate very strongly with human judgments and are competitive with the leading reference-based evaluation metrics. By interpolating both methods, we achieve state-of-the-art correlation with human judgments. Finally, we show that GEC metrics are much more reliable when they are calculated at the sentence level instead of the corpus level. We have set up a CodaLab site for benchmarking GEC output using a common dataset and different evaluation metrics.

1 Introduction

Grammatical error correction (GEC) has been evaluated by comparing the changes made by a system to the corrections made in gold-standard annotations. Following the recent shared tasks in this field (e.g., Ng et al. (2014)), several papers have critiqued GEC metrics and proposed new methods. Existing metrics depend on gold-standard corrections and therefore have a notable weakness: systems are penalized for making corrections that do not appear in the references.[1] For example, the following output has low metric scores even though three appropriate corrections were made to the input:

> However , people now can ~~contact~~ **communicate** with ~~anyone~~ **people** all over the world who can use computer**s** at any time , and there is no time delay of messages .

These changes (in red) were not seen in the references and therefore the metrics GLEU and M^2 (described in §2) score this output worse than 75% of 15,000 other generated sentences.

While grammaticality-based, reference-less metrics have been effective in estimating the quality of machine translation (MT) output, the utility of such metrics has not been investigated previously for GEC. We hypothesize that such methods can overcome this weakness in reference-based GEC metrics. This paper has four contributions: 1) We develop three *grammaticality* metrics that are competitive with current reference-based measures and correlate very strongly with human judgments. 2) We achieve state-of-the-art performance when interpolating a leading reference-based metric with a grammaticality metric. 3) We identify an interesting result that the mean of sentence-level scores is substantially better for evaluating systems than the system-level score. 4) We release code for two grammaticality metrics and establish an online platform for evaluating GEC output.

2 Prior work

To our knowledge, this is the first work to evaluate GEC without references. Within MT, this task is called *quality estimation*. MT output is evaluated by its *fluency*, or adherence to accepted conventions of grammaticality and style, and *adequacy*, which is the input's meaning conveyed in the output.

[1]We refer to the gold-standard corrections as *references* because *gold standard* suggests just one accurate correction.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2109–2115,
Austin, Texas, November 1-5, 2016.

Quality estimation targets fluency by estimating the amount of post-editing needed by the output. This has been the topic of recent shared tasks, e.g. Bojar et al. (2015). Specia et al. (2010) evaluated the quality of translations using sentence-level features from the output but not the references, predicting discrete and continuous scores. A strong baseline, *QuEst*, uses support vector regression trained over 17 features extracted from the output (Specia et al., 2013). Most closely related to our work, Parton et al. (2011) applied features from Educational Testing Service's e-rater® (Attali and Burstein, 2006) to evaluate MT output with a ranking SVM, without references, and improved performance by including features derived from MT metrics (BLEU, TERp, and METEOR).

Within the GEC field, recent shared tasks have prompted the development and scrutiny of new metrics for evaluating GEC systems. The Helping Our Own shared tasks evaluated systems using precision, recall, and F-score against annotated gold-standard corrections (Dale and Kilgarriff, 2011; Dale et al., 2012). The subsequent CoNLL Shared Tasks on GEC (Ng et al., 2013; Ng et al., 2014) were scored with the MaxMatch metric (M^2), which captures word- and phrase-level edits by calculating the F-score over an edit lattice (Dahlmeier and Ng, 2012). Felice and Briscoe (2015) identified shortcomings of M^2 and proposed I-measure to address these issues. I-measure computes the accuracy of a token-level alignment between the original, generated, and gold-standard sentences. These precision- and recall-based metrics measure fluency and adequacy by penalizing inappropriate changes, which alter meaning or introduce other errors. Changes consistent with the annotations indicate improved fluency and no change in meaning.

Unlike these metrics, GLEU scores output by penalizing n-grams found in the input and output but not the reference (Napoles et al., 2015). Like BLEU (Papineni et al., 2002), GLEU captures both fluency and adequacy with n-gram overlap. Recent work has shown that GLEU has the strongest correlation with human judgments compared to the GEC metrics described above (Sakaguchi et al., 2016). These GEC metrics are all defined at the corpus level, meaning that the statistics are accumulated over the entire output and then used to calculate a single system score.

3 Explicitly evaluating grammaticality

GLEU, I-measure, and M^2 are calculated based on comparison to reference corrections. These Reference-Based Metrics (*RBMs*) credit corrections seen in the references and penalize systems for ignoring errors and making bad changes (changing a span of text in an ungrammatical way or introducing errors to grammatical text). However, RBMs make two strong assumptions: that the annotations in the references are *correct* and that they are *complete*. We argue that these assumptions are invalid and point to a deficit in current evaluation practices. In GEC, the agreement between raters can be low due to the challenging nature of the task (Bryant and Ng, 2015; Rozovskaya and Roth, 2010; Tetreault and Chodorow, 2008), indicating that annotations may not be correct or complete.

An exhaustive list of all possible corrections would be time-consuming, if not impossible. As a result, RBMs penalize output that has a valid correction that is not present in the references or that addresses an error not corrected in the references. The example in §1 has low GLEU and M^2 scores, even though the output addresses two errors (GLEU=0.43 and $M^2 = 0.00$, in the bottom half and quartile of 15k system outputs, respectively).

To address these concerns, we propose three metrics to evaluate the grammaticality of output without comparing to the input or a gold-standard sentence (Grammaticality-Based Metrics, or *GBMs*). We expect GBMs to score sentences, such as our example in §1, more highly. The first two metrics are scored by counting the errors found by existing grammatical error detection tools. The error count score is simply calculated: $1 - \frac{\#\text{ errors}}{\#\text{ tokens}}$. Two different tools are used to count errors: e-rater®'s grammatical error detection modules (ER) and Language Tool (Miłkowski, 2010) (LT). We choose these because, while e-rater® is a large-scale, robust tool that detects more errors than Language Tool,[2] it is proprietary whereas Language Tool is publicly available and open sourced.

For our third method, we estimate a grammaticality score with a linguistic feature-based model (LFM), which is our implementation of Heilman et

[2]In the data used for this work, e-rater® detects approximately 15 times more errors than Language Tool.

al. (2014).[3] The LFM score is a ridge regression over a variety of linguistic features related to grammaticality, including the number of misspellings, language model scores, OOV counts, and PCFG and link grammar features. It has been shown to effectively assess the grammaticality of learner writing. LFM predicts a score for each sentence while ER and LT, like the RBMs, can be calculated with either sentence- or document-level statistics. To be consistent with LFM, for all metrics in this work we score each sentence individually and report the system score as the mean of the sentence scores. We discuss the effects of modifying metrics from a corpus-level to a sentence-level in §5.

Consistent with our hypothesis, ER and LT score the §1 example in the top quartile of outputs and LFM ranks it in the top half.

3.1 A hybrid metric

The obvious deficit of GBMs is that they do not measure the adequacy of generated sentences, so they could easily be manipulated with grammatical output that is unrelated to the input. An ideal GEC metric would measure both the grammaticality of a generated sentence and its meaning compared to the original sentence, and would not necessarily need references. The available data of scored system outputs are insufficient for developing a new metric due to their limited size, thus we turn to interpolation to develop a sophisticated metric that jointly captures grammaticality and adequacy.

To harness the advantage of RBMs (adequacy) and GBMs (fluency), we build combined metrics, interpolating each RBM with each GBM. For a sentence of system output, the interpolated score (S_{I}) of the GBM score (S_{G}) and RBM score (S_{R}) is computed as follows:

$$S_{\text{I}} = (1 - \lambda)S_{\text{G}} + \lambda S_{\text{R}}$$

All values of S_{G} and S_{R} are in the interval $[0, 1]$, except for I-measure, which falls between $[-1, 1]$, and the distribution varies for each metric.[4] The system score is the average S_{I} of all generated sentences.

[3] Our implementation is slightly modified in that it does not use features from the PET HPSG parser.

[4] Mean scores are GLEU 0.52 ± 0.21, M^2 0.21 ± 0.34, IM 0.10 ± 0.30, ER 0.91 ± 0.10, LFM 0.50 ± 0.16, LT 1.00 ± 0.01.

Metric	Spearman's ρ	Pearson's r
GLEU	**0.852**	**0.838**
ER	**0.852**	0.829
LT	0.808	0.811
I-measure	0.769	0.753
LFM	0.780	0.742
M^2	0.648	0.641

Table 1: Correlation between the human and metric rankings.

4 Experiments

To assess the proposed metrics, we apply the RBMs, GBMs, and interpolated metrics to score the output of 12 systems participating in the CoNLL-2014 Shared Task on GEC (Ng et al., 2014). Recent works have evaluated RBMs by collecting human rankings of these system outputs and comparing them to the metric rankings (Grundkiewicz et al., 2015; Napoles et al., 2015). In this section, we compare each metric's ranking to the human ranking of Grundkiewicz et al. (2015, Table 3c). We use 20 references for scoring with RBMs: 2 original references, 10 references collected by Bryant and Ng (2015), and 8 references collected by Sakaguchi et al. (2016). The motivations for using 20 references are twofold: the best GEC evaluation method uses these 20 references with the GLEU metric (Sakaguchi et al., 2016), and work in machine translation shows that more references are better for evaluation (Finch et al., 2004). Due to the low agreement discussed in §3, having more references can be beneficial for evaluating a system when there are multiple viable ways of correcting a sentence. Unlike previous GEC evaluations, all metrics reported here use the *mean* of the sentence-level scores for each system.

Results are presented in Table 1. The error-count metrics, ER and LT, have stronger correlation than all RBMs except for GLEU, which is the current state of the art. GLEU has the strongest correlation with the human ranking ($\rho = 0.852$, $r = 0.838$), followed closely by ER, which has slightly lower Pearson correlation ($r = 0.829$) but equally as strong rank correlation, which is arguably more important when comparing different systems. I-measure and LFM have similar strength correlations, and M^2 is the lowest performing metric, even though its correlation is still strong ($\rho = 0.648$, $r = 0.641$).

Next we compare the interpolated scores with the human ranking, testing 101 different values of λ

	ranked by Spearman's rank coefficient (ρ)				*ranked by Pearson's correlation coefficient (r)*			
		ER	**LFM**	**LT**		**ER**	**LFM**	**LT**
	no intrpl.	0.852 (0)	0.780 (0)	0.808 (0)	*no intrpl.*	0.829 (0)	0.742 (0)	0.811 (0)
GLEU	0.852 (1)	**0.885** (0.03)	0.874 (0.27)	0.857 (0.04)	0.838 (1)	**0.867** (0.27)	0.845 (0.84)	0.867 (0.09)
I-m.	0.769 (1)	0.874 (0.19)	0.863 (0.37)	0.852 (0.01)	0.753 (1)	0.837 (0.02)	0.791 (0.22)	0.828 (0.01)
M^2	0.648 (1)	0.868 (0.01)	0.852 (0.05)	0.808 (0.00)	0.641 (1)	0.829 (0.00)	0.754 (0.04)	0.811 (0.00)

Table 2: Oracle correlations between interpolated metrics and the human rankings. The λ value for each metric is in parentheses.

GLEU rank	**Intrpl. rank**	**Output sentence**
1	2	*Genectic* testing is a personal decision , with the knowledge that there is a *possiblity* that one could be a carrier or not .
2	3	*Genectic* testing is a personal decision , the *kowledge* that there is a *possiblity* that one could be a carrier or not .
3	1	Genetic testing is a personal decision , with the knowledge that there is a possibility that one could be a carrier or not .

Table 3: An example of system outputs ranked by GLEU and GLEU interpolated with ER. Words in italics are misspelled.

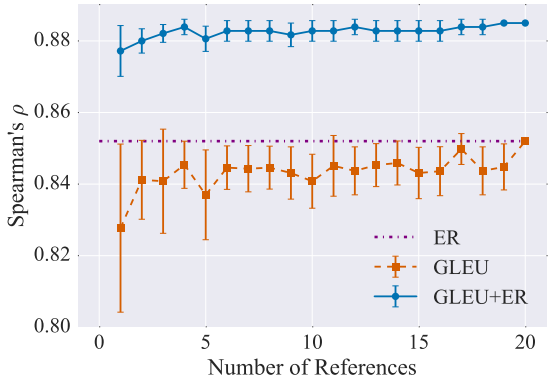


Figure 1: The mean correlation of oracle interpolated GLEU and ER scores across different reference sizes, compared to the uninterpolated metrics. Bars indicate a 95% confidence interval.

[0,1] to find the oracle value. Table 2 shows the correlations between the human judgments and the oracle interpolated metrics. Correlations of interpolated metrics are the upper bound and the correlations of uninterpolated metrics (in the first column and first row) are the lower bound. Interpolating GLEU and IM with GBMs has stronger correlation than any uninterpolated metric (i.e. $\lambda = 0$ or 1), and the oracle interpolation of ER and GLEU manifests the strongest correlation with the human ranking ($\rho = 0.885$, $r = 0.867$).[5] M^2 has the weakest correlation of all uninterpolated metrics and, when combined with GBMs, three of the interpolated metrics have $\lambda = 0$, meaning the interpolated score is equivalent to the GBM and M^2 does not contribute.

Table 3 presents an example of how interpolation can help evaluation. The top two sentences ranked by GLEU have misspellings that were not corrected in the NUCLE references. Interpolating with a GBM rightly ranks the misspelled output below the corrected output.

Since these experiments use a large number of references (20), we determine how different reference sizes affect the interpolated metrics by systematically increasing the number of references from 1 to 20 and randomly choosing n references to use as a gold standard when calculating the RBM scores, repeating 10 times for each value n (Figure 1). The correlation is nearly as strong with 3 and 20 references ($\rho = 0.884$ v. 0.885), and interpolating with just 1 reference is nearly as good (0.878) and improves over any uninterpolated metric.

We acknowledge that using GBMs is in effect using GEC systems to score other GEC systems. Interestingly, we find that even if the GBMs are imperfect, they still boost performance of the RBMs. GBMs have been trained to recognize errors in different contexts and, conversely, can identify correct grammatical constructions in diverse contexts, where the RBMs only recognize corrections made that appear in the gold standards, which are limited. Therefore GBMs can make a nice complement to shortcomings that RBMs may have.

5 Sentence-level evaluation

In the course of our experiments, we noticed that I-measure and GLEU have stronger correlations with the expert human ranking when using the

[5]To verify that these metrics cannot be gamed, we interpolated GBMs with RBMs scored against randomized references, and got scores 15% lower than un-gamed scores, on average.

	Corpus		Sentence	
Metric	ρ	r	ρ	r
GLEU	0.725	0.724	0.852	0.838
I-m.	-0.055*	0.061	0.769	0.753
M^2	0.692	0.617	0.648	0.641

Table 4: Correlation with human ranking when using corpus- and sentence-level metrics. * indicates a significant difference from the corresponding sentence-level correlation ($p < 0.05$).[7]

mean sentence-level score (Table 4).[6] Most notably, I-measure does not correlate at all as a corpus-level metric but has a very strong correlation at the sentence-level (specifically, ρ improves from -0.055 to 0.769). This could be because corpus-level statistics equally distribute counts of correct annotations over all sentences, even those where the output neglects to make any necessary corrections. Sentence-level statistics would not average the correct counts over all sentences in this same way. As a result, a corpus-level statistic may over-estimate the quality of system output. Deeper investigation into this phenomenon is needed to understand why the mean sentence-level scores do better.

6 Summary

We have identified a shortcoming of reference-based metrics: they penalize changes made that do not appear in the references, even if those changes are acceptable. To address this problem, we developed metrics to explicitly measure grammaticality without relying on reference corrections and showed that the error-count metrics are competitive with the best reference-based metric. Furthermore, by interpolating RBMs with GBMs, the system ranking has even stronger correlation with the human ranking. The ER metric, which was derived from counts of errors detected using e-rater®, is nearly as good as the state-of-the-art RBM (GLEU) and the interpolation of these metrics has the strongest reported correlation with the human ranking ($\rho = 0.885$, $r = 0.867$). However, since e-rater® is not widely available to researchers, we also tested a metric using Language Tool, which does nearly as well when interpolated with GLEU ($\rho = 0.857$, $r = 0.867$).

Two important points should be noted: First, due to the small sample size (12 system outputs), none of the rankings significantly differ from one another except for the corpus-level I-measure. Secondly, GLEU and the other RBMs already have strong to very strong correlation with the human judgments, which makes it difficult for any combination of metrics to perform substantially higher. The best uninterpolated and interpolated metrics use an extremely large number of references (20), however Figure 1 shows that interpolating GLEU using just one reference has stronger correlation than any uninterpolated metric. This supports the use of interpolation to improve GEC evaluation in any setting.

This work is the first exploration into applying fluency-based metrics to GEC evaluation. We believe that, for future work, fluency measures could be further improved with other methods, such as using existing GEC systems to *detect* errors, or even using an ensemble of systems to improve coverage (indeed, ensembles have been useful in the GEC task itself (Susanto et al., 2014)). There is also recent work from the MT community, such as the use of confidence bounds (Graham and Liu, 2016) or uncertainty measurement (Beck et al., 2016), which could be adopted by the GEC community.

Finally, in the course of our experiments, we determined that metrics calculated on the sentence-level is more reliable for evaluating GEC output, and we suggest that the GEC community adopt this modification to better assess systems.

To facilitate GEC evaluation, we have set up an online platform[8] for benchmarking system output on the same set of sentences evaluated using different metrics and made the code for calculating LT and LFM available.[9]

Acknowledgments

We would like to thank Matt Post, Martin Chodorow, and the three anonymous reviews for their comments and feedback. We also thank Educational Testing Service for providing e-rater® output. This material is based upon work partially supported by the NSF GRF under Grant No. 1232825.

[6] The correlations in Table 4 differ from what was reported in Grundkiewicz et al. (2015) and Napoles et al. (2015) due to the references and sentence-level computation used in this work.

[7] Significance is found by applying a two-tailed t-test to the Z-scores attained using Fisher's z-transformation.

[8] https://competitions.codalab.org/competitions/12731

[9] https://github.com/cnap/grammaticality-metrics

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Daniel Beck, Lucia Specia, and Trevor Cohn. 2016. Exploring prediction uncertainty in machine translation quality estimation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 208–218, Berlin, Germany, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, Beijing, China, July. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 242–249, Nancy, France, September. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada, June. Association for Computational Linguistics.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 578–587, Denver, Colorado, June. Association for Computational Linguistics.

Andrew M. Finch, Yasuhiro Akiba, and Eiichiro Sumita. 2004. How does automatic machine translation evaluation correlate with human scoring as the number of reference translations increases? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*.

Yvette Graham and Qun Liu. 2016. Achieving accurate conclusions in evaluation of automatic machine translation metrics. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, San Diego, California, June. Association for Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. 2015. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 461–470, Lisbon, Portugal, September. Association for Computational Linguistics.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 174–180, Baltimore, Maryland, June. Association for Computational Linguistics.

Marcin Miłkowski. 2010. Developing an open-source, rule-based proofreading tool. *Software: Practice and Experience*, 40(7):543–566.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China, July. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Kristen Parton, Joel Tetreault, Nitin Madnani, and Martin Chodorow. 2011. E-rating machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 108–115. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2010. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine translation*, 24(1):39–50.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst – a translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria, August. Association for Computational Linguistics.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 951–962, Doha, Qatar, October. Association for Computational Linguistics.

Joel Tetreault and Martin Chodorow. 2008. Native judgments of non-native usage: Experiments in preposition error detection. In *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*, pages 24–32, Manchester, UK, August. Coling 2008 Organizing Committee.

Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change

William L. Hamilton, Jure Leskovec, Dan Jurafsky
Department of Computer Science, Stanford University, Stanford CA, 94305
wleif,jure,jurafsky@stanford.edu

Abstract

Words shift in meaning for many reasons, including cultural factors like new technologies and regular linguistic processes like subjectification. Understanding the evolution of language and culture requires disentangling these underlying causes. Here we show how two different distributional measures can be used to detect two different types of semantic change. The first measure, which has been used in many previous works, analyzes global shifts in a word's distributional semantics; it is sensitive to changes due to regular processes of linguistic drift, such as the semantic generalization of *promise* ("I promise."→"It promised to be exciting."). The second measure, which we develop here, focuses on local changes to a word's nearest semantic neighbors; it is more sensitive to cultural shifts, such as the change in the meaning of *cell* ("prison cell" → "cell phone"). Comparing measurements made by these two methods allows researchers to determine whether changes are more cultural or linguistic in nature, a distinction that is essential for work in the digital humanities and historical linguistics.

1 Introduction

Distributional methods of embedding words in vector spaces according to their co-occurrence statistics are a promising new tool for diachronic semantics (Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kulkarni et al., 2014; Xu and Kemp, 2015; Hamilton et al., 2016). Previous work, however, does not consider the underlying causes of semantic change or how to distentangle different types of change.

We show how two computational measures can be used to distinguish between semantic changes caused by cultural shifts (e.g., technological advancements) and those caused by more regular processes of semantic change (e.g., grammaticalization or subjectification). This distinction is essential for research on linguistic and cultural evolution. Detecting cultural shifts in language use is crucial to computational studies of history and other digital humanities projects. By contrast, for advancing historical linguistics, cultural shifts amount to noise and only the more regular shifts matter.

Our work builds on two intuitions: that distributional models can highlight syntagmatic versus paradigmatic relations with neighboring words (Schutze and Pedersen, 1993) and that nouns are more likely to undergo changes due to irregular cultural shifts while verbs more readily participate in regular processes of semantic change (Gentner and France, 1988; Traugott and Dasher, 2001). We use this noun vs. verb mapping as a proxy to compare our two measures' sensitivities to cultural vs. linguistic shifts. Sensitivity to nominal shifts indicates a propensity to capture irregular cultural shifts in language, such as those due to technological advancements (Traugott and Dasher, 2001). Sensitivity to shifts in verbs (and other predicates) indicates a propensity to capture regular processes of linguistic drift (Gentner and France, 1988; Kintsch, 2000; Traugott and Dasher, 2001).

The first measure we analyze is based upon changes to a word's local semantic neighborhood;

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2116–2121,
Austin, Texas, November 1-5, 2016.

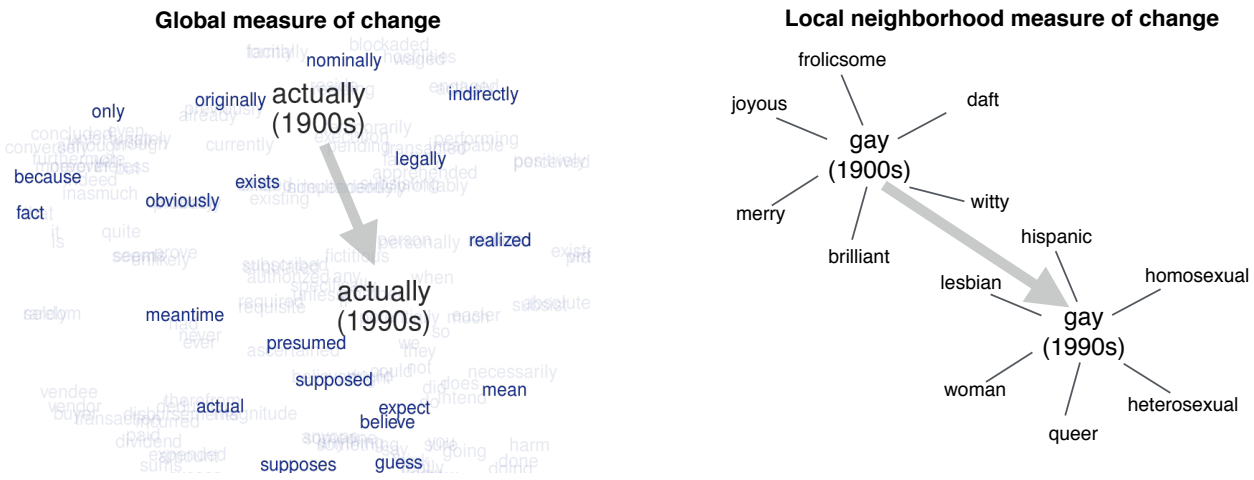


Figure 1: Two different measures of semantic change. With the global measure of change, we measure how far a word has moved in semantic space between two time-periods. This measure is sensitive to subtle shifts in usage and also global effects due to the entire semantic space shifting. For example, this captures how *actually* underwent subjectification during the 20th century, shifting from uses in objective statements about the world ("actually did try") to subjective statements of attitude ("I actually agree"; see Traugott and Dasher, 2001 for details). In contrast, with the local neighborhood measure of change, we measure changes in a word's nearest neighbors, which captures drastic shifts in core meaning, such as *gay*'s shift in meaning over the 20th century.

we show that it is more sensitive to changes in the nominal domain and captures changes due to unpredictable cultural shifts. Our second measure relies on a more traditional global notion of change; we show that it better captures changes, like those in verbs, that are the result of regular linguistic drift.

Our analysis relies on a large-scale statistical study of six historical corpora in multiple languages, along with case-studies that illustrate the fine-grained differences between the two measures.

2 Methods

We use the diachronic word2vec embeddings constructed in our previous work (Hamilton et al., 2016) to measure how word meanings change between consecutive decades.[1] In these representations each word w_i has a vector representation $\mathbf{w}^{(t)}$ (Turney and Pantel, 2010) at each time point, which captures its co-occurrence statistics for that time period. The vectors are constructed using the skip-gram with negative sampling (SGNS) algorithm (Mikolov et al., 2013) and post-processed to align the semantic spaces between years. Measuring the distance between word vectors for consecutive decades allows us to compute the rate at which the different words change in meaning (Gulordava and Baroni, 2011).

We analyzed the decades from 1800 to 1990 using vectors derived from the Google N-gram datasets (Lin et al., 2012) that have large amounts of historical text (English, French, German, and English Fiction). We also used vectors derived from the Corpus of Historical American English (COHA), which is smaller than Google N-grams but was carefully constructed to be genre balanced and contains word lemmas as well as surface forms (Davies, 2010). We examined all decades from 1850 through 2000 using the COHA dataset and used the part-of-speech tags provided with the corpora.

2.1 Measuring semantic change

We examine two different ways to measure semantic change (Figure 1).

Global measure

The first measure analyzes global shifts in a word's vector semantics and is identical to the measure used in most previous works (Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kim et al., 2014; Hamilton et al., 2016). We simply take a word's vectors for two consecutive decades and measure the cosine distance between them, i.e.

$$d^G(w_i^{(t)}, w_i^{(t+1)}) = \text{cos-dist}(\mathbf{w}_i^{(t)}, \mathbf{w}_i^{(t+1)}). \quad (1)$$

[1] http://nlp.stanford.edu/projects/histwords/. This URL also links to detailed dataset descriptions and the code needed to replicate the experiments in this paper.

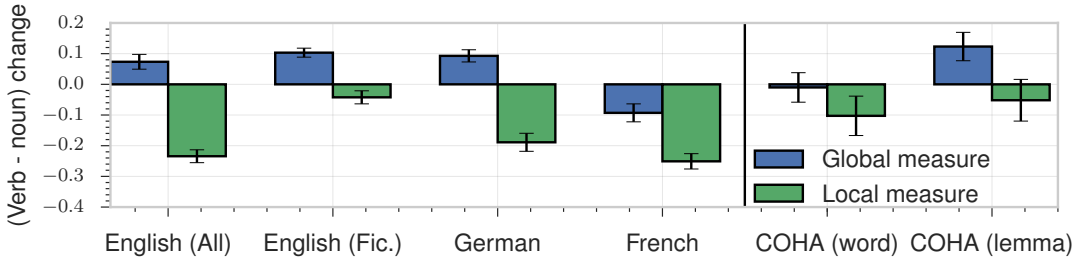


Figure 2: The global measure is more sensitive to semantic changes in verbs while the local neighborhood measure is more sensitive to noun changes. Examining how much nouns change relative to verbs (using coefficients from mixed-model regressions) reveals that the two measures are sensitive to different types of semantic change. Across all languages, the local neighborhood measure always assigns relatively higher rates of change to nouns (i.e., the right/green bars are lower than the left/blue bars for all pairs), though the results vary by language (e.g., French has high noun change-rates overall). 95% confidence intervals are shown.

Local neighborhood measure

The second measure is based on the intuition that only a word's nearest semantic neighbors are relevant. For this measure, we first find word w_i's set of k nearest-neighbors (according to cosine-similarity) within each decade, which we denote by the ordered set $\mathcal{N}_k(w_i^{(t)})$. Next, to measure the change between decades t and $t+1$, we compute a "second-order" similarity vector for $w_i^{(t)}$ from these neighbor sets with entries defined as

$$\mathbf{s}^{(t)}(j) = \text{cos-sim}(\mathbf{w}_i^{(t)}, \mathbf{w}_j^{(t)}) \quad \forall w_j \in \mathcal{N}_k(w_i^{(t)}) \cup \mathcal{N}_k(w_i^{(t+1)}), \quad (2)$$

and we compute an analogous vector for $w_i^{(t+1)}$. The second-order vector, $\mathbf{s}_i^{(t)}$, contains the cosine similarity of $\mathbf{w}_i$ and the vectors of all w_i's nearest semantic neighbors in the the time-periods t and $t+1$. Working with variants of these second-order vectors has been a popular approach in many recent works, though most of these works define these vectors against the full vocabulary and not just a word's nearest neighbors (del Prado Martin and Brendel, 2016; Eger and Mehler, 2016; Rodda et al., 2016).

Finally, we compute the local neighborhood change as

$$d^L(w_i^{(t)}, w_i^{(t+1)}) = \text{cos-dist}(\mathbf{s}_i^{(t)}, \mathbf{s}_i^{(t+1)}). \quad (3)$$

This measures the extent to which w_i's similarity with its nearest neighbors has changed.

The local neighborhood measure defined in (3) captures strong shifts in a word's paradigmatic relations but is less sensitive to global shifts in syntagmatic contexts (Schutze and Pedersen, 1993). We used $k = 25$ in all experiments (though we found the results to be consistent for $k \in [10, 50]$).

Dataset	# Nouns	# Verbs
Google English All	5299	2722
Google English Fic.	4941	3128
German	5443	1844
French	2310	4992
COHA (Word)	4077	1267
COHA (Lemma)	3389	783

Table 1: Number of nouns and verbs tested in each dataset.

2.2 Statistical methodology

To test whether nouns or verbs change more according to our two measures of change, we build on our previous work and used a linear mixed model approach (Hamilton et al., 2016). This approach amounts to a linear regression where the model also includes "random" effects to account for the fact that the measurements for individual words will be correlated across time (McCulloch and Neuhaus, 2001).

We ran two regressions per datatset: one with the global d^G values as the dependent variables (DVs) and one with the local neighborhood d^L values. In both cases we examined the change between all consecutive decades and normalized the DVs to zero-mean and unit variance. We examined nouns/verbs within the top-10000 words by frequency rank and removed all words that occurred <500 times in the smaller COHA dataset. The independent variables are word frequency, the decade of the change (represented categorically), and variable indicating

Word	1850s context	1990s context
actually	"...dinners which you have actually eaten."	"With that, I actually agree."
must	"O, George, we must have faith."	"Which you must have heard ten years ago..."
promise	"I promise to pay you...'	"...the day promised to be lovely."
gay	"Gay bridals and other merry-makings of men."	"...the result of gay rights demonstrations."
virus	"This young man is...infected with the virus."	"...a rapidly spreading computer virus."
cell	"The door of a gloomy cell..."	"They really need their cell phones."

Table 2: Example case-studies of semantic change. The first three words are examples of regular linguistic shifts, while the latter three are examples of words that shifted due to exogenous cultural factors. Contexts are from the COHA data (Davies, 2010).

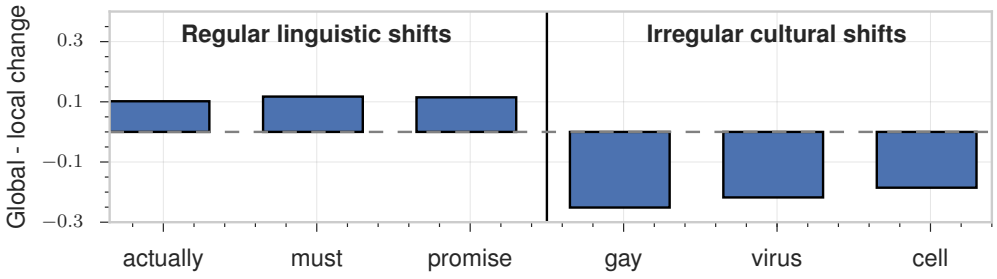


Figure 3: The global measure captures classic examples of linguistic drift while the local measure captures example cultural shifts. Examining the semantic distance between the 1850s and 1990s shows that the global measure is more sensitive to regular shifts (and vice-versa for the local measure). The plot shows the difference between the measurements made by the two methods.

whether a word is a noun or a verb (proper nouns are excluded, as in Hamilton et al., 2016).[2]

3 Results

Our results show that the two seemingly related measures actually result in drastically different notions of semantic change.

3.1 Nouns vs. verbs

The local neighborhood measure assigns far higher rates of semantic change to nouns across all languages and datasets while the opposite is true for the global distance measure, which tends to assign higher rates of change to verbs (Figure 2).

We focused on verbs vs. nouns since they are the two major parts-of-speech and previous research has shown that verbs are more semantically mutable than nouns and thus more likely to undergo linguistic drift (Gentner and France, 1988), while nouns are far more likely to change due to cultural shifts like new technologies (Traugott and Dasher, 2001). However, some well-known regular linguistic shifts include rarer parts of speech like adverbs (included in our case studies below). Thus we also confirmed that the differences shown in Figure 2 also hold when adverbs and adjectives are included along with the verbs. This modified analysis showed analogous significant trends, which fits with previous research arguing that adverbial and adjectival modifiers are also often the target of regular linguistic changes (Traugott and Dasher, 2001).

The results of this large-scale regression analysis show that the local measure is more sensitive to changes in the nominal domain, a domain in which change is known to be driven by cultural factors. In contrast, the global measure is more sensitive to changes in verbs, along with adjectives and adverbs, which are known to be the targets of many regular processes of linguistic change (Traugott and Dasher, 2001; Hopper and Traugott, 2003)

3.2 Case studies

We examined six case-study words grouped into two sets. These case studies show that three examples of well-attested regular linguistic shifts (set A) changed more according to the global measure, while three well-known examples of cultural changes (set B) change more according to the local neighborhood measure. Table 2 lists these words with some representative historical contexts (Davies, 2010).

[2]Frequency was included since it is known to strongly influence the distributional measures (Hamilton et al., 2016).

Set A contains three words that underwent attested regular linguistic shifts detailed in Traugott and Dasher (2001): *actually*, *must*, and *promise*. These three words represent three different types of regular linguistic shifts: *actually* is a case of subjectification (detailed in Figure 1); *must* shifted from a deontic/obligation usage ("you must do X") to a epistemic one ("X must be the case"), exemplifying a regular pattern of change common to many modal verbs; and *promise* represents the class of shifting "performative speech acts" that undergo rich changes due to their pragmatic uses and subjectification (Traugott and Dasher, 2001). The contexts listed in Table 2 exemplify these shifts.

Set B contains three words that were selected because they underwent well-known cultural shifts over the last 150 years: *gay*, *virus*, and *cell*. These words gained new meanings due to uses in community-specific vernacular (*gay*) or technological advances (*virus*, *cell*). The cultural shifts underlying these changes in usage — e.g., the development of the mobile "cell phone" — were unpredictable in the sense that they were not the result of regularities in human linguistic systems.

Figure 3 shows how much the meaning of these word changed from the 1850s to the 1990s according to the two different measures on the English Google data. We see that the words in set A changed more when measurements were made using the global measure, while the opposite holds for set B.

4 Discussion

Our results show that our novel local neighborhood measure of semantic change is more sensitive to changes in nouns, while the global measure is more sensitive to changes in verbs. This mapping aligns with the traditional distinction between irregular cultural shifts in nominals and more regular cases of linguistic drift (Traugott and Dasher, 2001) and is further reinforced by our six case studies.

This finding emphasizes that researchers must develop and use measures of semantic change that are tuned to specific tasks. For example, a cultural change-point detection framework would be more successful using our local neighborhood measure, while an empirical study of grammaticalization would be better off using the traditional global distance approach. Comparing measurements made by these two approaches also allows researchers to assess the extent to which semantic changes are linguistic or cultural in nature.

Acknowledgements

The authors thank C. Manning, V. Prabhakaran, S. Kumar, and our anonymous reviewers for their helpful comments. This research has been supported in part by NSF CNS-1010921, IIS-1149837, IIS-1514268 NIH BD2K, ARO MURI, DARPA XDATA, DARPA SIMPLEX, Stanford Data Science Initiative, SAP Stanford Graduate Fellowship, NSERC PGS-D, Boeing, Lightspeed, and Volkswagen.

References

Mark Davies. 2010. The Corpus of Historical American English: 400 million words, 1810-2009. http://corpus.byu.edu/coha/.

Fermin Moscoso del Prado Martin and Christian Brendel. 2016. Case and Cause in Icelandic: Reconstructing Causal Networks of Cascaded Language Changes. In *Proc. ACL*.

Steffen Eger and Alexander Mehler. 2016. On the Linearity of Semantic Change: Investigating Meaning Variation via Dynamic Graph Models. In *Proc. ACL*.

Dedre Gentner and Ilene M. France. 1988. The verb mutability effect: Studies of the combinatorial semantics of nouns and verbs. *Lexical ambiguity resolution: Perspectives from psycholinguistics, neuropsychology, and artificial intelligence*, pages 343–382.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proc. GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proc. ACL*.

Paul J Hopper and Elizabeth Closs Traugott. 2003. *Grammaticalization*. Cambridge University Press, Cambridge, UK.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proc. 14th ACM/IEEE-CS Conf. on Digital Libraries*, pages 229–238. IEEE Press.

Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515*.

Walter Kintsch. 2000. Metaphor comprehension: A computational theory. *Psychon. Bull. Rev.*, 7(2):257–266.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2014. Statistically significant detection of linguistic change. In *Proc. 24th WWW Conf.*, pages 625–635.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proc. ACL System Demonstrations*.

Charles E McCulloch and John M Neuhaus. 2001. *Generalized linear mixed models*. Wiley-Interscience, Hoboken, NJ.

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

Martina Rodda, Marco Senaldi, and Alessandro Lenci. 2016. Panta rei: Tracking Semantic Change with Distributional Semantics in Ancient Greek. In *Italian Conference of Computational Linguistics*.

Hinrich Schutze and Jan Pedersen. 1993. A vector model for syntagmatic and paradigmatic relatedness. In *Proc. 9th Annu. Conf. of the UW Centre for the New OED and Text Research*, pages 104–113. Citeseer.

Elizabeth Closs Traugott and Richard B Dasher. 2001. *Regularity in Semantic Change*. Cambridge University Press, Cambridge, UK.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.*, 37(1):141–188.

Yang Xu and Charles Kemp. 2015. A computational evaluation of two laws of semantic change. In *Proc. 37th Annu. Conf. Cogn. Sci. Soc.*

How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation

Chia-Wei Liu[1]*, **Ryan Lowe**[1]*, **Iulian V. Serban**[2]*, **Michael Noseworthy**[1]*,
Laurent Charlin[1], **Joelle Pineau**[1]
[1] School of Computer Science, McGill University
{chia-wei.liu,ryan.lowe,michael.noseworthy}@mail.mcgill.ca
{lcharlin, jpineau}@cs.mcgill.ca
[2] DIRO, Université de Montréal
iulian.vlad.serban@umontreal.ca

Abstract

We investigate evaluation metrics for dialogue response generation systems where supervised labels, such as task completion, are not available. Recent works in response generation have adopted metrics from machine translation to compare a model's generated response to a single target response. We show that these metrics correlate very weakly with human judgements in the non-technical Twitter domain, and not at all in the technical Ubuntu domain. We provide quantitative and qualitative results highlighting specific weaknesses in existing metrics, and provide recommendations for future development of better automatic evaluation metrics for dialogue systems.

1 Introduction

An important aspect of dialogue response generation systems, which are trained to produce a reasonable utterance given a conversational context, is how to evaluate the quality of the generated response. Typically, evaluation is done using human-generated supervised signals, such as a task completion test or a user satisfaction score (Walker et al., 1997; Möller et al., 2006; Kamm, 1995), which are relevant when the dialogue is task-focused. We call models optimized for such supervised objectives *supervised dialogue models*, while those that do not are *unsupervised dialogue models*.

This paper focuses on unsupervised dialogue response generation models, such as chatbots. These models are receiving increased attention, particularly using end-to-end training with neural networks (Serban et al., 2016; Sordoni et al., 2015; Vinyals and Le, 2015). This avoids the need to collect supervised labels on a large scale, which can be prohibitively expensive. However, automatically evaluating the quality of these models remains an open question. Automatic evaluation metrics would help accelerate the deployment of unsupervised response generation systems.

Faced with similar challenges, other natural language tasks have successfully developed automatic evaluation metrics. For example, BLEU (Papineni et al., 2002a) and METEOR (Banerjee and Lavie, 2005) are now standard for evaluating machine translation models, and ROUGE (Lin, 2004) is often used for automatic summarization. These metrics have recently been adopted by dialogue researchers (Ritter et al., 2011; Sordoni et al., 2015; Li et al., 2015; Galley et al., 2015b; Wen et al., 2015; Li et al., 2016). However these metrics assume that valid responses have significant word overlap with the ground truth responses. This is a strong assumption for dialogue systems, where there is significant diversity in the space of valid responses to a given context. This is illustrated in Table 1, where two reasonable responses are proposed to the context, but these responses do not share any words in common and do not have the same semantic meaning.

In this paper, we investigate the correlation between the scores from several automatic evaluation metrics and human judgements of dialogue response quality, for a variety of response generation models. We consider both statistical *word-overlap similar-*

* Denotes equal contribution.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2122–2132,
Austin, Texas, November 1-5, 2016.

Context of Conversation
Speaker A: Hey John, what do you want to do tonight?
Speaker B: Why don't we go see a movie?
Ground-Truth Response
Nah, I hate that stuff, let's do something active.
Model Response
Oh sure! Heard the film about Turing is out!

Table 1: Example showing the intrinsic diversity of valid responses in a dialogue. The (reasonable) model response would receive a BLEU score of 0.

ity metrics such as BLEU, METEOR, and ROUGE, and *word embedding metrics* derived from word embedding models such as Word2Vec (Mikolov et al., 2013). We find that all metrics show either weak or no correlation with human judgements, despite the fact that word overlap metrics have been used extensively in the literature for evaluating dialogue response models (see above, and Lasguido et al. (2014)). In particular, we show that these metrics have only a small positive correlation on the chitchat oriented Twitter dataset, and no correlation at all on the technical Ubuntu Dialogue Corpus. For the word embedding metrics, we show that this is true even though all metrics are able to significantly distinguish between baseline and state-of-the-art models across multiple datasets. We further highlight the shortcomings of these metrics using: a) a statistical analysis of our survey's results; b) a qualitative analysis of examples from our data; and c) an exploration of the sensitivity of the metrics.

Our results indicate that a shift must be made in the research community away from these metrics, and highlight the need for a new metric that correlates more strongly with human judgement.

2 Related Work

We focus on metrics that are *model-independent*, i.e. where the model generating the response does not also evaluate its quality; thus, we do not consider word perplexity, although it has been used to evaluate unsupervised dialogue models (Serban et al., 2015). This is because it is not computed on a per-response basis, and cannot be computed for retrieval models. Further, we only consider metrics that can be used to evaluate proposed responses against ground-truth responses, so we do not consider retrieval-based metrics such as recall, which has been used to evaluate dialogue models (Schatzmann et al., 2005; Lowe et al., 2015). We also do not consider evaluation methods for supervised evaluation methods.[1]

Several recent works on unsupervised dialogue systems adopt the BLEU score for evaluation. Ritter et al. (2011) formulate the unsupervised learning problem as one of translating a context into a candidate response. They use a statistical machine translation (SMT) model to generate responses to various contexts using Twitter data, and show that it outperforms information retrieval baselines according to both BLEU and human evaluations. Sordoni et al. (2015) extend this idea using a recurrent language model to generate responses in a context-sensitive manner. They also evaluate using BLEU, however they produce multiple ground truth responses by retrieving 15 responses from elsewhere in the corpus, using a simple bag-of-words model. Li et al. (2015) evaluate their proposed diversity-promoting objective function for neural network models using BLEU score with only a single ground truth response. A modified version of BLEU, deltaBLEU (Galley et al., 2015b), which takes into account several human-evaluated ground truth responses, is shown to have a weak to moderate correlation to human judgements using Twitter dialogues. However, such human annotation is often infeasible to obtain in practice. Galley et al. (2015b) also show that, even with several ground truth responses available, the standard BLEU metric does not correlate strongly with human judgements.

There has been significant previous work that evaluates how well automatic metrics correlate with human judgements in in both machine translation (Callison-Burch et al., 2010; Callison-Burch et al., 2011; Bojar et al., 2014; Graham et al., 2015) and natural language generation (NLG) (Stent et al., 2005; Cahill, 2009; Reiter and Belz, 2009; Espinosa et al., 2010). There has also been work criticizing the usefulness of BLEU in particular for machine translation (Callison-Burch et al., 2006). While many of the criticisms in these works apply to dialogue generation, we note that generating dialogue responses conditioned on the conversational

[1] Evaluation methods in the supervised setting have been well studied, see (Walker et al., 1997; Möller et al., 2006; Jokinen and McTear, 2009).

context is in fact a more difficult problem. This is because most of the difficulty in automatically evaluating language generation models lies in the large set of correct answers. Dialogue response generation given solely the context intuitively has a higher diversity (or entropy) than translation given text in a source language, or surface realization given some intermediate form (Artstein et al., 2009).

3 Evaluation Metrics

Given a dialogue context and a proposed response, our goal is to automatically evaluate how appropriate the proposed response is to the conversation. We focus on metrics that compare it to the ground truth response of the conversation. In particular, we investigate two approaches: word based similarity metrics and word-embedding based similarity metrics.

3.1 Word Overlap-based Metrics

We first consider metrics that evaluate the amount of *word-overlap* between the proposed response and the ground-truth response. We examine the BLEU and METEOR scores that have been used for machine translation, and the ROUGE score that has been used for automatic summarization. While these metrics have been shown to correlate with human judgements in their target domains (Papineni et al., 2002a; Lin, 2004), they have not been thoroughly investigated for dialogue systems.[2]

We denote the ground truth response as r (thus we assume that there is a single candidate ground truth response), and the proposed response as $\hat{r}$. The j'th token in the ground truth response r is denoted by w_j, with $\hat{w}_j$ denoting the j'th token in the proposed response $\hat{r}$.

BLEU. BLEU (Papineni et al., 2002a) analyzes the co-occurrences of n-grams in the ground truth and the proposed responses. It first computes an n-gram precision for the whole dataset (we assume that there is a single candidate ground truth response per context):

$$P_n(r, \hat{r}) = \frac{\sum_k \min(h(k, r), h(k, \hat{r}_i))}{\sum_k h(k, r_i)}$$

where k indexes all possible n-grams of length n and $h(k, r)$ is the number of n-grams k in r.[3] To avoid the drawbacks of using a precision score, namely that it favours shorter (candidate) sentences, the authors introduce a brevity penalty. BLEU-N, where N is the maximum length of n-grams considered, is defined as:

$$\text{BLEU-N} := b(r, \hat{r}) \exp(\sum_{n=1}^{N} \beta_n \log P_n(r, \hat{r}))$$

β_n is a weighting that is usually uniform, and $b(\cdot)$ is the brevity penalty. The most commonly used version of BLEU uses $N = 4$. Modern versions of BLEU also use sentence-level smoothing, as the geometric mean often results in scores of 0 if there is no 4-gram overlap (Chen and Cherry, 2014). Note that BLEU is usually calculated at the corpus-level, and was originally designed for use with multiple reference sentences.

METEOR. The METEOR metric (Banerjee and Lavie, 2005) was introduced to address several weaknesses in BLEU. It creates an explicit alignment between the candidate and target responses. The alignment is based on exact token matching, followed by WordNet synonyms, stemmed tokens, and then paraphrases. Given a set of alignments, the METEOR score is the harmonic mean of precision and recall between the proposed and ground truth sentence.

ROUGE. ROUGE (Lin, 2004) is a set of evaluation metrics used for automatic summarization. We consider ROUGE-L, which is a F-measure based on the Longest Common Subsequence (LCS) between a candidate and target sentence. The LCS is a set of words which occur in two sentences in the same order; however, unlike n-grams the words do not have to be contiguous, i.e. there can be other words in between the words of the LCS.

[2] To the best of our knowledge, only BLEU has been evaluated in the dialogue system setting quantitatively by Galley et al. (2015a) on the Twitter domain. However, they carried out their experiments in a very different setting with multiple ground truth responses, which are rarely available in practice, and without providing any qualitative analysis of their results.

[3] Note that the min in this equation is calculating the number of co-occurrences of n-gram k between the ground truth response r and the proposed response $\hat{r}$, as it computes the fewest appearances of k in either response.

3.2 Embedding-based Metrics

An alternative to using word-overlap based metrics is to consider the meaning of each word as defined by a *word embedding*, which assigns a vector to each word. Methods such as Word2Vec (Mikolov et al., 2013) calculate these embeddings using distributional semantics; that is, they approximate the meaning of a word by considering how often it co-occurs with other words in the corpus.[4] These embedding-based metrics usually approximate sentence-level embeddings using some heuristic to combine the vectors of the individual words in the sentence. The sentence-level embeddings between the candidate and target response are compared using a measure such as cosine distance.

Greedy Matching. Greedy matching is the one embedding-based metric that does not compute sentence-level embeddings. Instead, given two sequences r and $\hat{r}$, each token $w \in r$ is greedily matched with a token $\hat{w} \in \hat{r}$ based on the cosine similarity of their word embeddings (e_w), and the total score is then averaged across all words:

$$G(r, \hat{r}) = \frac{\sum_{w \in r;} \max_{\hat{w} \in \hat{r}} cos\_sim(e_w, e_{\hat{w}})}{|r|}$$

$$GM(r, \hat{r}) = \frac{G(r, \hat{r}) + G(\hat{r}, r)}{2}$$

This formula is asymmetric, thus we must average the greedy matching scores G in each direction. This was originally introduced for intelligent tutoring systems (Rus and Lintean, 2012). The greedy approach favours responses with key words that are semantically similar to those in the ground truth response.

Embedding Average. The embedding average metric calculates sentence-level embeddings using additive composition, a method for computing the meanings of phrases by averaging the vector representations of their constituent words (Foltz et al., 1998; Landauer and Dumais, 1997; Mitchell and Lapata, 2008). This method has been widely used in other domains, for example in textual similarity tasks (Wieting et al., 2015). The embedding average, $\bar{e}$, is defined as the mean of the word embeddings of each token in a sentence r:

$$\bar{e}_r = \frac{\sum_{w \in r} e_w}{|\sum_{w' \in r} e_{w'}|}.$$

To compare a ground truth response r and retrieved response $\hat{r}$, we compute the cosine similarity between their respective sentence level embeddings: EA $:= \cos(\bar{e}_r, \bar{e}_{\hat{r}})$.

Vector Extrema. Another way to calculate sentence-level embeddings is using vector extrema (Forgues et al., 2014). For each dimension of the word vectors, take the most extreme value amongst *all word vectors in the sentence*, and use that value in the sentence-level embedding:

$$e_{rd} = \begin{cases} \max_{w \in r} e_{wd} & \text{if } e_{wd} > |\min_{w' \in r} e_{w'd}| \\ \min_{w \in r} e_{wd} & \text{otherwise} \end{cases}$$

where d indexes the dimensions of a vector; e_{wd} is the d'th dimensions of e_w (w's embedding). The min in this equation refers to the selection of the largest negative value, if it has a greater magnitude than the largest positive value.

Similarity between response vectors is again computed using cosine distance. Intuitively, this approach prioritizes informative words over common ones; words that appear in similar contexts will be close together in the vector space. Thus, common words are pulled towards the origin because they occur in various contexts, while words carrying important semantic information will lie further away. By taking the extrema along each dimension, we are thus more likely to ignore common words.

4 Dialogue Response Generation Models

In order to determine the correlation between automatic metrics and human judgements of response quality, we obtain response from a diverse range of response generation models in the recent literature, including both *retrieval* and *generative* models.

4.1 Retrieval Models

Ranking or retrieval models for dialogue systems are typically evaluated based on whether they can retrieve the correct response from a corpus of predefined responses, which includes the ground truth

[4] To maintain statistical independence between the task and each performance metric, it is important that the word embeddings used are trained on corpora which do not overlap with the task corpus.

	Ubuntu Dialogue Corpus			Twitter Corpus		
	Embedding Averaging	Greedy Matching	Vector Extrema	Embedding Averaging	Greedy Matching	Vector Extrema
R-TFIDF	0.536 ± 0.003	0.370 ± 0.002	0.342 ± 0.002	0.483 ± 0.002	0.356 ± 0.001	0.340 ± 0.001
C-TFIDF	0.571 ± 0.003	0.373 ± 0.002	0.353 ± 0.002	0.531 ± 0.002	0.362 ± 0.001	0.353 ± 0.001
DE	$\mathbf{0.650 \pm 0.003}$	0.413 ± 0.002	0.376 ± 0.001	$\mathbf{0.597 \pm 0.002}$	0.384 ± 0.001	0.365 ± 0.001
LSTM	0.130 ± 0.003	0.097 ± 0.003	0.089 ± 0.002	0.593 ± 0.002	$\mathbf{0.439 \pm 0.002}$	$\mathbf{0.420 \pm 0.002}$
HRED	0.580 ± 0.003	$\mathbf{0.418 \pm 0.003}$	$\mathbf{0.384 \pm 0.002}$	$\mathbf{0.599 \pm 0.002}$	$\mathbf{0.439 \pm 0.002}$	$\mathbf{0.422 \pm 0.002}$

Table 2: Models evaluated using the vector-based evaluation metrics, with 95% confidence intervals.

response to the conversation (Schatzmann et al., 2005). Such systems can be evaluated using recall or precision metrics. However, when deployed in a real setting these models will not have access to the correct response given an unseen conversation. Thus, in the results presented below we *remove* one occurrence of the ground-truth response from the corpus and ask the model to retrieve the most appropriate response from the remaining utterances. Note that this does not mean the correct response will not appear in the corpus at all; in particular, if there exists another context in the dataset with an identical ground-truth response, this will be available for selection by the model.

We then evaluate each model by comparing the retrieved response to the ground truth response of the conversation. This closely imitates real-life deployment of these models, as it tests the ability of the model to generalize to unseen contexts.

TF-IDF. We consider a simple Term Frequency - Inverse Document Frequency (TF-IDF) retrieval model (Lowe et al., 2015). TF-IDF is a statistic that intends to capture how important a given word is to some document, which is calculated as: $\text{tfidf}(w, c, C) = f(w, c) \times \log \frac{N}{|\{c \in C : w \in c\}|}$, where C is the set of all contexts in the corpus, $f(w, c)$ indicates the number of times word w appeared in context c, N is the total number of dialogues, and the denominator represents the number of dialogues in which the word w appears.

In order to apply TF-IDF as a retrieval model for dialogue, we first compute the TF-IDF vectors for each context and response in the corpus. We then return the response with the largest cosine similarity in the corpus, either between the input context and corpus contexts (C-TFIDF), or between the input context and corpus responses (R-TFIDF).

Dual Encoder. Next we consider the recurrent neural network (RNN) based architecture called the Dual Encoder (DE) model (Lowe et al., 2015). The DE model consists of two RNNs which respectively compute the vector representation of an input context and response, $c, r \in \mathbb{R}^n$. The model then calculates the probability that the given response is the ground truth response given the context, by taking a weighted dot product: $p(r \text{ is correct} | c, r, M) = \sigma(c^T M r + b)$ where M is a matrix of learned parameters and b is a bias. The model is trained using negative sampling to minimize the cross-entropy error of all (context, response) pairs. To our knowledge, our application of neural network models to large-scale retrieval in dialogue systems is novel.

4.2 Generative Models

In addition to retrieval models, we also consider generative models. In this context, we refer to a model as generative if it is able to generate entirely new sentences that are unseen in the training set.

LSTM language model. The baseline model is an LSTM language model (Hochreiter and Schmidhuber, 1997) trained to predict the next word in the (context, response) pair. During test time, the model is given a context, encodes it with the LSTM and generates a response using a greedy beam search procedure (Graves, 2013).

HRED. Finally we consider the Hierarchical Recurrent Encoder-Decoder (HRED) (Serban et al., 2015). In the traditional Encoder-Decoder framework, all utterances in the context are concatenated together before encoding. Thus, information from previous utterances is far outweighed by the most recent utterance. The HRED model uses a *hierarchy* of encoders; each utterance in the context passes through an 'utterance-level' encoder, and the

	Twitter				Ubuntu			
Metric	**Spearman**	p-value	**Pearson**	p-value	**Spearman**	p-value	**Pearson**	p-value
Greedy	0.2119	0.034	0.1994	0.047	0.05276	0.6	0.02049	0.84
Average	0.2259	0.024	0.1971	0.049	-0.1387	0.17	-0.1631	0.10
Extrema	0.2103	0.036	0.1842	0.067	0.09243	0.36	-0.002903	0.98
METEOR	0.1887	0.06	0.1927	0.055	0.06314	0.53	0.1419	0.16
BLEU-1	0.1665	0.098	0.1288	0.2	-0.02552	0.8	0.01929	0.85
BLEU-2	0.3576	< 0.01	0.3874	< 0.01	0.03819	0.71	0.0586	0.56
BLEU-3	0.3423	< 0.01	0.1443	0.15	0.0878	0.38	0.1116	0.27
BLEU-4	0.3417	< 0.01	0.1392	0.17	0.1218	0.23	0.1132	0.26
ROUGE	0.1235	0.22	0.09714	0.34	0.05405	0.5933	0.06401	0.53
Human	0.9476	< 0.01	1.0	0.0	0.9550	< 0.01	1.0	0.0

Table 3: Correlation between each metric and human judgements for each response. Correlations shown in the human row result from randomly dividing human judges into two groups.

	Spearman	p-value	**Pearson**	p-value
BLEU-1	0.1580	0.12	0.2074	0.038
BLEU-2	0.2030	0.043	0.1300	0.20

Table 4: Correlation between BLEU metric and human judgements after removing stopwords and punctuation for the Twitter dataset.

	Mean score		
	$\Delta w <= 6$ (n=47)	$\Delta w >= 6$ (n=53)	p-value
BLEU-1	0.1724	0.1009	< 0.01
BLEU-2	0.0744	0.04176	< 0.01
Average	0.6587	0.6246	0.25
METEOR	0.2386	0.2073	< 0.01
Human	2.66	2.57	0.73

Table 5: Effect of differences in response length for the Twitter dataset, Δw = absolute difference in #words between a ground truth response and proposed response

output of these encoders is passed through another 'context-level' encoder, which enables the handling of longer-term dependencies.

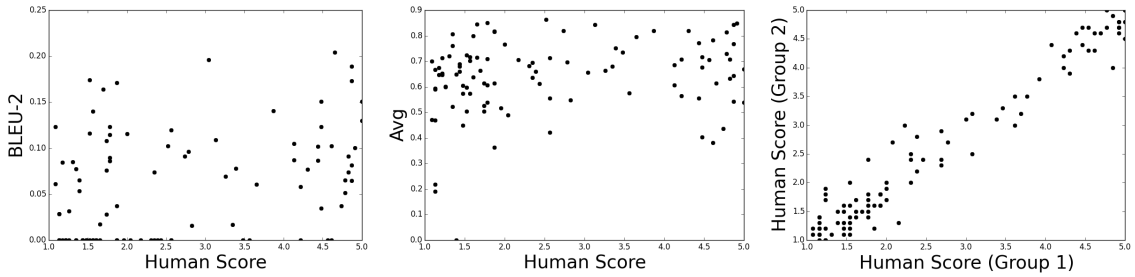
4.3 Conclusions from an Incomplete Analysis

When evaluation metrics are not explicitly correlated to human judgement, it is possible to draw misleading conclusions by examining how the metrics rate different models. To illustrate this point, we compare the performance of selected models according to the embedding metrics on two different domains: the Ubuntu Dialogue Corpus (Lowe et al., 2015), which contains technical vocabulary and where conversations are often oriented towards solving a particular problem, and a non-technical Twitter corpus collected following the procedure of Ritter et al. (2010). We consider these two datasets since they cover contrasting dialogue domains, i.e. technical help vs casual chit-chat, and because they are amongst the largest publicly available corpora, making them good candidates for building data-driven dialogue systems.

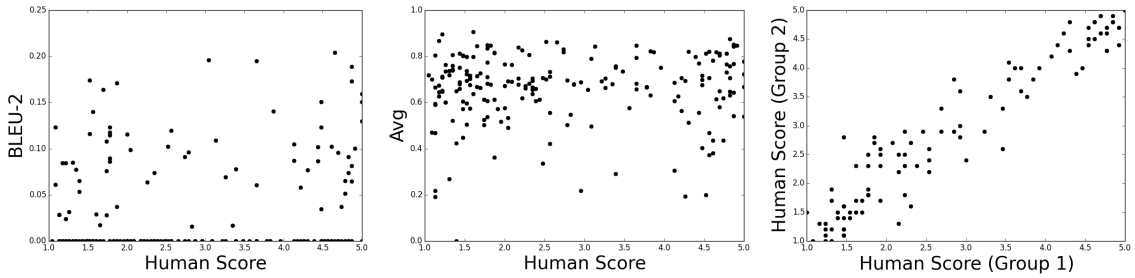
Results on the proposed embedding metrics are shown in Table 2. For the retrieval models, we observe that the DE model significantly outperforms both TFIDF baselines on all metrics across both datasets. Further, the HRED model significantly outperforms the basic LSTM generative model in both domains, and appears to be of similar strength as the DE model. Based on these results, one might be tempted to conclude that there is some information being captured by these metrics, that significantly differentiates models of different quality. However, as we show in the next section, the embedding-based metrics correlate only weakly with human judgements on the Twitter corpus, and not at all on the Ubuntu Dialogue Corpus. This demonstrates that metrics that have not been specifically correlated with human judgements on a new task should not be used to evaluate that task.

5 Human Correlation Analysis

Data Collection. We conducted a human survey to determine the correlation between human judgements on the quality of responses, and the score assigned by each metric. We aimed to follow the procedure for the evaluation of BLEU (Papineni et al.,



(a) Twitter



(b) Ubuntu

Figure 1: Scatter plots showing the correlation between metrics and human judgements on the Twitter corpus (a) and Ubuntu Dialogue Corpus (b). The plots represent BLEU-2 (left), embedding average (center), and correlation between two randomly selected halves of human respondents (right).

2002a). 25 volunteers from the Computer Science department at the author's institution were given a context and one proposed response, and were asked to judge the response quality on a scale of 1 to 5.[5]; a 1 indicates that the response is not appropriate or sensible given the context, and a 5 indicates that the response is very reasonable. Out of the 25 respondents, 23 had Cohen's kappa scores $\kappa > 0.2$ w.r.t. the other respondents, which is a standard measure for inter-rater agreement (Cohen, 1968). The 2 respondents with $\kappa < 0.2$, indicating slight agreement, were excluded from the analysis below. The median κ score was approximately 0.55, roughly indicating moderate to strong annotator agreement.

Each volunteer was given 100 questions per dataset. These questions correspond to 20 unique contexts, with 5 different responses: one utterance randomly drawn from elsewhere in the test set, the response selected from each of the TF-IDF, DE, and HRED models, and a response written by a human annotator. These were chosen as they cover the range of qualities almost uniformly (see Figure 1).

Survey Results. We present correlation results between the human judgements and each metric in Table 3. We compute the Pearson correlation, which estimates linear correlation, and Spearman correlation, which estimates any monotonic correlation.

The first observation is that in both domains the BLEU-4 score, which has previously been used to evaluate unsupervised dialogue systems, shows very weak if any correlation with human judgement. In fact we found that the BLEU-3 and BLEU-4 scores were near-zero for a majority of response pairs; for BLEU-4, only four examples had a score $> 10^{-9}$. Despite this, they still correlate with human judgements on the Twitter Corpus at a rate similar to BLEU-2. This is because of the smoothing constant, which gives a tiny weight to unigrams and bigrams despite the absence of higher-order n-grams. BLEU-3 and BLEU-4 behave as a scaled, noisy version of BLEU-2; thus, if one is to evaluate dialogue

[5]Studies asking humans to evaluate text often rate different aspects separately, such as 'adequacy', 'fluency' and 'informativeness' of the text (Hovy, 1999; Papineni et al., 2002b) Our evaluation focuses on adequacy. We did not consider fluency because 4 out of the 5 proposed responses to each context were generated by a human. We did not consider informativeness because in the domains considered, it is not necessarily important (in Twitter), or else it seems to correlate highly with adequacy (in Ubuntu).

Context of Conversation	**Context of Conversation**
A: dearest! question. how many thousands of people can panaad occupy? B: @user panaad has <number> k seat capacity while rizal has <number> k thats why they choose rizal i think .	A: never felt more sad than i am now B: @user aww why ? A: @user @user its a long story ! sure you wanna know it ? bahaha and thanks for caring btw <heart>
Ground Truth Response	**Ground Truth Response**
A: now i know about the siting capacity . thanks for the info @user great evening.	A: @user i don 't mind to hear it i 've got all day and youre welcome <number>
Proposed Response	**Proposed Response**
A: @user makes sense. thanks!	A: @user i know , i 'm just so happy for you !

Figure 2: Examples where the metrics rated the response poorly and humans rated it highly (left), and the converse (right). Both responses are given near-zero score by BLEU-N for N$>$ 1. While no metric will perform perfectly on all examples, we present these examples to provide intuition on how example-level errors become aggregated into poor correlation to human judgements at the corpus-level.

responses with BLEU, we recommend the choice of $N = 2$ over $N = 3$ or 4. Note that using a test corpus larger than the size reported in this paper may lead to stronger correlations for BLEU-3 and BLEU-4, due to a higher number of non-zero scores.

It is interesting to note that, while some of the embedding metrics and BLEU show small positive correlation in the non-technical Twitter domain, there is no metric that significantly correlates with humans on the Ubuntu Dialogue Corpus. This is likely because the correct Ubuntu responses contain specific technical words that are less likely to be produced by our models. Further, it is possible that responses in the Ubuntu Dialogue Corpus have intrinsically higher variability (or entropy) than Twitter when conditioned on the context, making the evaluation problem significantly more difficult.

Figure 1 illustrates the relationship between metrics and human judgements. We include only the best performing metric using word-overlaps, i.e. the BLEU-2 score (left), and the best performing metric using word embeddings, i.e. the vector average (center). These plots show how weak the correlation is: in both cases, they appear to be random noise. It seems as though the BLEU score obtains a positive correlation because of the large number of responses that are given a score of 0 (bottom left corner of the first plot). This is in stark contrast to the inter-rater agreement, which is plotted between two randomly sampled halves of the raters (right-most plots). We also calculated the BLEU scores after removing stopwords and punctuation from the responses. As shown in Table 4, this weakens the correlation with human judgements for BLEU-2 compared to the values in Table 3, and suggests that BLEU is sensitive to factors that do not change the semantics of the response.

Finally, we examined the effect of response length on the metrics, by considering changes in scores when the ground truth and proposed response had a large difference in word counts. Table 4 shows that BLEU and METEOR are particularly sensitive to this aspect, compared to the Embedding Average metric and human judgement.

Qualitative Analysis. In order to determine specifically why the metrics fail, we examine qualitative samples where there is a disagreement between the metrics and human rating. Although these only show inconsistencies at the example-level, they provide some intuition as to why the metrics don't correlate with human judgements at the corpus-level. We present in Figure 2 two examples where all of the embedding-based metrics and BLEU-1 score the proposed response significantly differently than the humans.

The left of Figure 2 shows an example where the embedding-based metrics score the proposed response lowly, while humans rate it highly. It is clear from the context that the proposed response is reasonable – indeed both responses intend to express gratitude. However, the proposed response has a different wording than the ground truth response, and therefore the metrics are unable to separate the salient words from the rest. This suggests that the embedding-based metrics would ben-

efit from a weighting of word saliency.

The right of the figure shows the reverse scenario: the embedding-based metrics score the proposed response highly, while humans do not. This is most likely due to the frequently occurring 'i' token, and the fact that 'happy' and 'welcome' may be close together in the embedding space. However, from a human perspective there is a significant semantic difference between the responses as they pertain to the context. Metrics that take into account the context may be required in order to differentiate these responses. Note that in both responses in Figure 2, there are no overlapping n-grams greater than unigrams between the ground truth and proposed responses; thus, all of BLEU-2,3,4 would assign a score near 0 to the response.

6 Discussion

We have shown that many metrics commonly used in the literature for evaluating unsupervised dialogue systems do not correlate strongly with human judgement. Here we elaborate on important issues arising from our analysis.

Constrained tasks. Our analysis focuses on relatively unconstrained domains. Other work, which separates the dialogue system into a dialogue planner and a natural language generation component for applications in constrained domains, may find stronger correlations with the BLEU metric. For example, Wen et al. (2015) propose a model to map from dialogue acts to natural language sentences and use BLEU to evaluate the quality of the generated sentences. Since the mapping from dialogue acts to natural language sentences has lower diversity and is more similar to the machine translation task, it seems likely that BLEU will correlate better with human judgements. However, an empirical investigation is still necessary to justify this.

Incorporating multiple responses. Our correlation results assume that only one ground truth response is available given each context. Indeed, this is the common setting in most of the recent literature on training end-to-end conversation models. There has been some work on using a larger set of automatically retrieved plausible responses when evaluating with BLEU (Galley et al., 2015b). However, there is no standard method for doing this in the literature. Future work should examine how retrieving additional responses affects the correlation with word-overlap metrics.

Searching for suitable metrics. While we provide evidence against existing metrics, we do not yet provide good alternatives for unsupervised evaluation. Despite the poor performance of the word embedding-based metrics in this survey, we believe that metrics based on distributed sentence representations hold the most promise for the future. This is because word-overlap metrics will simply require too many ground-truth responses to find a significant match for a reasonable response, due to the high diversity of dialogue responses. As a simple example, the skip-thought vectors of Kiros et al. (2015) could be considered. Since the embedding-based metrics in this paper only consist of basic averages of vectors obtained through distributional semantics, they are insufficiently complex for modeling sentence-level compositionality in dialogue. Instead, these metrics can be interpreted as calculating the *topicality* of a proposed response (i.e. how on-topic the proposed response is, compared to the ground-truth).

All of the metrics considered in this paper directly compare a proposed response to the ground-truth, without considering the context of the conversation. However, metrics that take into account the context could also be considered. Such metrics could come in the form of an *evaluation model* that is learned from data. This model could be either a discriminative model that attempts to distinguish between model and human responses, or a model that uses data collected from the human survey in order to provide human-like scores to proposed responses. Finally, we must consider the hypothesis that learning such models from data is no easier than solving the problem of dialogue response generation. If this hypothesis is true, we must concede and always use human evaluations together with metrics that only roughly approximate human judgements.

References

R. Artstein, S. Gandhe, J. Gerten, A. Leuski, and D. Traum. 2009. Semi-formal evaluation of conversational characters. In *Languages: From Formal to Natural*, pages 22–35. Springer.

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. Association for Computational Linguistics Baltimore, MD, USA.

A. Cahill. 2009. Correlating human and automatic evaluation of a german surface realiser. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 97–100. Association for Computational Linguistics.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *EACL*, volume 6, pages 249–256.

C. Callison-Burch, P. Koehn, C. Monz, K. Peterson, M. Przybocki, and O. F. Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53. Association for Computational Linguistics.

C. Callison-Burch, P. Koehn, C. Monz, and O. F. Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64. Association for Computational Linguistics.

B. Chen and C. Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. *ACL 2014*, page 362.

J. Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

D. Espinosa, R. Rajkumar, M. White, and S. Berleant. 2010. Further meta-evaluation of broad-coverage surface realization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 564–574. Association for Computational Linguistics.

P. W. Foltz, W. Kintsch, and T. K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307.

G. Forgues, J. Pineau, J.-M. Larcheveque, and R. Tremblay. 2014. Bootstrapping dialog systems with word embeddings.

M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. l, J. Gao, and B. Dolan. 2015a. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (Short Papers)*.

M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. 2015b. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863*.

Y. Graham, N. Mathur, and T. Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proc. of NAACL-HLT*, pages 1183–1191. Citeseer.

A. Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

E. Hovy. 1999. Toward finely differentiated evaluation metrics for machine translation. In *Proceedings of the Eagles Workshop on Standards and Evaluation*.

K. Jokinen and M. McTear. 2009. *Spoken Dialogue Systems*. Morgan Claypool.

C. Kamm. 1995. User interfaces for voice applications. *Proceedings of the National Academy of Sciences*, 92(22):10031–10037.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

T. K. Landauer and S. T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

N. Lasguido, S. Sakti, G. Neubig, T. Tomoki, and S. Nakamura. 2014. Utilizing human-to-human conversation examples for a multi domain chat-oriented dialog system. *IEICE TRANSACTIONS on Information and Systems*, 97(6):1497–1505.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

C.-Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

R. Lowe, N. Pow, I. V. Serban, and J. Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL*.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

S. Möller, R. Englert, K. Engelbrecht, V. Hafner, A. Jameson, A. Oulasvirta, A. Raake, and N. Reithinger. 2006. MeMo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *INTERSPEECH*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002a. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics (ACL)*.

K. Papineni, S. Roukos, T. Ward, J. Henderson, and F. Reeder. 2002b. Corpus-based comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish results. In *Proceedings of the second international conference on Human Language Technology Research*, pages 132–137.

E. Reiter and A. Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

A. Ritter, C. Cherry, and B. Dolan. 2010. Unsupervised modeling of twitter conversations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

A. Ritter, C. Cherry, and W. B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

V. Rus and M. Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Schatzmann, K. Georgila, and S. Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. 2015. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Networks. In *AAAI Conference on Artificial Intelligence*.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2015)*.

A. Stent, M. Marge, and M. Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer.

O. Vinyals and Q. Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. ACL.

T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2015. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

Addressee and Response Selection for Multi-Party Conversation

Hiroki Ouchi
Nara Institute of Science and Technology
ouchi.hiroki.nt6@is.naist.jp

Yuta Tsuboi
IBM Research - Tokyo
yutat@jp.ibm.com

Abstract

To create conversational systems working in actual situations, it is crucial to assume that they interact with multiple agents. In this work, we tackle addressee and response selection for multi-party conversation, in which systems are expected to select *whom* they address as well as *what* they say. The key challenge of this task is to jointly model *who* is talking about *what* in a previous context. For the joint modeling, we propose two modeling frameworks: 1) static modeling and 2) dynamic modeling. To show benchmark results of our frameworks, we created a multi-party conversation corpus. Our experiments on the dataset show that the recurrent neural network based models of our frameworks robustly predict addressees and responses in conversations with a large number of agents.

1 Introduction

Short text conversation (STC) has been gaining popularity: given an input message, predict an appropriate response in a single-round, two-party conversation (Wang et al., 2013; Shang et al., 2015). Modeling STC is simpler than modeling a complete conversation, but instantly helps applications such as chat-bots and automatic short-message replies (Ji et al., 2014).

Beyond two-party conversations, there is also a need for modeling *multi-party conversation*, a form of conversation with several interlocutors conversing with each other (Traum, 2003; Dignum and Vreeswijk, 2003; Uthus and Aha, 2013). For example, in the Ubuntu Internet Relay Chat (IRC), several users cooperate to find a solution for a technical issue contributed by another user. Each agent might have one part of the solution, and these pieces have to be combined through conversation in order to come up with the whole solution.

User	Addressee	Utterance
User 1	-	I have a problem when I install ...
SYSTEM	-	did you set initial params ?
User 2	-	Show the error message, and ...
User 1	*SYSTEM*	how ?
User 1	User 2	ok just a moment !
SYSTEM	[To Whom?]	[What?]
	1. User 1	1. see this URL : http://xxxx
	2. User 2	2. It 's already in os

Figure 1: Addressee and response selection for multi-party conversation. A *SYSTEM* is required to select an appropriate addressee from the interlocutors in the conversational context and an appropriate response from the fixed set of candidates.

A unique issue of such multi-party conversations is *addressing*, a behavior whereby interlocutors indicate to whom they are speaking (Jovanović and Akker, 2004; Akker and Traum, 2009). In face-to-face communication, the basic clue for specifying addressees is turning one's face toward the addressee. In contrast, in voice-only or text-based communication, the explicit declaration of addressee's names is more common.

In this work, we tackle *addressee and response selection* for multi-party conversation: given a context, predict an addressee and response. As Figure 1 shows, a system is required to select an addressee from the agents appearing in the previous context and a response from a fixed set of candidate responses (Section 3).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2133–2143,
Austin, Texas, November 1-5, 2016.

The key challenge for predicting appropriate addressees and responses is to jointly capture *who* is talking about *what* at each time step in a context. For jointly modeling the speaker-utterance information, we present two modeling frameworks: 1) *static modeling* and 2) *dynamic modeling* (Section 5). While speakers are represented as fixed vectors in the static modeling, they are represented as hidden state vectors that dynamically change with time steps in the dynamic modeling. In practice, our models trained for the task can be applied to retrieval-based conversation systems, which retrieves candidate responses from a large-scale repository with the matching model and returns the highest scoring one with the ranking model (Wang et al., 2013; Ji et al., 2014; Wang et al., 2015). Our trained models work as the ranking model and allow the conversation system to produce addressees as well as responses.

To evaluate the trained models, we provide a corpus and dataset. By exploiting Ubuntu IRC Logs[1], we build a large-scale multi-party conversation corpus, and create a dataset from it (Section 6). Our experiments on the dataset show the models instantiated by the static and dynamic modeling outperform a strong baseline. In particular, the model based on the dynamic modeling robustly predicts appropriate addressees and responses even if the number of interlocutors in a conversation increases.[2]

We make three contributions in this work:

1. We formalize the task of addressee and response selection for multi-party conversation.

2. We present modeling frameworks and the performance benchmarks for the task.

3. We build a large-scale multi-party conversation corpus and dataset for the task.

2 Related Work

This work follows in the footsteps of Ritter et al. (2011), who tackled the response generation problem: given a context, generate an appropriate response. While previous response generation approaches utilize statistical models on top of heuristic rules or templates (Levin et al., 2000; Young et al., 2010; Walker et al., 2003), they apply *statistical machine translation based techniques* without such heuristics, which leads to recent work utilizing the SMT-based techniques with neural networks (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016).

As another popular approach, *retrieval-based techniques* are used to retrieve candidate responses from a repository and return the highest scoring one with the ranking model (Ji et al., 2014; Wang et al., 2015; Hu et al., 2014; Wang et al., 2013; Lu and Li, 2013). Stemming from this approach, the next utterance classification (NUC) task has been proposed, in which a system is required to select an appropriate response from a fixed set of candidates (Lowe et al., 2015; Kadlec et al., 2015). The NUC is regarded as focusing on the ranking problem of retrieval-based system, since it omits the candidate retrieving step. The merit of NUC is that it allows us to easily evaluate the model performance on the basis of accuracy.

Our proposed addressee and response selection task is an extension of the NUC. We generalize the task by integrating the addressee detection, which has been regarded as a problematic issue in multi-party conversation (Traum, 2003; Jovanović and Akker, 2004; Uthus and Aha, 2013). Basically, the addressee detection has been tackled in the spoken/multimodal dialog system research, and the models largely rely on acoustic signal or gaze information (Jovanović et al., 2006; Akker and Traum, 2009; Ravuri and Stolcke, 2014). This current work is different from such previous work in that our models predict addressees with only textual information.

For predicting addressees or responses, how the context is encoded is crucial. In single-round conversation, a system is expected to encode only one utterance as a context (Ritter et al., 2011; Wang et al., 2013). In contrast, in multi-turn conversation, a system is expected to encode multiple utterances (Shang et al., 2015; Lowe et al., 2015). Very recently, individual personalities have been encoded as distributed embeddings used for response generation in two-party conversation (Li et al., 2016). Our work is different from that work in that our proposed personality-independent representation allows us to handle new agents unseen in the training data.

[1] http://irclogs.ubuntu.com/

[2] Our code, corpus, and dataset are publicly available at https://github.com/hiroki13/response-ranking

	Type	Notation
Input	Responding Agent	a_{res}
	Context	$\mathcal{C}$
	Candidate Responses	$\mathcal{R}$
Output	Addressee	$a \in \mathcal{A}(\mathcal{C})$
	Response	$\boldsymbol{r} \in \mathcal{R}$

Table 1: Notations for the ARS task.

3 Addressee and Response Selection

We propose and formalize the task of *addressee and response selection* (ARS) for multi-party conversation. The ARS task assumes the situation where a responding agent gives a response to an addressee following a context.[3]

Notation

Table 1 shows the notations for the formalization. We denote vectors with bold lower-case (e.g. $\mathbf{x}_t$, $\mathbf{h}$), matrices with bold upper-case (e.g. $\mathbf{W}$, $\mathbf{H}_a$), scalars with italic lower-case or upper-case (e.g. a_m, Q), and sets with bold italic lower-case or cursive upper-case (e.g. $\boldsymbol{x}$, $\mathcal{C}$) letters.

Formalization

Given an input conversational situation $\boldsymbol{x}$, an addressee a and a response $\boldsymbol{r}$ are predicted:

$$\text{GIVEN} : \boldsymbol{x} = (a_{res},\ \mathcal{C},\ \mathcal{R})$$

$$\text{PREDICT} : a, \boldsymbol{r}$$

where a_{res} is a responding agent, $\mathcal{C}$ is a context and $\mathcal{R}$ is a set of candidate responses. The context $\mathcal{C}$ is a sequence of previous utterances up to the current time step T:

$$\mathcal{C} = (\boldsymbol{u}_{a_1,1}, \cdots, \boldsymbol{u}_{a_T,T})$$

where $\boldsymbol{u}_{a_t,t}$ is an utterance given by an agent a_t at a time step t. Each utterance $\boldsymbol{u}_{a_t,t}$ is a sequence of N_t tokens:

$$\boldsymbol{u}_{a_t,t} = (w_{a_t,t,1}, \cdots, w_{a_t,t,N_t})$$

where $w_{a_t,t,n}$ is a token index in the vocabulary $\mathcal{V}$.

[3] In actual situations, responses can be addressed to multiple agents. In this work, we assume the situation where one specific agent can be the addressee of a response.

To predict an addressee a as a target output, we select an agent from a set of the agents appearing in a context $\mathcal{A}(\mathcal{C})$. Note that a ground-truth addressee is always included in $\mathcal{A}(\mathcal{C})$. To predict an appropriate response $\boldsymbol{r}$, we select a response from a set of candidate responses $\mathcal{R}$, which consists of Q candidates:

$$\mathcal{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_Q\}$$

$$\boldsymbol{r}_q = (w_{q,1}, \cdots, w_{q,N_q})$$

where $\boldsymbol{r}_q$ is a candidate response, which consists of N_q tokens, and $w_{q,n}$ is an token index in the vocabulary $\mathcal{V}$.

4 Dual Encoder Models

Our proposed models are extensions of the *dual encoder* (DE) model in (Lowe et al., 2015). The DE model consists of two recurrent neural networks (RNN) that respectively compute the vector representation of an input context and candidate response.

A generic RNN, with input $\mathbf{x}_t \in \mathbb{R}^{d_w}$ and *recurrent* state $\mathbf{h}_t \in \mathbb{R}^{d_h}$, is defined as:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) = \pi(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t) \quad (1)$$

where π is a non-linear function, $\mathbf{W}_x \in \mathbb{R}^{d_h \times d_w}$ is a parameter matrix for $\mathbf{x}_t$, $\mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}$ is a parameter matrix for $\mathbf{h}_{t-1}$, and the recurrence is seeded with the $\mathbf{0}$ vector, i.e. $\mathbf{h}_0 = \mathbf{0}$. The recurrent state $\mathbf{h}_t$ acts as a compact *summary* of the inputs seen up to time step t.

In the DE model, each word vector of the context $\mathcal{C}$ and the response $\boldsymbol{r}_q$ is consumed by each RNN, and is then summarized into the context vector $\mathbf{h}_c \in \mathbb{R}^{d_h}$ and the response vector $\mathbf{h}_q \in \mathbb{R}^{d_h}$. Using these vectors, the model calculates the probability that the given candidate response is the ground-truth response given the context as follows:

$$Pr(y(\boldsymbol{r}_q) = 1 | \mathcal{C}, \boldsymbol{r}_q) = \sigma(\mathbf{h}_c^{\mathrm{T}}\ \mathbf{W}\ \mathbf{h}_q) \quad (2)$$

where y is a binary function mapping from $\boldsymbol{r}_q$ to $\{0, 1\}$, in which 1 represents the ground-truth sample and 0 represents the false one, σ is the logistic sigmoid function, and $\mathbf{W} \in \mathbb{R}^{d_h \times d_h}$ is a parameter matrix. As extensions of this model, we propose our multi-party encoder models.

5 Multi-Party Encoder Models

For capturing multi-party conversational streams, we jointly encode *who* is speaking *what* at each time step. Each agent and its utterance are integrated into the hidden states of an RNN.

We present two multi-party modeling frameworks: (i) *static modeling* and (ii) *dynamic modeling*, both of which jointly utilize agent and utterance representation for encoding multiple-party conversation. What distinguishes the models is that while the agent representation in the static modeling framework is fixed, the one in the dynamic modeling framework changes along with each time step t in a conversation.

Modeling Frameworks

As an instance of the static modeling, we propose a static model to capture the *speaking-orders* of agents in conversation. As an instance of the dynamic modeling, we propose a dynamic model using an RNN to track *agent states*. Note that the agent representations are independent of each personality (unique user). The personality-independent representation allows us to handle new agents unseen in the training data.

Formally, similar to Eq. 2, both of the models calculate the probability that the addressee a_p or response $\boldsymbol{r}_q$ is the ground-truth given the input $\mathbf{x}$:

$$Pr(y(a_p) = 1|\mathbf{x}) = \sigma\left([\mathbf{a}_{res}\,;\,\mathbf{h}_c]^{\mathrm{T}}\,\mathbf{W}_a\;\mathbf{a}_p\right) \quad (3)$$

$$Pr(y(\boldsymbol{r}_q) = 1|\mathbf{x}) = \sigma\left([\mathbf{a}_{res}\,;\,\mathbf{h}_c]^{\mathrm{T}}\,\mathbf{W}_r\,\mathbf{h}_q\right) \quad (4)$$

where y is a binary function mapping from a_p or $\boldsymbol{r}_q$ to $\{0, 1\}$, in which 1 represents the ground-truth sample and 0 represents the false one. The function σ is the logistic sigmoid function. $\mathbf{a}_{res} \in \mathbb{R}^{d_a}$ is a responding agent vector, $\mathbf{a}_p \in \mathbb{R}^{d_a}$ is a candidate addressee vector, $\mathbf{h}_c \in \mathbb{R}^{d_h}$ is a context vector, $\mathbf{h}_q \in \mathbb{R}^{d_h}$ is a candidate response vector. These vectors are respectively defined in each model. $\mathbf{W}_a \in \mathbb{R}^{(d_a+d_h)\times d_h}$ is a parameter matrix for the addressee selection probability, and $\mathbf{W}_r \in \mathbb{R}^{(d_a+d_h)\times d_h}$ is a parameter matrix for the response selection probability. These model parameters are learned during training.

On the basis of Eqs. 3 and 4, a resulting addressee and response are selected as follows:

$$\hat{a} = \underset{a_p \in \mathcal{A}(\mathcal{C})}{\operatorname{argmax}}\, Pr(y(a_p) = 1|\mathbf{x}) \quad (5)$$

$$\hat{\boldsymbol{r}} = \underset{\boldsymbol{r}_q \in \mathcal{R}}{\operatorname{argmax}}\, Pr(y(\boldsymbol{r}_q) = 1|\mathbf{x}) \quad (6)$$

where $\hat{a}$ is the highest probability addressee of a set of agents in the context $\mathcal{A}(\mathcal{C})$, and $\hat{\boldsymbol{r}}$ is the highest probability response of a set of candidate responses $\mathcal{R}$.

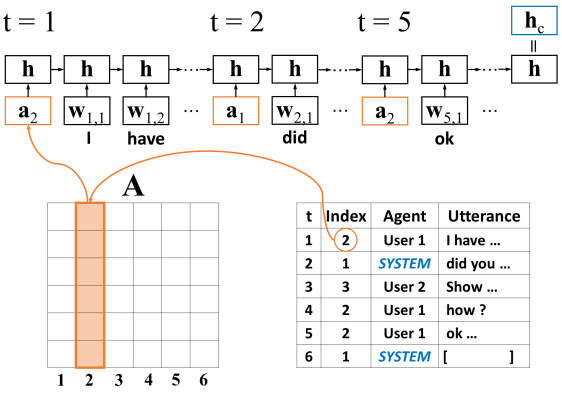


Figure 2: Illustrative example of our static model.

5.1 A Static Model

In the static model, agent matrix $\mathbf{A}$ is defined for the agent vectors in Eqs. 3 and 4. This agent matrix can be defined arbitrarily. We define the agent matrix $\mathbf{A}$ on the basis of *agents' speaking orders*. Intuitively, the agents that spoke in recent time steps are more likely to be an addressee. Our static model captures such property.

The static model is shown in Figure 2. First, agents in the context $\mathcal{A}(\mathcal{C})$ and a responding agent a_{res} are sorted in descending order based on each latest speaking time. Then the order is assigned as an agent index $a_m \in (1, \cdots, |\mathcal{A}(\mathcal{C})|)$ to each agent. In the table shown in Figure 2, the responding agent (represented as `SYSTEM`) has the agent index 1 because he spoke at the most recent time step $t = 6$. Similarly, `User 1` has the index 2 because he spoke at the second most recent time step $t = 5$, and `User 2` has the index 3 because he spoke at the third $t = 3$.

Each speaking-order index a_m is associated with the a_m-th column of the agent matrix $\mathbf{A}$:

$$\mathbf{a}_m = \mathbf{A}[*, a_m]$$

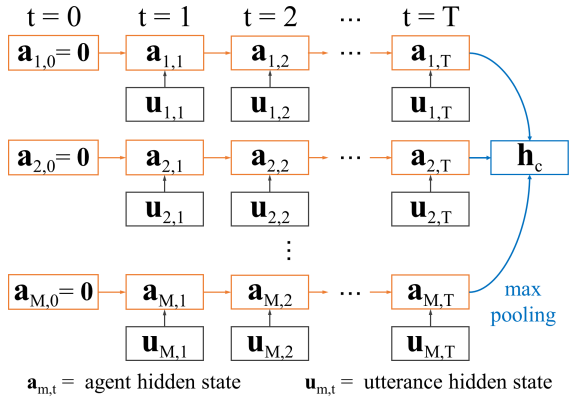


Figure 3: Illustrative example of our dynamic model.

Similarly, a responding agent vector a_{res} and a candidate addressee vector a_p in Eqs. 3 and 4 are respectively extracted from $\mathbf{A}$, i.e. $\mathbf{a}_{res} = \mathbf{A}[*, a_{res}]$ and $\mathbf{a}_p = \mathbf{A}[*, a_p]$.

Consuming the agent vectors, an RNN updates its hidden state. For example, at the time step $t = 1$ in Figure 2, the agent vector $\mathbf{a}_1$ of `User` 1 is extracted from $\mathbf{A}$ on the basis of agent index 2 and then consumed by the RNN. Then, the RNN consumes each word vector $\mathbf{w}$ of `User` 1's utterance. By consuming the agent vector before word vectors, the model can capture which agent speaks the utterance. The last state of the RNN is regarded as $\mathbf{h}_c$. As the transition function f of RNN (Eq. 1), we use the Gated Recurrent Unit (GRU) (Cho et al., 2014; Chung et al., 2014).

For the candidate response vector $\mathbf{h}_q$, each word vector $(\mathbf{w}_{q,1}, \cdots, \mathbf{w}_{q,N_q})$ in the response $\boldsymbol{r}_q$ is summarized with the RNN. Using these vectors $\mathbf{a}_{res}$, $\mathbf{a}_p$, $\mathbf{h}_c$, and $\mathbf{h}_q$, we predict a next addressee and response with the Eqs. 3 and 4.

5.2 A Dynamic Model

In the static model, agent representation $\mathbf{A}$ is a fixed matrix that does not change in a conversational stream. In contrast, in the dynamic model, agent representation $\mathbf{A}_t$ tracks each agent's hidden state which dynamically changes with time steps t.

Figure 3 shows the overview of the dynamic model. Initially, we set a zero matrix as initial agent state $\mathbf{A}_0$, and each column vector of the agent matrix corresponds to an agent hidden state vector. Then, each agent state is updated by consuming the utterance vector at each time step. Note that the states of the agents that are not speaking at the time are updated by zero vectors.

Formally, each column of $\mathbf{A}_t$ corresponds to an agent state vector:

$$\mathbf{a}_{m,t} = \mathbf{A}_t[*, a_m]$$

where an agent state vector $\mathbf{a}_{m,t}$ of an agent a_m at a time step t is the a_m-th column of the agent matrix $\mathbf{A}_t$.

Each vector of the matrix is updated at each time step, as shown in Figure 3. An agent state vector $\mathbf{a}_{m,t} \in \mathbb{R}^{d_a}$ for each agent a_m at each time step t is recurrently computed:

$$\mathbf{a}_{m,t} = g(\mathbf{a}_{m,t-1}, \mathbf{u}_{m,t}), \;\; \mathbf{a}_{m,0} = \mathbf{0}$$

where $\mathbf{u}_{m,t} \in \mathbb{R}^{d_w}$ is a summary vector of an utterance of an agent a_m and computed with an RNN. As the transition function g, we use the GRU. For example, at a time step $t = 2$ in Figure 3, the agent state vector $\mathbf{a}_{1,2}$ is influenced by its utterance vector $\mathbf{u}_{1,2}$ and updated from the previous state $\mathbf{a}_{1,1}$.

The agent matrix updated up to the time step T is denoted as $\mathbf{A}_T$, which is max-pooled and used as a summarized context vector:

$$\mathbf{h}_c = \max_i \; \mathbf{A}_T[i]$$

The agent matrix $\mathbf{A}_T$ is also used for a responding agent vector $\mathbf{a}_{res}$ and a candidate addressee vector $\mathbf{a}_p$, i.e. $\mathbf{a}_{res} = \mathbf{A}_T[*, a_{res}]$ and $\mathbf{a}_p = \mathbf{A}_T[*, a_p]$. $\boldsymbol{r}_q$ is summarized into a response vector $\mathbf{h}_q$ in the same way as the static model.

5.3 Learning

We train the model parameters by minimizing the joint loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \alpha \, \mathcal{L}_a(\boldsymbol{\theta}) + (1 - \alpha) \, \mathcal{L}_r(\boldsymbol{\theta}) + \frac{\lambda}{2} ||\boldsymbol{\theta}||^2$$

where $\mathcal{L}_a$ is the loss function for the addressee selection, $\mathcal{L}_r$ is the loss function for the response selection, α is the hyper-parameter for the interpolation, and λ is the hyper-parameter for the L2 weight decay.

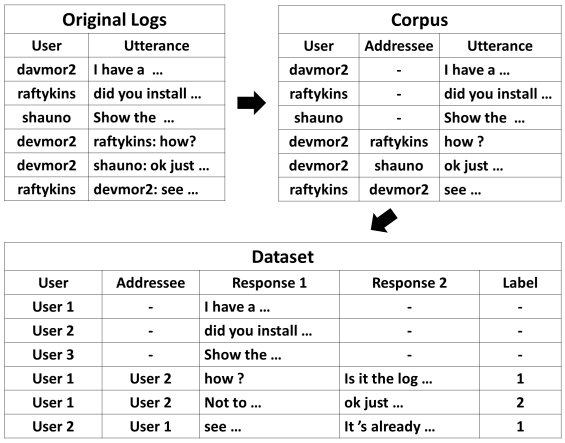


Figure 4: The flow of the corpus and dataset creation. From the original logs, we extract addressee IDs and add them to the corpus. As the dataset, we add candidate responses and the labels.

For addressee and response selection, we use the cross-entropy loss functions:

$$\mathcal{L}_a(\boldsymbol{\theta}) = -\sum_n [\log\ Pr(y(a^+) = 1|\boldsymbol{x}) + \log\left(1 - Pr(y(a^-) = 1|\boldsymbol{x})\right]$$

$$\mathcal{L}_r(\boldsymbol{\theta}) = -\sum_n [\log\ Pr(y(\boldsymbol{r}^+) = 1|\boldsymbol{x}) + \log\left(1 - Pr(y(\boldsymbol{r}^-) = 1|\boldsymbol{x})\right]$$

where $\boldsymbol{x}$ is the input set for the task, i.e. $\boldsymbol{x} = (a_{res}, \mathcal{C}, \mathcal{R})$, a^+ is a ground-truth addressee, a^- is a false addressee, $\boldsymbol{r}^+$ is a ground-truth response, and $\boldsymbol{r}^-$ is a false response. As a false addressee a^-, we pick up and use the addressee with the highest probability from the set of candidate addressees except the ground-truth one ($\mathcal{A}(\mathcal{C}) \setminus a^+$). As a false response, we randomly pick up and use a response from the set of candidate responses except the ground-truth one ($\mathcal{R} \setminus \boldsymbol{r}^+$).

6 Corpus and Dataset

Our goal is to provide a multi-party conversation corpus/dataset that can be used over a wide range of conversation research, such as turn-taking modeling (Raux and Eskenazi, 2009) and disentanglement modeling (Elsner and Charniak, 2010), as well as for the ARS task. Figure 4 shows the flow of the corpus and dataset creation process. We firstly crawl Ubuntu IRC Logs and preprocess the obtained logs.

	Corpus	Dataset		
		Train	Dev	Test
No. of Docs	7355	6,606	367	382
No. of Utters	2.4 M	2.1 M	13.2 k	15.1 k
No. of Words	27.0 M	23.8 M	1.5 M	1.7 M
No. of Samples	-	665.6 k	45.1 k	51.9 k
Avg. W. / U.	11.1	11.1	11.2	11.3
Avg. A. / D.	26.8	26.3	30.68	32.1

Table 2: Statistics of the corpus and dataset. "Docs" is documents, "Utters" is utterances, "W. / U." is the number of words per utterance, "A. / D." is the number of agents per document.

Then, from the logs, we extract and add addressee information to the corpus. In the final step, we set candidate responses and labels as the dataset. Table 2 shows the statistics of the corpus and dataset.

6.1 Ubuntu IRC Logs

The Ubuntu IRC Logs is a collection of logs from Ubuntu-related chat rooms. In each chat room, a number of users chat on and discuss various topics, mainly related to technical support with Ubuntu issues.

The logs are put together into one file per day for each room. Each file corresponds to a document $\mathcal{D}$. In a document, one line corresponds to one log given by a user. Each log consists of three items (`Time`, `UserID`, `Utterance`). Using such information, we create a multi-party conversation corpus.

6.2 The Multi-Party Conversation Corpus

To pick up only the documents written in English, we use a language detection library (Nakatani, 2010). Then, we remove the system logs from each document and leave only user logs. For segmenting the words in each utterance, we use a word tokenizer (`TreebankWordTokenizer`) of the Natural Language Toolkit[4]. Using the preprocessed documents, we create a corpus, whose row consists of the three items (`UserID`, `Addressee`, `Utterance`).

First, the IDs of the users in a document are collected into the user ID list by referring to the `UserID` in each log. Then, as the addressee user ID, we extract the first word of each utterance. In the Ubuntu IRC Logs, users follow the *name mention* convention (Uthus and Aha, 2013), in which they express

[4]http://www.nltk.org/

their addressee by mentioning the addressee's user ID at the beginning of the utterance. By exploiting the name mentions, if the first word of each utterance is identical to a user ID in the user ID list, we extract the addressee ID and then create a table consisting of (UsetID, Addressee, Utterance). In the case that addressee IDs are not explicitly mentioned at the beginning of the utterance, we do not extract anything.

6.3 The ARS Dataset

By exploiting the corpus, we create a dataset for the ARS task. If the line of the corpus includes an addressee ID, we regard it as a sample for the task. As the ground truth addressees and responses, we straightforwardly use the obtained addressee IDs and the preprocessed utterances.

As false responses, we sample utterances elsewhere within a document. This document-within sampling method makes the response selection task more difficult than the random sampling method[5]. One reason for this is that common or similar topics in a document are often discussed and the used words tend to be similar, which makes the word-based features for the task less effective. We partitioned the dataset randomly into a training set (90%), a development set (5%) and a test set (5%).

7 Experiments

We provide performance benchmarks of our learning architectures on the addressee and response selection (ARS) task for multi-party conversation.

7.1 Experimental Setup

Datasets

We use the created dataset for the experiments. The number of candidate responses RES−CAND ($|\mathcal{R}|$) is set to 2 or 10.

Evaluation Metrics

We evaluate performance by accuracies on three aspects: addressee-response pair selection (ADR−RES), addressee selection (ADR), and response selection (RES). In the addressee-response pair selection, we regard the answer as correct if both the addressee and the response are correctly selected. In the addressee/response selection, we regard the answer as correct if the addressee/response is correctly selected.

Optimization

The models are trained by backpropagation through time (Werbos, 1990; Graves and Schmidhuber, 2005). For the backpropagation, we use stochastic gradient descent (SGD) with a mini-batch training method. The mini-batch size is set to 128. The hyper-parameter α for the interpolation between the two loss functions (Section 5.3) is set to 0.5. For the L2 weight decay, the hyper-parameter λ is selected from $\{0.001, 0.0005, 0.0001\}$.

Parameters of the models are randomly initialized over a uniform distribution with support $[-0.01, 0.01]$. To update parameters, we use *Adam* (Kingma and Ba, 2014) with the default setting suggested by the authors. As the word embeddings, we used the 300 dimension vectors pre-trained by *GloVe*[6] (Pennington et al., 2014). To avoid overfitting, the word vectors are fixed across all experiments. The hidden dimensions of parameters are set to $d_w = 300$ and $d_h = 50$ in the both models, and d_a is set to 300 in the static model and 50 in the dynamic model.

To identify the best training epoch and model configuration, we use the *early stopping* method (Yao et al., 2007). In this method, if the best accuracy of ADR−RES on the development set has not been updated for consecutive 5 epochs, training is stopped and the best performing model is picked up. The max epochs is set to 30, which is sufficient for convergence.

Implementation Details

For computational efficiency, we limit the length of a context $\mathcal{C}$ as $\mathcal{C}_{T-N_c+1:T} = (\boldsymbol{u}_{T-N_c+1}, \cdots, \boldsymbol{u}_T)$, where N_c, called *context window*, is the number of utterances prior to a time step t. We set N_c to $\{5, 10, 15\}$. In addition, we truncate the utterances and responses at a maximum of 20 words. For batch processing, we zero-pad them so that the number of words is constant. Out-of-vocabulary words are replaced with <unk>, whose vector is the averaged vector over all word vectors.

[5] Lowe et al. (2015) adopted the random sampling method.

[6] http://nlp.stanford.edu/projects/glove/

	N_c	RES−CAND = 2			RES−CAND = 10		
		ADR−RES	ADR	RES	ADR−RES	ADR	RES
Chance	-	0.62	1.24	50.00	0.12	1.24	10.00
Baseline	5	36.97	55.73	65.68	16.34	55.73	28.19
	10	37.42	55.63	67.79	16.11	55.63	29.48
	15	37.13	55.62	67.89	15.44	55.62	29.19
Static	5	46.99	60.39	75.07	21.98	60.26	33.27
	10	48.67	60.97	77.75	23.31	60.66	35.91
	15	49.27	61.95	78.14	23.49	60.98	36.58
Dynamic	5	49.80	63.19	76.07	23.72	63.28	33.62
	10	53.85	66.94	78.16	25.95	66.70	36.14
	15	**54.88**	**68.54**	**78.64**	**27.19**	**68.41**	**36.93**

Table 3: Benchmark results: accuracies on addressee-response selection (ADR−RES), addressee selection (ADR), and response selection (RES). N_c is the context window. Bolded are the best per column.

Baseline Model

We set a baseline using the term frequency-inverse document frequency (TF-IDF) retrieval model for the response selection (Lowe et al., 2015). We firstly compute two TF-IDF vectors, one for a context window and one for a candidate response. Then, we compute a cosine similarity for these vectors, and select the highest scoring candidate response as a result. For the addressee selection, we adopt a rule-based method: to determine the agent that gives an utterance most recently except a responding agent, which captures the tendency that agents often respond to the other that spoke immediately before.

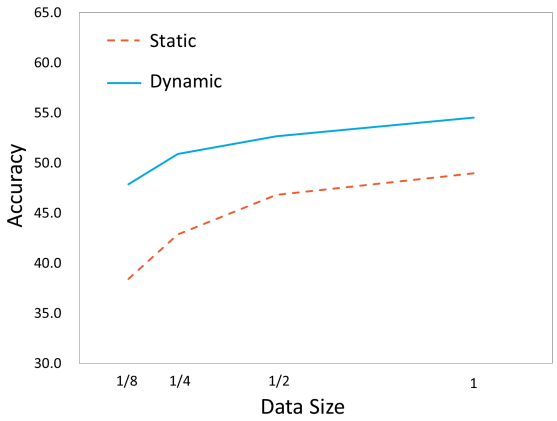


Figure 5: Accuracies in addressee-response selection using different amount of samples for training.

7.2 Results

Overall Performance

Table 3 shows the empirical benchmark results. The dynamic model achieves the best results in all the metrics. The static model outperforms the baseline, but is inferior to the dynamic model.

In addressee selection (ADR), the baseline model achieves around 55% in accuracy. This means that if you select the agents that spoke most recently as an addressee, the half of them are correct. Compared with the baseline, our proposed models achieve better results, which suggests that the models can select the correct addressees that spoke at more previous time steps. In particular, the dynamic model achieves 68% in accuracy, which is 7 point higher than the accuracy of static model.

In response selection (RES), our models outperform the baseline. Compared with the static model, the dynamic model achieves around 0.5 point higher in accuracy.

Effects of the Context Window

In response selection, a performance boost of our proposed models is observed for the context window $\mathcal{N}_c = 10$ over $\mathcal{N}_c = 5$. Comparing the results of the models with the context window $\mathcal{N}_c = 10$ and $\mathcal{N}_c = 15$, the performance is improved but relatively small, which suggests that the performance almost reaches the convergence. In addressee selection, the performance improvements of the static model with the broader context window is limited. In contrast, in the dynamic model, a steady performance boost is observed, yielding an increase of over 5 points between $N_c = 15$ and $N_c = 5$,

No. of Agents	2-5	6-10	11-15	16-20	21-30	31-100	101-305
No. of Samples	3731	5962	5475	4495	5619	7956	18659
	ADR−RES						
Baseline	52.13	43.51	39.98	42.96	39.70	36.55	29.22
Static	64.17	55.92	50.72	53.04	48.69	49.61	42.86
Dynamic	**66.90**	**57.73**	**54.32**	**55.64**	**51.61**	**55.88**	**52.14**
	ADR						
Baseline	84.94	70.82	62.14	65.52	58.89	51.28	41.47
Static	86.33	74.37	66.12	68.54	63.43	59.24	50.99
Dynamic	**87.64**	**76.48**	**69.99**	**72.21**	**66.90**	**66.78**	**62.11**
	RES						
Baseline	60.71	61.24	64.51	65.58	67.93	71.66	71.38
Static	73.60	73.45	74.54	**75.95**	75.17	81.50	81.60
Dynamic	**75.64**	**74.12**	**75.53**	75.17	**76.05**	**81.96**	**81.81**

Table 4: Performance comparison for different numbers of agents appearing in the context. The numbers are accuracies on the test set with the number of candidate responses CAND−RES $= 2$ and the context window $N_c = 15$.

Effects of the Sample Size

Figure 5 shows the accuracy curves of addressee-response selection (ADR−RES) for different training sample sizes. We use $1/2$, $1/4$, and $1/8$ of the whole training samples for training. The results show that as the amount of the data increases, the performance of our models are improved and gradually approaches the convergence. Remarkably, the performance of the dynamic models using the $1/8$ samples is comparable to that of the static model using the whole samples.

Effects of the Number of Participants

To shed light on the relationship between the model performance and the number of agents in multi-party conversation, we investigate the effect of the number of agents participating in each context. Table 4 compares the performance of the models for different numbers of agents in a context.

In addressee selection, the performance of all models gradually gets worse as the number of agents in the context increases. However, compared with the baseline, our proposed models suppress the performance degradation. In particular, the dynamic model predicts correct addressees most robustly.

In response selection, unexpectedly, the performance of all the models gets better as the number of agents increases. Detailed investigation on the interaction between the number of agents and the response selection complexity is an interesting line of future work.

8 Conclusion

We proposed addressee and response selection for multi-party conversation. Firstly, we provided the formal definition of the task, and then created a corpus and dataset. To present benchmark results, we proposed two modeling frameworks, which jointly model speakers and their utterances in a context. Experimental results showed that our models of the frameworks outperform a baseline.

Our future objective to tackle the task of predicting *whether to respond* to a particular utterance. In this work, we assume that the situations where there is a specific addressee that needs an appropriate response and a system is required to respond. In actual multi-party conversation, however, a system sometimes has to wait and listen to the conversation that other participants are engaging in without needless interruption. Hence, the prediction of *whether to respond* in a multi-party conversation would be an important next challenge.

Acknowledgments

We thank Graham Neubig, Yuya Taguchi, Ryosuke Kohita, Ander Martinez, the members of the NAIST Computational Linguistics Laboratory, the members of IBM Research - Tokyo, Long Duong, and the reviewers for their helpful comments.

References

Rieks Akker and David Traum. 2009. A comparison of addressee detection methods for multiparty conversations. In *Workshop on the Semantics and Pragmatics of Dialogue*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv: 1412.3555*.

Frank PM Dignum and Gerard AW Vreeswijk. 2003. Towards a testbed for multi-party dialogues. *Advances in Agent Communication*, pages 212–230.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, pages 389–409.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv: 1408.6988*.

Natasa Jovanović and op den Rieks Akker. 2004. Towards automatic addressee identification in multiparty dialogues. In *Proceedings of SIGDIAL*.

Natasa Jovanović, op den Rieks Akker, and Anton Nijholt. 2006. Addressee identification in face-to-face meetings. In *Proceedings of EACL*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindiest. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv: 1510.03753*.

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, pages 11–23.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of ACL*.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGDIAL*, pages 285–294.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of NIPS*, pages 1367–1375.

Shuyo Nakatani. 2010. Language detection library for java.

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

Antoine Raux and Maxine Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of NAACL*, pages 629–637.

Suman V Ravuri and Andreas Stolcke. 2014. Neural network models for lexical addressee detection. In *Proceedings of INTERSPEECH*, pages 298–302.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNL*, pages 583–593.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL/IJCNLP*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL/HLT*, pages 196–205.

David Traum. 2003. Issues in multiparty dialogues. *Advances in Agent communication*, pages 201–211.

David C Uthus and David W Aha. 2013. Multiparticipant chat analysis: A survey. *Artificial Intelligence*, pages 106–121.

Oriol Vinyals and V. Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv: 1506.05869*.

Marilyn A Walker, Rashmi Prasad, and Amanda Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *Proceedings of INTERSPEECH*. Citeseer.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of EMNLP*, pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Proceedings of IJCAI*, pages 1354–1361.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, pages 150–174.

Nonparametric Bayesian Models for Spoken Language Understanding

Kei Wakabayashi
Tsukuba University, 1-2 Kasuga,
Tsukuba, Ibaraki 305-8550, Japan
kwakaba@slis.tsukuba.ac.jp

Johane Takeuchi, Kotaro Funakoshi
and **Mikio Nakano**
Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako, Saitama 351-0188, Japan
{johane.takeuchi,funakoshi,nakano}
@jp.honda-ri.com

Abstract

In this paper, we propose a new generative approach for semantic slot filling task in spoken language understanding using a nonparametric Bayesian formalism. Slot filling is typically formulated as a sequential labeling problem, which does not directly deal with the posterior distribution of possible slot values. We present a nonparametric Bayesian model involving the generation of arbitrary natural language phrases, which allows an explicit calculation of the distribution over an infinite set of slot values. We demonstrate that this approach significantly improves slot estimation accuracy compared to the existing sequential labeling algorithm.

1 Introduction

Spoken language understanding (SLU) refers to the challenge of recognizing a speaker's intent from a natural language utterance, which is typically defined as a *slot filling task*. For example, in the utterance "Remind me to call John at 9am tomorrow", the specified information {**"time": *"9am tomorrow"***} and {**"subject": *"to call John"***} should be extracted. The term *slot* refers to a variable such as the **time** or **subject** that is expected to be filled with a value provided through the user's utterance.

The slot filling task is typically formulated as a sequential labeling problem as shown in Figure 1. This labeling scheme naturally represents the recognition of arbitrary phrases that appear in the transcription of an utterance. Formally speaking, when we assume a given set of slots $\{s_1, ..., s_M\}$ and denote the corresponding slot values by $\{v_{s_1}, ..., v_{s_M}\}$ where $v_{s_i} \in V_{s_i}$, the domain of each slot value V_{s_i} is an infinite set of word sequences. In this paper, we use the term *arbitrary slot filling task* to refer to this implicit problem statement, which inherently underlies the sequential labeling formulation.

In contrast, a different line of work has explored the case where V_{s_i} is provided as a finite set of possible values that can be handled by a backend system (Henderson, 2015). We refer to this type of task as a *categorical slot filling task*. In this case, the slot filling task is regarded as a classification problem that explicitly considers a value-based prediction, as shown in Figure 2. From this point of view, we can say that a distribution of slot values is actually concentrated in a small set of typical phrases, even in the arbitrary slot filling task, because users basically know what kind of function is offered by the system.

To reflect this observation, in this paper we explore the value-based formulation approach for arbitrary slot filling tasks. Unlike the sequential labeling formulation, which is basically position-based label prediction, our method directly estimates the posterior distribution over an infinite set of possible values for each slot V_{s_i}. The distribution is represented by using a Dirichlet process (Gershman and Blei, 2012), which is a nonparametric Bayesian formalism that generates a categorical distribution for any space. We demonstrate that this approach improves estimation accuracy in the arbitrary slot filling task compared with conventional sequential labeling approach.

The rest of this paper is organized as follows. In Section 2, we review the existing approaches for categorical and arbitrary slot filling tasks and intro-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2144–2152,
Austin, Texas, November 1-5, 2016.

O	O	O	O	B-type	O	O	B-area	I-area	O
i'm	looking	for	a	restaurant	in	the	fen	ditton	area

Figure 1: sequential labeling formulation for slot filling tasks.

u = i'm looking for a restaurant in the fen ditton area

p(type)	p(area)	p(food)
$P(restaurant \mid u) = 0.96$	$P(fen\_ditton \mid u) = 0.98$	$P(italian \mid u) = 0.005$
$P(pub \mid u) = 0.03$	$P(girton \mid u) = 0.005$	$P(mexican \mid u) = 0.01$
$P(None \mid u) = 0.01$	$P(None \mid u) = 0.01$	$P(None \mid u) = 0.85$
...	...	...

Figure 2: Value-based formulation. The posterior probabilities of values for each slot are explicitly computed.

duce related work. In Section 3, we present our nonparametric Bayesian formulation, the hierarchical Dirichlet process slot model (HDPSM), which directly models an infinite set of slot values. On the basis of the HDPSM, we develop a generative utterance model that allows us to compute the posterior probability of slot values in Section 4. In Section 5, we introduce a two-stage slot filling algorithm that consists of a candidate generation step and a candidate ranking step using the proposed model. In Section 6, we show the experimental results for multiple datasets in different domains to demonstrate that the proposed algorithm performs better than the baseline sequential labeling method. We conclude in Section 7 with a brief summary.

2 Related Work

The difference between the categorical and arbitrary slot filling approaches has not been explicitly discussed in a comparative manner to date. In this section, we review existing work for both approaches.

For the categorical slot filling approach, various algorithms that directly model the distribution of slot values have been proposed, including generative models (Williams, 2010), maximum entropy linear classifiers (Metallinou et al., 2013), and neural networks (Ren et al., 2014). However, none of these models are applicable for predicting a variable that ranges over an infinite set, and it is not straightforward to extend them suitably. In particular, a discriminative approach is not applicable for arbitrary slot filling tasks because it requires a fixed finite set of slot values to take statistics.

The arbitrary slot filling approach is a natural application of shallow semantic parsing (Gildea, 2002), which is naturally formulated as a sequential labeling problem. Various sequential labeling algorithms have been applied to this task, including support vector machines, conditional random fields (CRF) (Lafferty et al., 2001; Hahn et al., 2011), and deep neural networks (Mesnil et al., 2015; Xu and Sarikaya, 2013). Vukotic et al. (2015) reported that the CRF is still the most accurate, rapid, and stable method among them. Because the focus of this paper is arbitrary slot filling tasks, we use CRFs as our baseline method.

In this paper, we apply nonparametric Bayesian models (Gershman and Blei, 2012) to represent the distribution over arbitrary phrases for each slot. The effectiveness of this phrase modeling approach has been examined in various applications including morphological analysis (Goldwater et al., 2011) and infinite vocabulary topic models (Zhai and Boyd-graber, 2013). Our method can be regarded as an application of this idea, although it is not straightforward to integrate it with the utterance generation process, as we explain later.

Consequently, our proposed method is categorized as a generative approach. There are many advantages inherent in generative approaches that have been examined, including unsupervised SLU (Chen et al., 2015), automatic feature extraction (Tur et al., 2013), and integration with syntactic modeling (Lorenzo et al., 2013). Another convenient property of generative models is that prior knowledge can be integrated in an intuitive way (Raymond et al., 2006). This often leads to better performance with less training data compared with discriminative models trained completely from scratch (Komatani et al., 2010).

3 Hierarchical Dirichlet Process Slot Model

In this section, we present a nonparametric Bayesian formulation that directly models the distribution over an infinite set of possible values for each slot. Let $S = \{s_1, ..., s_{M_S}\}$ be a given set of slots and M_S be the number of slots. We define each slot s_i as a random variable ranging over an infinite set of

letter sequences V, which is represented as follows:

$$V = \{b_1, ..., b_L | b_\iota \in C, L \geq 0\}$$

where C is a set of characters including the blank character and any other character that potentially appears in the transcription of an utterance. Consequently, we regard the set of slots S as also being a random variable that ranges over V^{M_S}. The objective of this section is to develop the formulation of the probabilistic distribution $p(S)$.

3.1 Dirichlet Process

We apply the Dirichlet process (DP) to model both the distribution for an individual slot $p_i(s_i)$ and the joint distribution $p(S)$. In this subsection, we review the definition and key properties of DP with general notation for the target distribution G over the domain $\mathcal{X}$. In the DP for the prior of $p_i(s_i)$ that is described in Section 3.2, the domain $\mathcal{X}$ corresponds to a set of slot values V, e.g., "fen ditton", "new chesterton", and None. In the DP for $p(S)$ presented in Section 3.3, $\mathcal{X}$ indicates a set of tuples of slot values V^{M_S}, e.g., ("restaurant", "new chesterton", "fast food") and ("restaurant", "fen ditton", None).

The DP is a probabilistic distribution over the distribution G. DP is parameterized by α^0 and G^0, where $\alpha^0 > 0$ is a concentration parameter and G^0 is a base distribution over $\mathcal{X}$. If G is drawn from $DP(\alpha^0, G^0)$ (i.e., $G \sim DP(\alpha^0, G^0)$), then the following Dirichlet distributed property holds for any partition of $\mathcal{X}$ denoted by $\{A_1, ..., A_L\}$:

$$(G(A_1), ..., G(A_L)) \sim Dir(\alpha(A_1), ..., \alpha(A_L))$$

where $\alpha(A) = \alpha^0 G^0(A)$, which is known as the base measure of DP.

Ferguson (1973) proved an important property of a posterior distribution of repeated i.i.d. samples $\mathbf{x}_{1:N} = \{x_1, ..., x_N\}$ drawn from $G \sim DP(\alpha^0, G^0)$. Consider a countably infinite set of *atoms* $\boldsymbol{\phi} = \{\phi_1, \phi_2, ...\}$ that are independently drawn from G^0. Let $c_i \in \mathbb{N}$ be the *assignment* of an atom for sample x_i, which is generated by a sequential draw with the following conditional probability:

$$p(c_{N+1} = k | \mathbf{c}_{1:N}) = \begin{cases} \frac{n_k}{N+\alpha^0} & k \leq K \\ \frac{\alpha^0}{N+\alpha^0} & k = K+1 \end{cases}$$

where n_k is the number of times that the kth atom appears in $\mathbf{c}_{1:N}$ and K is the number of different atoms in $\mathbf{c}_{1:N}$. Given the assignment $\mathbf{c}_{1:N}$, the predictive distribution of $x_{N+1} \in \mathcal{X}$ is represented in the following form:

$$P(x_{N+1} = \theta | \mathbf{c}_{1:N}, \boldsymbol{\phi}_{1:K}, \alpha^0, G^0) = \sum_{k=1}^{K} \frac{n_k}{N+\alpha^0} \delta(\phi_k, \theta) + \frac{\alpha^0}{N+\alpha^0} G^0(\theta)$$

The base distribution possibly generates an identical value for different atoms, such as (ϕ_1 = "fen ditton", ϕ_2 = "new chesterton", ϕ_3 = "fen ditton"). The assignment c_i is an auxiliary variable to indicate which of these atoms is assigned to the ith data point x_i; when x_i = "fen ditton", c_i can be 1 or 3. The posterior distribution above depends on the frequency of atom n_k, not on the frequency of θ itself. The atoms $\boldsymbol{\phi}$ and the assignment $\mathbf{c}$ are latent variables that should be determined at runtime.

3.2 Individual Slot Model

First we formulate the distribution for an individual slot as $p_i(s_i) \sim DP(\alpha_i^0, G_i^0)$ where G_i^0 is a base distribution over the set of phrases V. [1] We define G_i^0 as a generative model that consists of two-step generation: generation of the phrase length $0 \leq L_i \leq L^{max}$ using a categorical distribution and generation of a letter sequence $s_i^{1:L}$ using an n-gram model, as follows:

$$L_i \sim Categorical(\boldsymbol{\lambda}_i)$$
$$s_i^\iota \sim p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i)$$

where $\boldsymbol{\lambda}_i$ and $\boldsymbol{\eta}_i$ are parameters for the categorical distribution and the n-gram model for slot s_i, respectively. This explicit modeling of the length helps avoid the bias toward shorter phrases and leads to a better distribution, as reported by Zhai and Boydgraber (2013). We define G_i^0 as a joint distribution of these models:

$$G_i^0(s_i^{1:L_i}) = p(L_i | \boldsymbol{\lambda}_i) \prod_{\iota=1}^{L_i} p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i) \quad (1)$$

G_i^0 potentially generates an empty phrase of $L_i = 0$ to express the case that the slot value v_{s_i} is not

[1] Note that the subscript i for s, p, α^0 and G^0 indicates the slot type such as "type", "area" and "food" in Figure 2.

provided by an utterance. Therefore, the distribution $p_i(s_i)$ can naturally represent the probability of $None$, which is shown in Figure 2.

We consider prior distributions of the parameters $\boldsymbol{\lambda}_i$ and $\boldsymbol{\eta}_i$ to treat the n-gram characteristics of each slot in a fully Bayesian manner. $p(\boldsymbol{\lambda})$ is given as a L^{max}-dimensional symmetric Dirichlet distribution with parameter a. We also define the $|C|$-dimensional symmetric Dirichlet distributions with parameter b for each n-gram context, since given the context $p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i)$ is just a categorical distribution that ranges over C. Consider we observe N phrases $\mathbf{s}_i$ for slot i. Let $n_{i\iota}^L$ be the number of phrases that have length ι and n_{ih}^γ be the number of times that letter $s^\iota = h$ appears after context $s^{\iota-n+1:\iota-1} = \gamma$. The predictive probability of a phrase is represented as follows:

$$G_i^0(s_i^{1:L_i}|\mathbf{s}_i) = \frac{n_{i\iota}^L + b}{N + bC} \prod_{\iota=1}^{L_i} \frac{n_{is_i^\iota}^\gamma + a}{\sum_c n_{ic}^\gamma + a \sum_{l=1}^{L^{max}} n_{il}^L}$$

3.3 Generative Model for a Set of Slot Values

A naive definition of the joint distribution $p(S)$ is a product of all slot probabilities $\prod_{i=1}^{M_S} p_i(s_i)$ for making an independence assumption. However, the slot values are generally correlated with each other (Chen et al., 2015). To obtain more accurate distribution, we formulate $p(S)$ using another DP that recognizes a frequent combination of slot values, as $p(S) \sim DP(\alpha^1, G^2)$ where G^2 is a base distribution over V^{M_S}. We apply the naive independence assumption to G^2 as follows:

$$G^2(S) = \prod_{i=1}^{M_S} p_i(s_i)$$

The whole generation process of S involves two-layered DPs that share atoms among them. In this sense, this generative model is regarded as a hierarchical Dirichlet process (Teh et al., 2005).

Let $G_i^1(s_i) = p_i(s_i)$ and $G^3(S) = p(S)$ for consistent notations. In summary, we define the hierarchical Dirichlet process slot model (HDPSM) as a generative model that has the following generation process.

$$\begin{aligned} G_i^1 &\sim DP(\alpha_i^0, G_i^0) \\ G^3 &\sim DP(\alpha^1, G^2) \\ S &\sim G^3 \end{aligned}$$

3.4 Inference of HDPSM

In a slot filling task, observations of $\mathbf{S}_{1:T} = \{S_1, ..., S_T\}$ are available as training data. The inference of HDPSM refers to the estimation of $\boldsymbol{\lambda}$, $\boldsymbol{\eta}$ and the atom assignments for each DP.

We formulate the HDPSM in a form of the Chinese restaurant franchise process, which is one of the explicit representations of hierarchical DPs obtained by marginalizing out the base distributions. Teh et al. (2005) presents a Gibbs sampler for this representation, which involves a repetitive resampling of atoms and assignment. In our method, we prefer to adopt a single pass inference, which samples the assignment for each observation only once. Our preliminary experiments showed that the quality of inference is not affected because S is observed unlike the settings in Teh et al. (2005).

We denote the atoms and the atom assignment in the first level DP $DP(\alpha^1, G^2)$ by $\boldsymbol{\phi}^1$ and $\mathbf{c}_{1:N}^1$, respectively. The posterior probability of atom assignment for a new observation S_{N+1} is represented as follows:

$$\begin{aligned} &p(c_{N+1}^1 = k | \mathbf{c}_{1:N}^1, \boldsymbol{\phi}^1, S_{N+1}) \\ &\quad \propto \begin{cases} n_k^1 \delta(\phi_k^1, S_{N+1}) & k \leq K \\ \alpha^1 G^2(S_{N+1}) & k = K+1 \end{cases} \end{aligned}$$

where n_k^1 is the number of times that the kth atom appears in $\mathbf{c}_{1:N}^1$ and K is the number of different atoms in $\mathbf{c}_{1:N}^1$.

$\boldsymbol{\phi}_i^0$ and $\mathbf{c}_{i1:K}^0$ denote the atoms and the assignment in the second level DPs $DP(\alpha_i^0, G_i^0)$. The second level DPs assign atoms to each first level atom ϕ_k^1, i.e. the second level atom ϕ_{it}^0 is generated only when a new atom is assigned for S_{N+1} at the first level. The posterior probability of atom assignment at the second level is:

$$\begin{aligned} &p(c_{iK+1}^0 = t | \mathbf{c}_{i1:K}^0, \boldsymbol{\phi}_i^0, s_{N+1i}) \\ &\quad \propto \begin{cases} n_{it}^0 \delta(\phi_{it}^0, S_{N+1}) & t \leq T_i \\ \alpha_i^0 G^0(S_{N+1}) & t = T_i + 1 \end{cases} \end{aligned}$$

where n_{it}^0 is the number of times that the tth atom appears in $\mathbf{c}_{i1:K}^0$ and T_i is the number of different atoms in $\mathbf{c}_{i1:K}^0$.

The single pass inference procedure is presented in Algorithm 1. Given the atoms $\boldsymbol{\phi}$ and the assignments $\mathbf{c}$, the predictive distribution of $S_{N+1} =$

Algorithm 1 Single pass inference of HDPSM

Input: A set of observations $\mathbf{S}_{1:N}$

Set empty list to $\mathbf{c}^1$ and $\mathbf{c}^0_i$
for $d = 1$ to N **do**
  $k \sim p(c^1_d = k|\mathbf{c}^1_{1:d-1}, \boldsymbol{\phi}^1, S_d)$
  if $k = K + 1$ **then**
    for $i = 1$ to M_S **do**
      $t_i \sim p(c^0_{iK+1} = t_i|\mathbf{c}^0_{i1:K}, \boldsymbol{\phi}^0_i, s_{di})$
      if $t_i = T_i + 1$ **then**
        Update n^L_i and n^γ_i with s_{di}
      end if
      $c^0_{K+1} \leftarrow t_i$ and $\phi^0_{it_i} \leftarrow s_{di}$
    end for
  end if
  $c^1_d \leftarrow k$ and $\phi^1_k \leftarrow S$
end for

$\{s_{N+11}, ..., s_{N+1M_S}\}$ is calculated as follows:

$$P(S_{N+1}|\mathbf{c}, \boldsymbol{\phi}) = \sum_{k=1}^{K} \frac{n^1_k}{N + \alpha^1} \delta(\phi^1_k, S_{N+1}) + \frac{\alpha^1}{N + \alpha^1} \prod_{i=1}^{M_S} P(s_{N+1i}|\mathbf{c}^0_i, \boldsymbol{\phi}^0_i) \quad (2)$$

$$P(s_{N+1i}|\mathbf{c}^0_i, \boldsymbol{\phi}^0_i) = \sum_{t=1}^{T_i} \frac{n^0_{it}}{K + \alpha^0_i} \delta(\phi^0_{it}, s_{N+1i}) + \frac{\alpha^0_i}{K + \alpha^0_i} G^0_i(s_{N+1i}|\boldsymbol{\phi}^0_i)$$

4 Generative Model for an Utterance

We present a generative utterance model to derive a slot estimation algorithm given utterance u. Figure 3 presents the basic concept of our generative model. In the proposed model, we formulate the distribution of slot values as well as the distribution of non-slot parts. In Figure 3, the phrases "hi we're in um" and "and we need a" should be removed to identify the slot information. We call these non-slot phrases as *functional fillers* because they more or less have a function to convey information. Identifying the set of non-slot phrases is equivalent to identifying the set of slot phrases. Therefore, we define a generative model of functional fillers in the same way as the slot values.

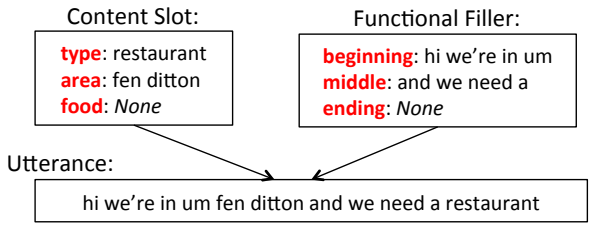


Figure 3: The proposed generative utterance model. We attempt to find the best combination of the slot parts and the non-slot parts (i.e., functional filler parts) by using this model.

4.1 Functional Filler

We assume an utterance u is a concatenation of slot values S and functional fillers F. A functional filler is represented as a phrase that ranges over V. To derive the utterance model, we first formulate a generative model for functional fillers.

In our observation, the distribution of the functional filler depends on its position in an utterance. For example, utterances often begin with typical phrases such as "Hello I'm looking for ..." or "Hi please find ...", which can hardly ever appear at other positions. To reflect this observation, we introduce a *filler slot* to separately model the functional fillers based on a position feature. Specifically, we define three filler slots: *beginning filler* f_1, which precedes any slot value, *ending filler* f_3, which appears at the end of an utterance, and *middle filler* f_2, which is inserted between slot values. We use the term *content slot* to refer to S when we intend to explicitly distinguish it from a filler slot.

Let $F = \{f_1, f_2, f_3\}$ be a set of filler slots and $M_F = 3$ be the number of filler slots. Each slot f_i is a random variable ranging over V and F is a random variable over V^{M_F}. These notations for filler slots indicate compatibility to a content slot, which suggests that we can formulate F using HDPSMs, as follows:

$$H^1_i \sim DP(\beta^0_i, H^0_i)$$
$$H^3 \sim DP(\beta^1, H^2)$$
$$F \sim H^3$$

where H^0_i is an n-gram-based distribution over V that is defined in an identical way to (1) and $H^2(F) = \prod_{i=1}^{M_F} H^1_i(F)$.

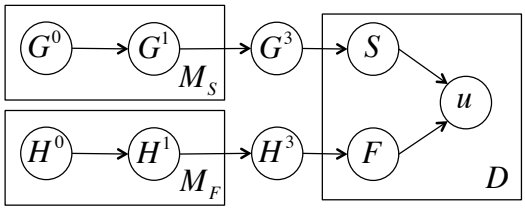


Figure 4: Graphical model of the utterance model.

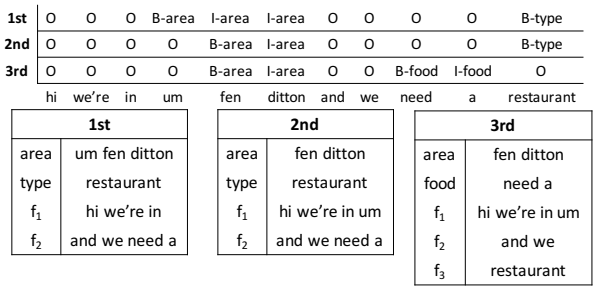


Figure 5: Candidate generation using sequential labeling algorithm. The figure shows the case of $N = 3$.

4.2 Utterance Model

Figure 4 presents the graphical model of our utterance model. We assume that an utterance u is built with phrases provided by S and F. Therefore, the conditional distribution $p(u|S, F)$ basically involves a distribution over the permutation of these slot values with two constraints: f_1 is placed first and f_3 has to be placed last. In our formulation, we simply adopt a uniform distribution over all possible permutations.

For training the utterance model, we assume that a set of annotated utterances is available. Each training instance consists of utterance u and annotated slot values S. Given u and S, we assume that the functional fillers F can be uniquely identified. For the example in Figure 3, we can identify the subsequence in u that corresponds to each content slot value of "restaurant" and "fen ditton". This matching result leads to the identification of filler slot values. Consequently, a triple (u, S, F) is regarded as an observation. Because the HDPSMs of the content slot and of the filler slot are conditionally independent given S and F, we can separately apply Algorithm 1 to train each HDPSM.

For slot filling, we examine the posterior probability of content slot values S given u, which can be reformed as follows:

$$P(S|u) \propto \sum_F P(u|S, F)P(S)P(F)$$

In this equation, we can remove the summation of F because filler slot values F are uniquely identified regarding u and S in our assumption. Additionally, we approximately regard $P(u|S, F)$ as a constant if u can be built with S and F. By using these assumptions, the posterior probability is reduced to the following formula:

$$P(S|u) \propto P(S)P(F) \quad (3)$$

where F in this formula is fillers identified given u and S. Consequently, the proposed method attempts to find the most likely combination of the slot values and the non-slot phrases, since all words in an utterance have to belong to either of them. By using trained HDPSM (i.e., the posterior given all training data), $P(S)$ and $P(F)$ can be computed by (2).

5 Candidate Generation

For estimating slot values given u, we adopt a candidate generation approach (Williams, 2014) that leverages another slot filling algorithm to enumerate likely candidates. [2] Specifically, we assume a candidate generation function $g(u)$ that generates N candidates $\{S_1, ..., S_N\}$ regarding u. Our slot filling algorithm computes the posterior probability by (3) for each candidate slot S_j and takes the candidate that has the highest posterior probability. In this estimation process, our utterance model works as a secondary filter that covers the error of the primary analysis.

Figure 5 provides an example of candidate generation by using a sequential labeling algorithm with IOB tags. The subsequences to which the O tag is assigned can be regarded as functional fillers. The values for each filler slot are identified depending on the position of the subsequence, as the figure shows.

6 Experiments

We evaluate the performance of the proposed generative model with an experiment using the algorithm

[2] The direct inference of the generative utterance model is a topic for near future work. The MCMC method will circumvent the difficulty of searching the entire candidate space.

name	#utterances	#slots	max. diversity
DSTC	1,441	6	55
Weather	1,442	3	191

Table 1: Datasets in the experiment. Max. diversity refers to the maximum number of value types that are taken by a slot.

described in Section 5. We adopt a conditional random field (CRF) as a candidate generation algorithm that generates N-best estimation as candidates. For the CRF, we apply commonly used features including unigram and bigram of the surface form and part of speech of the word. We used CRF++[3] as the CRF implementation.

6.1 Dataset

The performance of our method is evaluated using two datasets from different languages, as summarized in Table 1. The first dataset is provided by the third Dialog State Tracking Challenge (Henderson, 2015), hereafter referred to as the DSTC corpus. The DSTC corpus consists of dialogs in the tourist information domain. In our experiment, we use the user's first utterance in each dialog, which typically describes the user's query to the system. Utterances without any slot information are excluded. We manually modified the annotated slot values into "as-is form" to allow a sequential labeling method to extract the ground-truth values. This identification process can be done in a semi-automatic manner that involves no expert knowledge. We apply the part of speech tagger in NLTK[4] for the CRF application.

The second dataset is a weather corpus consisting of user utterances in an in-house corpus of human-machine dialogues in the weather domain. It contains 1,442 questions spoken in Japanese. In this corpus, the number of value types for each slot is higher than DSTC, which indicates a more challenging task. We applied the Japanese morphological analyzer MeCab (Kudo et al., 2004) to segment the Japanese text into words before applying CRF.

For both datasets, we examine the effect of the amount of available annotated utterances by varying the number of training data in $25, 50, 75, 100, 200, 400, 800, all$.

[3]https://taku910.github.io/crfpp/

[4]http://www.nltk.org/

#train	CRF best	HDP $N = 5$	HDP $N = 300$
25	0.560	0.706*	0.684*
50	0.709	0.791*	0.765*
75	0.748	0.824*	0.817*
100	0.791	0.845*	0.837*
200	0.839	0.901*	0.876*
400	0.904	0.938*	0.936*
800	0.926	0.953*	0.947*
1296	0.938	0.960*	0.951

Table 2: Slot estimation accuracy for the DSTC corpus. The asterisk (*) indicates that the accuracy is statistically significant compared against CRF best ($p < 0.005$).

#train	CRF best	HDP $N = 5$	HDP $N = 300$
25	0.327	0.452*	0.480*
50	0.379	0.488*	0.499*
75	0.397	0.504*	0.522*
100	0.418	0.501*	0.512*
200	0.493	0.526*	0.531*
400	0.512	0.551*	0.549*
800	0.533	0.555*	0.554*
1297	0.546	0.560*	0.554

Table 3: Slot estimation accuracy for the Japanese weather corpus. An asterisk (*) indicates statistical significance against CRF best ($p < 0.01$).

6.2 Evaluation Metrics

The methods are compared in terms of slot estimation accuracy. Let n_c be the number of utterances for which the estimated slot S and the ground-truth slot $\hat{S}$ are perfectly matched, and let n_e be the number of the utterances including an estimation error. The slot estimation accuracy is simply calculated as $\frac{n_c}{n_c+n_e}$. All evaluation scores are calculated as the average of 10-fold cross validation. We also conduct a binomial test to examine the statistical significance of the improvement in the proposed algorithm compared to the CRF baseline.

6.3 Results

Tables 2 and 3 present the slot estimation accuracy for the DSTC corpus and the Japanese weather corpus, respectively. The baseline (CRF best) is a method that takes only one best output of CRF for slot estimation. HDP with $N = 5$ and $N = 300$ is the proposed method, where N is the number of candidates generated by the CRF candidate genera-

utterance	estimation by CRF best	estimation by HDP $N = 5$
im looking for a restaurant that serves fast food	type:restaurant (*)	type:restaurant,food:fast food
i want a moderate restaurant in the new chesterton area	area:new chesterton, type:restaurant, food:moderate (*)	area:new chesterton, type:restaurant, pricerange:moderate
im looking for a cheap chine chinese takeaway restaurant	pricerange:cheap,type:restaurant, food:chinese takeaway	pricerange:cheap,type:restaurant, food:chine chinese takeaway (*)

Table 4: Examples of estimated slot values for the condition of #train is 800. An asterisk (*) indicates misrecognition.

tor. The asterisks (*) beside the HDP accuracy indicate the statistical significance against CRF best, which is tested using the binomial test.

Results show that our proposed method performs significantly better than CRF. Especially when the amount of training data is limited, the proposed method outperforms the baseline. This property is attractive for practical speech recognition systems that offer many different functions. Accurate recognition at an early stage of development allows a practitioner to launch a service that results in quickly collecting hundreds of speech examples.

Since we use the CRF as a candidate generator, we expect that the CRF N-best can rank the correct answer higher in the candidate list. In fact, the top five candidates cover almost all of the correct answers. Therefore, the result in the comparison of $N = 5$ and $N = 300$ suggests the stability of the proposed method against the mostly noisy 295 candidates. Because the proposed algorithm makes no use of the original ranking order, $N = 300$ is a harder condition in which to identify the correct answer. Nevertheless, the result shows that the drop in the performance is limited; the accuracy is still significantly better than the baseline. This result suggests that the proposed method is less dependent on the performance of the candidate generator.

Table 4 presents some examples of the slot values estimated by CRF best and HDP with $N = 5$ for the condition where the number of training utterances is 800. The first two are samples where CRF best failed to predict the correct values. These errors are attributed to infrequent sequential patterns caused by the less trained expressions "that serves fast food" and "moderate restaurant" because CRF is a position-based classifier. The value-based formulation allows the model to learn that the phrase "fast food" is more likely to be a food name than to be a functional filler and to reject the candidate.

The third example in Table 4 shows an error using HDP, which extracted "chine chinese takeaway" which includes a reparandum of disfluency (Georgila et al., 2010). This error can be attributed to the fact that this kind of disfluency resembles the true slot value, which leads to a higher probability of "chine" in the food slot model compared to in the functional filler model. Regarding this type of error, preliminary application of a disfluency detection method (Zayats et al., 2016) is promising for improving accuracy.

The execution time for training the proposed HDP utterance model with 1297 training data in the Japanese weather corpus was about 0.3 seconds. This is a good performance since the CRF training takes about 5.5 seconds. Moreover, the training of the proposed HDP model is scalable and works in an online manner because it is a single pass algorithm. When we have a very large number of training examples, the bottleneck is the CRF training, which requires scanning the whole dataset repeatedly.

7 Conclusion

In this paper, we proposed an arbitrary slot filling method that directly deals with the posterior probability of slot values by using nonparametric Bayesian models. We presented a two-stage method that involves an N-best candidate generation step, which is typically done using a CRF. Experimental results show that our method significantly improves recognition accuracy. This empirical evidence suggests that the value-based formulation is a promising approach for arbitrary slot filling tasks, which is worth exploring further in future work.

References

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander Rudnicky. 2015. Matrix Factorization with Knowledge Graph Propagation for Unsupervised Spoken Language Understanding. In *Proc. Annual Meeting of the Association for Computational Linguistics*.

Thomas S. Ferguson. 1973. A Bayesian Analysis of Some Nonparametric Problems. *The Annual of Statistics*, 1(2):209–230.

Kallirroi Georgila, Ning Wang, and Jonathan Gratch. 2010. Cross-Domain Speech Disfluency Detection. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Samuel J. Gershman and David M. Blei. 2012. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12.

Daniel Gildea. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2011. Producing Power-Law Distributions and Damping Word Frequencies with Two-Stage Language Models. *Journal of Machine Learning Research*, 12:2335–2382.

Stefan Hahn, Marco Dinarelli, Christian Raymond, Fabrice Lefevre, Patrick Lehnen, Renato De Mori, Alessandro Moschitti, Hermann Ney, and Giuseppe Riccardi. 2011. Comparing stochastic approaches to spoken language understanding in multiple languages. *IEEE Transactions on Audio, Speech and Language Processing*, 19(6):1569–1583.

Matthew Henderson. 2015. Machine Learning for Dialog State Tracking: A Review. In *Proc. Workshop on Machine Learning in Spoken Language Processing*.

Kazunori Komatani, Masaki Katsumaru, Mikio Nakano, Kotaro Funakoshi, Tetsuya Ogata, and Hiroshi G. Okuno. 2010. Automatic Allocation of Training Data for Rapid Prototyping. In *Proc. International Conference on Computational Linguistics*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proc. Empirical Methods in Natural Language Processing*.

John Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning*.

Alejandra Lorenzo, Lina M Rojas-barahona, and Christophe Cerisara. 2013. Unsupervised structured semantic inference for spoken dialog reservation tasks. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Gregoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Angeliki Metallinou, Dan Bohus, and Jason Williams. 2013. Discriminative state tracking for spoken dialog systems. *Proc. Annual Meeting of the Association for Computational Linguistics*.

Christian Raymond, Frédéric Béchet, Renato De Mori, and Géraldine Damnati. 2006. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48(3-4):288–304.

Hang Ren, Weiqun Xu, and Yonghong Yan. 2014. Markovian discriminative modeling for dialog state tracking. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2005. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101:1566–1581.

Gokhan Tur, Asli Celikyilmaz, and Dilek Hakkani-Tur. 2013. Latent Semantic Modeling for Slot Filling in Conversational Understanding. In *Proc. International Conference on Acoustics, Speech and Signal Processing*.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2015. Is it Time to Switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding? In *Proc. Interspeech*.

Jason D. Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proc. International Conference on Acoustics, Speech and Signal Processing*.

Jason D Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency Detection using a Bidirectional LSTM. arXiv preprint arXiv:1604.03209.

Ke Zhai and Jordan Boyd-graber. 2013. Online Latent Dirichlet Allocation with Infinite Vocabulary. In *Proc. International Conference on Machine Learning*.

Conditional Generation and Snapshot Learning in Neural Dialogue Systems

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Steve Young
Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{thw28,mg436,nm480,lmr46,phs26,su259,djv27,sjy11}@cam.ac.uk

Abstract

Recently a variety of LSTM-based conditional language models (LM) have been applied across a range of language generation tasks. In this work we study various model architectures and different ways to represent and aggregate the source information in an end-to-end neural dialogue system framework. A method called snapshot learning is also proposed to facilitate learning from supervised sequential signals by applying a *companion* cross-entropy objective function to the conditioning vector. The experimental and analytical results demonstrate firstly that competition occurs between the conditioning vector and the LM, and the differing architectures provide different trade-offs between the two. Secondly, the discriminative power and transparency of the conditioning vector is key to providing both model interpretability and better performance. Thirdly, snapshot learning leads to consistent performance improvements independent of which architecture is used.

1 Introduction

Recurrent Neural Network (RNN)-based conditional language models (LM) have been shown to be very effective in tackling a number of real world problems, such as machine translation (MT) (Cho et al., 2014) and image caption generation (Karpathy and Fei-Fei, 2015). Recently, RNNs were applied to task of generating sentences from an explicit semantic representation (Wen et al., 2015a). Attention-based methods (Mei et al., 2016) and Long Short-term Memory (LSTM)-like (Hochreiter and Schmidhuber, 1997) gating mechanisms (Wen et al., 2015b) have both been studied to improve generation quality. Although it is now clear that LSTM-based conditional LMs can generate plausible natural language, less effort has been put in comparing the different model architectures. Furthermore, conditional generation models are typically tested on relatively straightforward tasks conditioned on a single source (e.g. a sentence or an image) and where the goal is to optimise a single metric (e.g. BLEU). In this work, we study the use of conditional LSTMs in the generation component of neural network (NN)-based dialogue systems which depend on multiple conditioning sources and optimising multiple metrics.

Neural conversational agents (Vinyals and Le, 2015; Shang et al., 2015) are direct extensions of the sequence-to-sequence model (Sutskever et al., 2014) in which a conversation is cast as a source to target transduction problem. However, these models are still far from real world applications because they lack any capability for supporting domain specific tasks, for example, being able to interact with databases (Sukhbaatar et al., 2015; Yin et al., 2016) and aggregate useful information into their responses. Recent work by Wen et al. (2016a), however, proposed an end-to-end trainable neural dialogue system that can assist users to complete specific tasks. Their system used both distributed and symbolic representations to capture user intents, and these collectively condition a NN language generator to generate system responses. Due to the diversity of the conditioning information sources, the best way to represent and combine them is non-trivial.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2153–2162,
Austin, Texas, November 1-5, 2016.

In Wen et al. (2016a), the objective function for learning the dialogue policy and language generator depends solely on the likelihood of the output sentences. However, this sequential supervision signal may not be informative enough to learn a good conditioning vector representation resulting in a generation process which is dominated by the LM. This can often lead to inappropriate system outputs.

In this paper, we therefore also investigate the use of snapshot learning which attempts to mitigate this problem by heuristically applying *companion* supervision signals to a subset of the conditioning vector. This idea is similar to deeply supervised nets (Lee et al., 2015) in which the final cost from the output layer is optimised together with the companion signals generated from each intermediary layer. We have found that snapshot learning offers several benefits: (1) it consistently improves performance; (2) it learns discriminative and robust feature representations and alleviates the vanishing gradient problem; (3) it appears to learn transparent and interpretable subspaces of the conditioning vector.

2 Related Work

Machine learning approaches to task-oriented dialogue system design have cast the problem as a partially observable Markov Decision Process (POMDP) (Young et al., 2013) with the aim of using reinforcement learning (RL) to train dialogue policies online through interactions with real users (Gašić et al., 2013). In order to make RL tractable, the state and action space must be carefully designed (Young et al., 2010) and the understanding (Henderson et al., 2014; Mrkšić et al., 2015) and generation (Wen et al., 2015b; Wen et al., 2016b) modules were assumed available or trained standalone on supervised corpora. Due to the underlying hand-coded semantic representation (Traum, 1999), the conversation is far from natural and the comprehension capability is limited. This motivates the use of neural networks to model dialogues from end to end as a conditional generation problem.

Interest in generating natural language using NNs can be attributed to the success of RNN LMs for large vocabulary speech recognition (Mikolov et al., 2010; Mikolov et al., 2011). Sutskever et al. (2011) showed that plausible sentences can be obtained by sampling characters one by one from the output layer of an RNN. By conditioning an LSTM on a sequence of characters, Graves (2013) showed that machines can synthesise handwriting indistinguishable from that of a human. Later on, this idea has been tried in several research fields, for example, generating image captions by conditioning an RNN on a convolutional neural network (CNN) output (Karpathy and Fei-Fei, 2015; Xu et al., 2015); translating a source to a target language by conditioning a decoder LSTM on top of an encoder LSTM (Cho et al., 2014; Bahdanau et al., 2015); or generating natural language by conditioning on a symbolic semantic representation (Wen et al., 2015b; Mei et al., 2016). Among all these methods, attention-based mechanisms (Bahdanau et al., 2015; Hermann et al., 2015; Ling et al., 2016) have been shown to be very effective improving performance using a dynamic source aggregation strategy.

To model dialogue as conditional generation, a sequence-to-sequence learning (Sutskever et al., 2014) framework has been adopted. Vinyals and Le (2015) trained the same model on several conversation datasets and showed that the model can generate plausible conversations. However, Serban et al. (2015b) discovered that the majority of the generated responses are generic due to the maximum likelihood criterion, which was latter addressed by Li et al. (2016a) using a maximum mutual information decoding strategy. Furthermore, the lack of a consistent system persona was also studied in Li et al. (2016b). Despite its demonstrated potential, a major barrier for this line of research is data collection. Many works (Lowe et al., 2015; Serban et al., 2015a; Dodge et al., 2016) have investigated conversation datasets for developing chat bot or QA-like general purpose conversation agents. However, collecting data to develop goal oriented dialogue systems that can help users to complete a task in a specific domain remains difficult. In a recent work by Wen et al. (2016a), this problem was addressed by designing an online, parallel version of Wizard-of-Oz data collection (Kelley, 1984) which allows large scale and cheap in-domain conversation data to be collected using Amazon Mechanical Turk. An NN-based dialogue model was also proposed to learn from the collected dataset and was shown to be able to assist human subjects to complete specific tasks.

Snapshot learning can be viewed as a special form of weak supervision (also known as distant- or self supervision) (Craven and Kumlien, 1999; Snow et al., 2004), in which supervision signals are heuristically labelled by matching unlabelled corpora with entities or attributes in a structured database. It has been widely applied to relation extraction (Mintz et al., 2009) and information extraction (Hoffmann et al., 2011) in which facts from a knowledge base (e.g. Freebase) were used as objectives to train classifiers. Recently, self supervision was also used in memory networks (Hill et al., 2016) to improve the discriminative power of memory attention. Conceptually, snapshot learning is related to curriculum learning (Bengio et al., 2009). Instead of learning easier examples before difficult ones, snapshot learning creates an easier target for each example. In practice, snapshot learning is similar to deeply supervised nets (Lee et al., 2015) in which *companion* objectives are generated from intermediary layers and optimised altogether with the output objective.

3 Neural Dialogue System

The testbed for this work is a neural network-based task-oriented dialogue system proposed by Wen et al. (2016a). The model casts dialogue as a source to target sequence transduction problem (modelled by a sequence-to-sequence architecture (Sutskever et al., 2014)) augmented with the dialogue history (modelled by a belief tracker (Henderson et al., 2014)) and the current database search outcome (modelled by a database operator). The model consists of both encoder and decoder modules. The details of each module are given below.

3.1 Encoder Module

At each turn t, the goal of the encoder is to produce a distributed representation of the system action $\mathbf{m}_t$, which is then used to condition a decoder to generate the next system response in skeletal form[1]. It consists of four submodules: intent network, belief tracker, database operator, and policy network.

Intent Network The intent network takes a sequence of tokens[1] and converts it into a sentence embedding representing the user intent using an LSTM network. The hidden layer of the LSTM at the last encoding step $\mathbf{z}_t$ is taken as the representation. As mentioned in Wen et al. (2016a), this representation can be viewed as a distributed version of the speech act (Traum, 1999) used in traditional systems.

Belief Trackers In addition to the intent network, the neural dialogue system uses a set of slot-based belief trackers (Henderson et al., 2014; Mrkšić et al., 2015) to track user requests. By taking each user input as new evidence, the task of a belief tracker is to maintain a multinomial distribution p over values $v \in V_s$ for each informable slot[2] s, and a binary distribution for each requestable slot[2]. These probability distributions $\mathbf{p}_t^s$ are called belief states of the system. The belief states $\mathbf{p}_t^s$, together with the intent vector $\mathbf{z}_t$, can be viewed as the system's comprehension of the user requests up to turn t.

Database Operator Based on the belief states $\mathbf{p}_t^s$, a DB query is formed by taking the union of the maximum values of each informable slot. A vector $\mathbf{x}_t$ representing different degrees of matching in the DB (no match, 1 match, ... or more than 5 matches) is produced by counting the number of matched entities and expressing it as a 6-bin 1-hot encoding. If $\mathbf{x}_t$ is not zero, an associated entity pointer is maintained which identifies one of the matching DB entities selected at random. The entity pointer is updated if the current entity no longer matches the search criteria; otherwise it stays the same.

Policy Network Based on the vectors $\mathbf{z}_t$, $\mathbf{p}_t^s$, and $\mathbf{x}_t$ from the above three modules, the policy network combines them into a single action vector $\mathbf{m}_t$ by a three-way matrix transformation,

$$\mathbf{m}_t = \tanh(\mathbf{W}_{zm}\mathbf{z}_t + \mathbf{W}_{xm}\mathbf{x}_t + \textstyle\sum_{s\in\mathbb{G}} \mathbf{W}_{pm}^s \mathbf{p}_t^s) \quad (1)$$

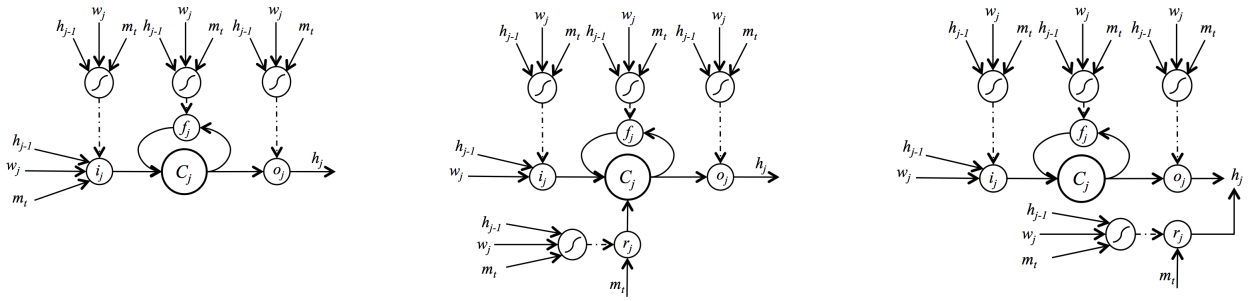
where matrices $\mathbf{W}_{zm}$, $\mathbf{W}_{pm}^s$, and $\mathbf{W}_{xm}$ are parameters and $\mathbb{G}$ is the domain ontology.

3.2 Decoder Module

Conditioned on the system action vector $\mathbf{m}_t$ provided by the encoder module, the decoder module uses a conditional LSTM LM to generate the required system output token by token in skeletal form[1]. The final system response can then be formed

[1]Delexicalisation: slots and values are replaced by generic tokens (e.g. keywords like *Chinese food* are replaced by *[v.food] [s.food]* to allow weight sharing.

[2]Informable slots are slots that users can use to constrain the search, such as food type or price range; Requestable slots are slots that users can ask a value for, such as phone number. This information is specified in the domain ontology.



(a) Language model type LSTM (b) Memory type LSTM (c) Hybrid type LSTM

Figure 1: Three different conditional generation architectures.

by substituting the actual values of the database entries into the skeletal sentence structure.

3.2.1 Conditional Generation Network

In this paper we study and analyse three different variants of LSTM-based conditional generation architectures:

Language Model Type The most straightforward way to condition the LSTM network on additional source information is to concatenate the conditioning vector $\mathbf{m}_t$ together with the input word embedding $\mathbf{w}_j$ and previous hidden layer $\mathbf{h}_{j-1}$,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \hat{\mathbf{c}}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \tanh \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j)$$

where index j is the generation step, n is the hidden layer size, $\mathbf{i}_j, \mathbf{f}_j, \mathbf{o}_j \in [0,1]^n$ are input, forget, and output gates respectively, $\hat{\mathbf{c}}_j$ and $\mathbf{c}_j$ are proposed cell value and true cell value at step j, and $\mathbf{W}_{4n,3n}$ are the model parameters. The model is shown in Figure 1a. Since it does not differ significantly from the original LSTM, we call it the *language model type* (lm) conditional generation network.

Memory Type The *memory type* (mem) conditional generation network was introduced by Wen et al. (2015b), shown in Figure 1b, in which the conditioning vector $\mathbf{m}_t$ is governed by a standalone reading gate $\mathbf{r}_j$. This reading gate decides how much information should be read from the conditioning vector and directly writes it into the memory cell $\mathbf{c}_j$,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \mathbf{r}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_j = \tanh\left(\mathbf{W}_c(\mathbf{w}_j \oplus \mathbf{h}_{j-1})\right)$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j + \mathbf{r}_j \odot \mathbf{m}_t$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j)$$

where $\mathbf{W}_c$ is another weight matrix to learn. The idea behind this is that the model isolates the conditioning vector from the LM so that the model has more flexibility to learn to trade off between the two.

Hybrid Type Continuing with the same idea as the *memory type* network, a complete separation of conditioning vector and LM (except for the gate controlling the signals) is provided by the *hybrid type* network shown in Figure 1c,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \mathbf{r}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_j = \tanh\left(\mathbf{W}_c(\mathbf{w}_j \oplus \mathbf{h}_{j-1})\right)$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j) + \mathbf{r}_j \odot \mathbf{m}_t$$

This model was motivated by the fact that long-term dependency is not needed for the conditioning vector because we apply this information at every step j anyway. The decoupling of the conditioning vector and the LM is attractive because it leads to better interpretability of the results and provides the potential to learn a better conditioning vector and LM.

3.2.2 Attention and Belief Representation

Attention An attention-based mechanism provides an effective approach for aggregating multiple information sources for prediction tasks. Like Wen et al.

(2016a), we explore the use of an attention mechanism to combine the tracker belief states in which the policy network in Equation 1 is modified as

$$\mathbf{m}_t^j = \tanh(\mathbf{W}_{zm}\mathbf{z}_t + \mathbf{W}_{xm}\mathbf{x}_t + \sum_{s\in\mathbb{G}} \alpha_s^j \mathbf{W}_{pm}^s \mathbf{p}_t^s)$$

where the attention weights α_s^j are calculated by,

$$\alpha_s^j = \text{softmax}\big(\mathbf{r}^\intercal \tanh\left(\mathbf{W}_r \cdot (\mathbf{v}_t \oplus \mathbf{p}_t^s \oplus \mathbf{w}_j^t \oplus \mathbf{h}_{j-1}^t)\right)\big)$$

where $\mathbf{v}_t = \mathbf{z}_t + \mathbf{x}_t$ and matrix $\mathbf{W}_r$ and vector $\mathbf{r}$ are parameters to learn.

Belief Representation The effect of different belief state representations on the end performance are also studied. For user informable slots, the *full* belief state $\mathbf{p}_t^s$ is the original state containing all categorical values; the *summary* belief state contains only three components: the summed value of all categorical probabilities, the probability that the user said they "don't care" about this slot and the probability that the slot has not been mentioned. For user requestable slots, on the other hand, the full belief state is the same as the summary belief state because the slot values are binary rather than categorical.

3.3 Snapshot Learning

Learning conditional generation models from sequential supervision signals can be difficult, because it requires the model to learn both long-term word dependencies and potentially distant source encoding functions. To mitigate this difficulty, we introduce a novel method called *snapshot learning* to create a vector of binary labels $\mathbf{\Upsilon}_t^j \in [0,1]^d$, $d < dim(\mathbf{m}_t^j)$ as the *snapshot* of the remaining part of the output sentence $T_{t,j:|T_t|}$ from generation step j. Each element of the snapshot vector is an indicator function of a certain event that will happen in the future, which can be obtained either from the system response or dialogue context at training time. A *companion* cross entropy error is then computed to force a subset of the conditioning vector $\hat{\mathbf{m}}_t^j \subset \mathbf{m}_t^j$ to be close to the snapshot vector,

$$L_{ss}(\cdot) = -\sum_t \sum_j \mathbb{E}[H(\Upsilon_t^j, \hat{m}_t^j)] \quad (2)$$

where $H(\cdot)$ is the cross entropy function, Υ_t^j and $\hat{m}_t^j$ are elements of vectors $\mathbf{\Upsilon}_t^j$ and $\hat{\mathbf{m}}_t^j$, respectively. In order to make the $\tanh$ activations of $\hat{\mathbf{m}}_t^j$ compatible with the 0-1 snapshot labels, we squeeze each value of $\hat{\mathbf{m}}_t^j$ by adding 1 and dividing by 2 before computing the cost.

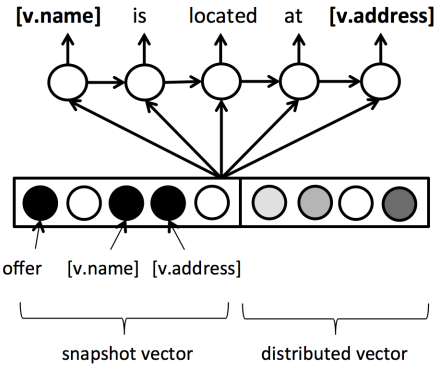


Figure 2: The idea of snapshot learning. The snapshot vector was trained with additional supervisions on a set of indicator functions heuristically labelled using the system response.

The indicator functions we use in this work have two forms: (1) whether a particular slot value (e.g., *[v.food]*[1]) is going to occur, and (2) whether the system has offered a venue[3], as shown in Figure 2. The *offer* label in the snapshot is produced by checking the delexicalised name token (*[v.name]*) in the entire dialogue. If it has occurred, every label in subsequent turns is labelled with 1. Otherwise it is labelled with 0. To create snapshot targets for a particular slot value, the output sentence is matched with the corresponding delexicalised token turn by turn, per generation step. At each generation step, the target is labelled with 0 if that delexicalised token has been generated; otherwise it is set to 1. However, for the models without attention, the targets per turn are set to the same because the condition vector will not be able to learn the dynamically changing behaviour without attention.

4 Experiments

Dataset The dataset used in this work was collected in the Wizard-of-Oz online data collection described by Wen et al. (2016a), in which the task of the system is to assist users to find a restaurant in Cambridge, UK area. There are three informable slots (*food*, *pricerange*, *area*) that users can use to constrain the search and six requestable slots (*address*, *phone*, *postcode* plus the three informable

[3]Details of the specific application used in this study are given in Section 4 below.

Architecture	Belief	Success(%)	SlotMatch(%)	T5-BLEU	T1-BLEU
Belief state representation					
lm	full	72.6 / 74.5	52.1 / 60.3*	0.207 / 0.229*	0.216 / 0.238*
lm	summary	74.5 / 76.5	57.4 / 61.2*	0.221 / 0.231*	0.227 / 0.240*
Conditional architecture					
lm	summary	74.5 / 76.5	57.4 / 61.2*	0.221 / 0.231*	0.227 / 0.240*
mem	summary	75.5 / 77.5	59.2 / **61.3***	0.222 / **0.232***	0.231 / **0.243***
hybrid	summary	76.1 / 79.2	52.4 / 60.6*	0.202 / 0.228*	0.212 / 0.237*
Attention-based model					
lm	summary	79.4 / 78.2	60.6 / 60.2	0.228 / 0.231	0.239 / 0.241
mem	summary	76.5 / 80.2*	57.4 / 61.0*	0.220 / 0.229	0.228 / 0.239
hybrid	summary	79.0 / **81.8***	56.2 / 60.5*	0.214 / 0.227*	0.224 / 0.240*

Table 1: Performance comparison of different model architectures, belief state representations, and snapshot learning. The numbers to the left and right of the **/** sign are learning without and with snapshot, respectively. The model with the best performance on a particular metric (column) is shown in bold face. The *lm* models in *Conditional architecture* and *Attention-based model* are the same models as in Wen et al. (2016a). Statistical significance was computed using two-tailed Wilcoxon Signed-Rank Test (* p $<$0.05) to compare models w/ and w/o snapshot learning.

slots) that the user can ask a value for once a restaurant has been offered. There are 676 dialogues in the dataset (including both finished and unfinished dialogues) and approximately 2750 turns in total. The database contains 99 unique restaurants.

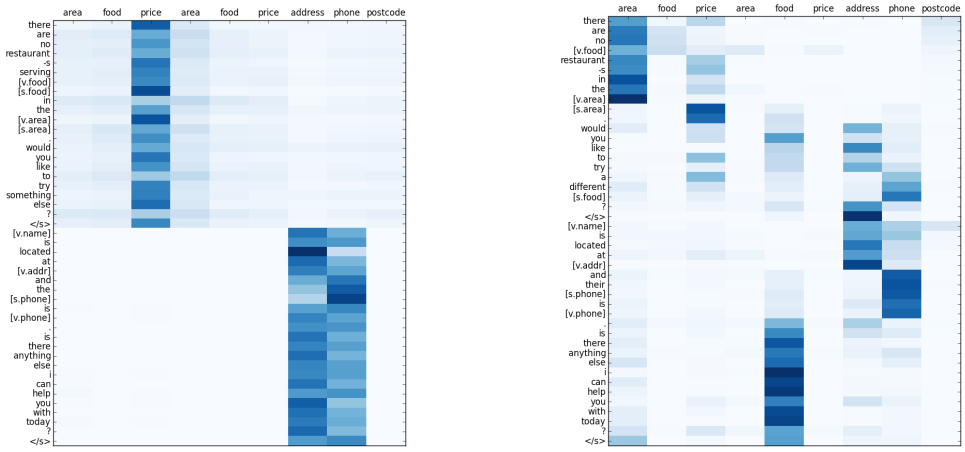
Training The training procedure was divided into two stages. Firstly, the belief tracker parameters θ_b were pre-trained using cross entropy errors between tracker labels and predictions. Having fixed the tracker parameters, the remaining parts of the model $\theta_{\setminus b}$ are trained using the cross entropy errors from the generation network LM,

$$L(\theta_{\setminus b}) = -\sum_t \sum_j H(\mathbf{y}_j^t, \mathbf{p}_j^t) + \lambda L_{ss}(\cdot) \qquad (3)$$

where $\mathbf{y}_j^t$ and $\mathbf{p}_j^t$ are output token targets and predictions respectively, at turn t of output step j, $L_{ss}(\cdot)$ is the snapshot cost from Equation 2, and λ is the tradeoff parameter in which we set to 1 for all models trained with snapshot learning. We treated each dialogue as a batch and used stochastic gradient descent with a small $l2$ regularisation term to train the model. The collected corpus was partitioned into a training, validation, and testing sets in the ratio 3:1:1. Early stopping was implemented based on the validation set considering only LM log-likelihoods. Gradient clipping was set to 1. The hidden layer sizes were set to 50, and the weights were randomly initialised between -0.3 and 0.3 including word embeddings. The vocabulary size is around 500 for both input and output, in which rare words and words that can be delexicalised have been removed.

Decoding In order to compare models trained with different recipes rather than decoding strategies, we decode all the trained models with the average log probability of tokens in the sentence. We applied beam search with a beamwidth equal to 10, the search stops when an end-of-sentence token is generated. In order to consider language variability, we ran decoding until 5 candidates were obtained and performed evaluation on them.

Metrics We compared models trained with different recipes by performing a corpus-based evaluation in which the model is used to predict each system response in the held-out test set. Three evaluation metrics were used: BLEU score (on top-1 and top-5 candidates) (Papineni et al., 2002), slot matching rate and objective task success rate (Su et al., 2015). The dialogue is marked as successful if both: (1) the offered entity matches the task that was specified to the user, and (2) the system answered all the associated information requests (e.g. *what is the address?*) from the user. The slot matching rate is the percentage of delexicalised tokens (e.g. *[s.food]* and *[v.area]*[1]) appear in the candidate also appear in the



(a) *Hybrid* LSTM w/o snapshot learning

(b) *Hybrid* LSTM w/ snapshot learning

Figure 3: Learned attention heat maps over trackers. The first three columns in each figure are informable slot trackers and the rest are requestable slot trackers. The generation model is the *hybrid* type LSTM.

reference. We computed the BLEU scores on the skeletal sentence forms before substituting with the actual entity values. All the results were averaged over 10 random initialised networks.

Results Table 1 shows the evaluation results. The numbers to the left and right of each table cell are the same model trained w/o and w/ snapshot learning. The first observation is that snapshot learning consistently improves on most metrics regardless of the model architecture. This is especially true for BLEU scores. We think this may be attributed to the more discriminative conditioning vector learned through the snapshot method, which makes the learning of the conditional LM easier.

In the first block *belief state representation*, we compare the effect of two different belief representations. As can be seen, using a succinct representation is better (*summary*>*full*) because the identity of each categorical value in the belief state does not help when the generation decisions are done in skeletal form. In fact, the full belief state representation may encourage the model to learn incorrect co-adaptation among features when the data is scarce.

In the *conditional architecture* block, we compare the three different conditional generation architectures as described in section 3.2.1. This result shows that the language model type (*lm*) and memory type (*mem*) networks perform better in terms of BLEU score and slot matching rate, while the hybrid type (*hybrid*) networks achieve higher task success. This is probably due to the degree of separation between the LM and conditioning vector: a coupling approach (*lm, mem*) sacrifices the conditioning vector but learns a better LM and higher BLEU; while a complete separation (*hybrid*) learns a better conditioning vector and offers a higher task success.

Model	$\mathbf{i}_j$	$\mathbf{f}_j$	$\mathbf{r}_j/\mathbf{o}_j$
hybrid, full	0.567	0.502	0.405
hybrid, summary	0.539	0.540	0.428
+ att.	0.540	0.559	0.459

Table 2: Average activation of gates on test set.

Lastly, in the *attention-based model* block we train the three architectures with the attention mechanism and compare them again. Firstly, the characteristics of the three models we observed above also hold for attention-based models. Secondly, we found that the attention mechanism improves all the three architectures on task success rate but not BLEU scores. This is probably due to the limitations of using n-gram based metrics like BLEU to evaluate the generation quality (Stent et al., 2005).

5 Model Analysis

Gate Activations We first studied the average activation of each individual gate in the models by averaging them when running generation on the test set. We analysed the *hybrid* models because their reading gate to output gate activation ratio ($\mathbf{r}_j/\mathbf{o}_j$) shows clear tradeoff between the LM and the conditioning vector components. As can be seen in Ta-

Figure 4: Three example responses generated from the hybrid model trained with snapshot and attention. Each line represents a neuron that detects a particular snapshot event.

ble 2, we found that the average forget gate activations ($\mathbf{f}_j$) and the ratio of the reading gate to the output gate activation ($\mathbf{r}_j/\mathbf{o}_j$) have strong correlations to performance: a better performance (*row 3>row 2>row 1*) seems to come from models that can learn a longer word dependency (higher forget gate $\mathbf{f}_t$ activations) and a better conditioning vector (therefore higher reading to output gate ratio $\mathbf{r}_j/\mathbf{o}_j$).

Learned Attention We have visualised the learned attention heat map of models trained with and without snapshot learning in Figure 3. The attention is on both the informable slot trackers (first three columns) and the requestable slot trackers (the other columns). We found that the model trained with snapshot learning (Figure 3b) seems to produce a more accurate and discriminative attention heat map comparing to the one trained without it (Figure 3a). This may contribute to the better performance achieved by the snapshot learning approach.

Snapshot Neurons As mentioned earlier, snapshot learning forces a subspace of the conditioning vector $\hat{\mathbf{m}}_t^j$ to become discriminative and interpretable. Three example generated sentences together with the snapshot neuron activations are shown in Figure 4. As can be seen, when generating words one by one, the neuron activations were changing to detect different events they were assigned by the snapshot training signals: e.g. in Figure 4b the *light blue* and *orange* neurons switched their domination role when the token *[v.address]* was generated; the *offered* neuron is in a high activation state in Figure 4b because the system was offering a venue, while in Figure 4a it is not activated because the system was still helping the user to find a venue.

6 Conclusion and Future Work

This paper has investigated different conditional generation architectures and a novel method called snapshot learning to improve response generation in a neural dialogue system framework. The results showed three major findings. Firstly, although the *hybrid type* model did not rank highest on all metrics, it is nevertheless preferred because it achieved the highest task success and also it provided more interpretable results. Secondly, snapshot learning provided gains on virtually all metrics regardless of the architecture used. The analysis suggested that the benefit of snapshot learning mainly comes from the more discriminative and robust subspace representation learned from the heuristically labelled companion signals, which in turn facilitates optimisation of the final target objective. Lastly, the results suggested that by making a complex system more interpretable at different levels not only helps our understanding but also leads to the highest success rates.

However, there is still much work left to do. This work focused on conditional generation architectures and snapshot learning in the scenario of generating dialogue responses. It would be very helpful if the same comparison could be conducted in other application domains such as machine translation or image caption generation so that a wider view of the effectiveness of these approaches can be assessed. Furthermore, removing slot-value delexicalisation and learning confirmation behaviour in noisy speech conditions are also main research problems from the system development prospective.

Acknowledgments

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge Research Laboratory.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *ICLR*.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint:1308.0850*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *SIGdial*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

John F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transaction on Information Systems*.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *AISTATS*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. *arXiv perprint:1603.06155*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and

Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint:1603.06744*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGdial*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*.

Tomáš Mikolov, Martin Karafit, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *InterSpeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan H. Černocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*.

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

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015a. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint:1512.05742*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015b. Hierarchical neural network generative models for movie dialogues. In *AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *CICLing 2005*.

Pei-Hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve J. Young. 2015. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Interspeech*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

David R. Traum, 1999. *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGdial*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016a. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint:1604.04562*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2016b. Multi-domain neural network language generation for spoken dialogue systems. In *NAACL-HLT*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables. In *IJCAI*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer, Speech and Language*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of IEEE*.

Relations such as Hypernymy: Identifying and Exploiting Hearst Patterns in Distributional Vectors for Lexical Entailment

Stephen Roller
Department of Computer Science
The University of Texas at Austin
roller@cs.utexas.edu

Katrin Erk
Department of Linguistics
The University of Texas at Austin
katrin.erk@mail.utexas.edu

Abstract

We consider the task of predicting *lexical entailment* using distributional vectors. We perform a novel qualitative analysis of one existing model which was previously shown to only measure the prototypicality of word pairs. We find that the model strongly learns to identify hypernyms using *Hearst patterns*, which are well known to be predictive of lexical relations. We present a novel model which exploits this behavior as a method of feature extraction in an iterative procedure similar to Principal Component Analysis. Our model combines the extracted features with the strengths of other proposed models in the literature, and matches or outperforms prior work on multiple data sets.

1 Introduction

As the field of Natural Language Processing has developed, more ambitious semantic tasks are starting to be addressed, such as Question Answering (QA) and Recognizing Textual Entailment (RTE). These systems often depend on the use of lexical resources like WordNet in order to infer entailments for individual words, but these resources are expensive to develop, and always have limited coverage.

To address these issues, many works have considered on how lexical entailments can be derived automatically using distributional semantics. Some focus mostly on the use of unsupervised techniques, and study measures which emphasize particular word relations (Baroni and Lenci, 2011). Many are based on the Distributional Inclusion Hypothesis, which states that the contexts in which a hypernym appears are a superset of its hyponyms' contexts (Zhitomirsky-Geffet and Dagan, 2005; Kotlerman et al., 2010). More recently, a great deal of work has pushed toward using supervised methods (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015; Kruszewski et al., 2015), varying by their experimental setup or proposed model.

Yet the literature disagrees about which models are strongest (Weeds et al., 2014; Roller et al., 2014), or even if they work at all (Levy et al., 2015). Indeed, Levy et al. (2015) showed that two existing lexical entailment models fail to account for similarity between the antecedent and consequent, and conclude that such models are only learning to predict *prototypicality*: that is, they predict that *cat* entails *animal* because *animal* is usually entailed, and therefore will also predict that *sofa* entails *animal*. Yet it remains unclear why such models make for such strong baselines (Weeds et al., 2014; Kruszewski et al., 2015; Levy et al., 2015).

We present a novel qualitative analysis of one prototypicality classifier, giving new insight into why prototypicality classifiers perform strongly in the literature. We find the model overwhelmingly learns to identify hypernyms using Hearst patterns available in the distributional space, like "animals such as cats" and "animals including cats." These patterns have long been used to identify lexical relations (Hearst, 1992; Snow et al., 2004).

We propose a novel model which exploits this behavior as a method of feature extraction, which we call *H-feature detectors*. Using an iterative procedure similar to Principal Component Analysis, our

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2163–2172,
Austin, Texas, November 1-5, 2016.

model is able to extract and learn using multiple H-feature detectors. Our model also integrates overall word similarity and Distributional Inclusion, bringing together strengths of several models in the literature. Our model matches or outperforms prior work on multiple data sets. The code, data sets, and model predictions are made available for future research.[1]

2 Background

Research on lexical entailment using distributional semantics has now spanned more than a decade, and has been approached using both unsupervised (Weeds et al., 2004; Kotlerman et al., 2010; Lenci and Benotto, 2012; Santus, 2013) and supervised techniques (Baroni et al., 2012; Fu et al., 2014; Roller et al., 2014; Weeds et al., 2014; Kruszewski et al., 2015; Levy et al., 2015; Turney and Mohammad, 2015; Santus et al., 2016). Most of the work in unsupervised methods is based on the Distributional Inclusion Hypothesis (Weeds et al., 2004; Zhitomirsky-Geffet and Dagan, 2005), which states that the contexts in which a hypernym appear should be a superset over its hyponyms' contexts.

This work focuses primarily on the supervised works in the literature. Formally, we consider methods which treat lexical entailment as a supervised classification problem, which take as input the distributional vectors for a pair of words, $(H,\ w)$, and predict on whether the antecedent w entails the consequent H.[2]

One of the earliest supervised approaches was Concat (Baroni et al., 2012). In this work, the *concatenation* of the pair $\langle H,\ w\rangle$ was used as input to an off-the-shelf SVM classifier. At the time, it was very successful, but later works noted that it had major problems with *lexical memorization* (Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015). That is, when the training and test sets were carefully constructed to ensure they were completely disjoint, it performed extremely poorly. Nonetheless, Concat is continually used as a strong baseline in more recent work (Kruszewski et al., 2015).

In response to these issues of lexical memorization, alternative models were proposed. Of particular note are the *Diff* (Fu et al., 2014; Weeds et al., 2014) and *Asym* classifiers (Roller et al., 2014). The Diff model takes the vector difference $H - w$ as input, while the Asym model uses both the vector difference and the squared vector difference as input. Weeds et al. (2014) found that Concat moderately outperformed Diff, while Roller et al. (2014) found that Asym outperformed Concat. Both Diff and Asym can also be seen as a form of supervised Distributional Inclusion Hypothesis, with the vector difference being analogous to the set-inclusion measures of some unsupervised techniques (Roller et al., 2014). All of these works focused exclusively on *hypernymy detection*, rather than the more general task of lexical entailment.

Recently, other works have begun to analyze Concat and Diff for their ability to go beyond just hypernymy detection. Vylomova et al. (2016) take an extensive look at Diff's ability to model a wide variety of lexical relations and conclude it is generally robust, and Kruszewski et al. (2015) have success with a neural network model based on the Distributional Inclusion Hypothesis.

On the other hand, Levy et al. (2015) analyze both Concat and Diff in their ability to detect general lexical entailment on five data sets: two consisting of only hypernymy, and three covering a wide variety of other entailing word relations. They find that both Concat and Diff fail, and analytically show that they are learning to predict the *prototypicality* of the consequent H, rather than the relationship between the antecedent and the consequent, and consider this a form of lexical memorization. They propose a new model, Ksim, which addresses their concerns, but lacks any notion of Distributional Inclusion. In particular, they argue for directly including the cosine similarity of w and H as a term in a custom SVM kernel, in order to determine whether w and H are related all. Ultimately, Levy et al. (2015) conclude that distributional vectors may simply be the wrong tool for the job.

3 Data and Resources

Prior work on lexical entailment relied on a variety of data sets, each constructed in a different manner.

[1] `http://github.com/stephenroller/emnlp2016`

[2] We use the notation w and H for *word* and *hypernym*. These variables refer to either the lexical items, or their distributional vectors, depending on context.

We focus on four different data sets, each of which has been used for evaluation in prior work. Two data sets contain only hypernymy relations, and two consider general lexical entailment.

Our first data set is **LEDS**, the Lexical Entailment Data Set, originally created by Baroni et al. (2012). The data set contains 1385 hyponym-hypernym pairs extracted directly from WordNet, forming a set of positive examples. Negative examples were generated by randomly shuffling the original set of 1385 pairs. As such, LEDS only contains examples of hypernymy and random relations.

Another major data set has been **BLESS**, the Baroni and Lenci (2011) Evaluation of Semantic Spaces. The data set contains annotations of word relations for 200 unambiguous, concrete nouns from 17 broad categories. Each noun is annotated with its co-hyponyms, meronyms, hypernym and some random words. In this work, we treat hypernymy as positive, and other relations as negative.

These two data sets form our hypernymy data sets, but we cannot overstate their important differences: LEDS is balanced, while BLESS contains mostly negative examples; negatives in BLESS include both random pairs *and* pairs exhibiting other strong semantic relations, while LEDS only contains random pairs. Furthermore, all of the negative examples in LEDS are the same lexical items as the positive items, which has strong implications on the prototypicality argument of Levy et al. (2015).

The next data set we consider is **Medical** (Levy et al., 2014). This data set contains high quality annotations of subject-verb-object entailments extracted from medical texts, and transformed into noun-noun entailments by argument alignments. The data contains 12,600 annotations, but only 945 positive examples encompassing various relations like hypernymy, meronomy, synonymy and contextonymy.[3] This makes it one of the most difficult data sets: it is both domain specific and highly unbalanced.

The final data set we consider is **TM14**, a variation on the SemEval 2012 Shared Task of identifying the degree to which word pairs exhibit various relations. These relationships include a small amount of hypernymy, but also many more uncommon relations (agent-object, cause-effect, time-activity, etc). Relationships were binarized into (non-)entailing pairs by Turney and Mohammad (2015). The data set covers 2188 pairs, 1084 of which are entailing.

These two entailment data sets also contain important differences, especially in contrast to the hypernymy data sets. Neither contains any random negative pairs, meaning general semantic similarity measures should be less useful; And both exhibit a variety of non-hypernymy relations, which are less strictly defined and more difficult to model.

3.1 Distributional Vectors

In all experiments, we use a standard, count-based, syntactic distributional vector space. We use a corpus composed of the concatenation of Gigaword, Wikipedia, BNC and ukWaC. We preprocess the corpus using Stanford CoreNLP 3.5.2 (Chen and Manning, 2014) for tokenization, lemmatization, POS-tagging and universal dependency parses. We compute a syntactic distributional space for the 250k most frequent lemmas by counting their dependency neighbors across the corpus. We use only the top 1M most frequent dependency attachments as contexts. We use CoreNLP's "collapsed dependencies", in which prepositional dependencies are collapsed e.g. "go to the store" emits the tuples (go, prep:to+store) and (store, prep:to^{-1}+go). After collecting counts, vectors are transformed using PPMI, SVD reduced to 300 dimensions, and normalized to unit length. The use of collapsed dependencies is very important, as we will see in Section 4, but other parameters are reasonably robust.

4 Motivating Analysis

As discussed in Section 2, the Concat classifier is a classifier trained on the concatenation of the word vectors, $\langle H,\ w\rangle$. As additional background, we first review the findings of Levy et al. (2015), who showed that Concat trained using a linear classifier is only able to capture notions of *prototypicality*; that is, Concat guesses that (*animal*, *sofa*) is a positive example because *animal* looks like a hypernym.

Formally, a linear classifier like Logistic Regression or Linear SVM learns a decision hyperplane represented by a vector $\hat{p}$. Data points are compared to this plane with the inner product: those above

[3] A term for entailments that occur in some contexts, but do not cleanly fit in other categories; e.g. *hospital* entails *doctor*.

the plane (positive inner product) are classified as entailing, and those below as non-entailing. Crucially, since the input features are the concatenation of the pair vectors $\langle H,\ w\rangle$, the hyperplane $\hat{p}$ vector can be *decomposed* into separate H and w components. Namely, if we rewrite the decision plane $\hat{p} = \langle \hat{H}, \hat{w}\rangle$, we find that each pair $\langle H,\ w\rangle$ is classified using:

$$\begin{aligned}&\hat{p}^\top \langle H,\ w\rangle \\ &= \langle \hat{H}, \hat{w}\rangle^\top \langle H,\ w\rangle \\ &= \hat{H}^\top H + \hat{w}^\top w.\end{aligned} \tag{1}$$

This analysis shows that, when the hyperplane $\hat{p}$ is evaluated on a novel pair, it lacks any form of direct interaction between H and w like the inner product $H^\top w$. Without any interaction terms, the Concat classifier has no way of estimating the relationship *between* the two words, and instead only makes predictions based on two independent terms, $\hat{H}$ and $\hat{w}$, the *prototypicality vectors*. Furthermore, the Diff classifier can be analyzed in the same fashion and therefore has the same fatal property.

We agree with this prototypicality interpretation, although we believe it is incomplete: while it places a fundamental ceiling on the performance of these classifiers, it does not explain *why* others have found them to persist as strong baselines (Weeds et al., 2014; Roller et al., 2014; Kruszewski et al., 2015; Vylomova et al., 2016). To approach this question, we consider a baseline Concat classifier trained using a linear model. This classifier should most strongly exhibit the prototypicality behavior according to Equation 1, making it the best choice for analysis. We first consider the most pessimistic hypothesis: is it only learning to memorize which words are hypernyms at all?

We train the baseline Concat classifier using Logistic Regression on each of the four data sets, and extract the vocabulary words which are most similar to the $\hat{H}$ half of the learned hyperplane $\hat{p}$. If the classifier is only learning to memorize the training data, we would expect items from the data to dominate this list of closest vocabulary terms. Table 1 gives the five most similar words to the learned hyperplane, with bold words appearing directly in the data set.

Interestingly, we notice there are very few bold words at all in the list. In LEDS, we actually see some hypernyms of data set items that do not even appear in the data set, and the Medical and TM14 words do not even appear related to the content of the data sets. Similar results were also found for Diff and Asym, and both when using Linear SVM and Logistic Regression. These lists cannot explain the success of the prototypicality classifiers in prior work. Instead, we propose an alternative interpretation of the hyperplane: that of a feature detector for hypernyms, or an *H-feature detector*.

LEDS	BLESS	Medical	TM14
material	goods	item	sensitiveness
structure	lifeform	unlockable	tactility
object	**item**	succor	palate
process	**equipment**	team-up	stiffness
activity	**herbivore**	non-essential	content

Table 1: Most similar words to the prototype $\hat{H}$ learned by the Concat model. Bold items appear in the data set.

4.1 H-Feature Detectors

Recall that distributional vectors are derived from a matrix M containing counts of how often words co-occur with the different syntactic contexts. This co-occurrence matrix is factorized using Singular Value Decomposition, producing both W, the ubiquitous word-embedding matrix, and C, the context-embedding matrix (Levy and Goldberg, 2014):

$$M \approx WC^\top$$

Since the word and context embeddings implicitly live in the same vector space (Melamud et al., 2015), we can also compare Concat's hyperplane with the context matrix C. Under this interpretation, the Concat model does not learn what words are hypernyms, but rather what *contexts* or *features* are indicative of hypernymy. Table 2 shows the syntactic contexts with the highest cosine similarity to the $\hat{H}$ prototype for each of the different data sets.

This view of Concat as an H-feature detector produces a radically different perspective on the classifier's hyperplane. Nearly all of the features learned take the form of Hearst patterns (Hearst, 1992; Snow et al., 2004). The most recognizable and common pattern learned is the "such as" pattern, as in "animals such as cats". These patterns have been well known to be indicative of hypernymy for over two decades. Other interesting pat-

LEDS	BLESS
nmod:such_as+animal	nmod:such_as+submarine
acl:relcl+identifiable	nmod:such_as+ship
nmod:of^{-1}+determine	nmod:such_as+seal
nmod:of^{-1}+categorisation	nmod:such_as+plane
compound+many	nmod:such_as+rack
nmod:such_as+pot	nmod:such_as+rope
Medical	**TM14**
nmod:such_as+patch	amod+desire
nmod:such_as+skin	amod+heighten
nmod:including+skin	nsubj^{-1}+disparate
nmod:such_as+tooth	nmod:such_as+honey
nmod:such_as+feather	nmod:with^{-1}+body
nmod:including+finger	nsubj^{-1}+unconstrained

Table 2: Most similar contexts to the prototype $\hat{H}$ learned by the Concat model.

terns are the "including" pattern ("animals including cats") and "many" pattern ("many animals"). Although we list only the six most similar context items for the data sets, we find similar contexts continue to dominate the list for the next 30-50 items. Taken together, it is remarkable that the model identified these patterns using only distributional vectors and only the positive/negative example pairs. However, the reader should note these are not true Hearst patterns: Hearst patterns explicitly relate a hypernym and hyponym using an exact pattern match of a *single* co-occurrence. On the other hand, these *H-features* are *aggregate indicators* of hypernymy across a large corpus.

These learned features are much more interpretable than those found in the analysis of prior work like Roller et al. (2014) and Levy et al. (2015). Roller et al. (2014) found no signals of H-features in their analysis of one classifier, but their model was focused on *bag-of-words* distributional vectors, which perform significantly worse on the task. Levy et al. (2015) also performed an analysis of lexical entailment classifiers, and found weak signals like "such" and "of" appearing as prominent contexts in their classifier, giving an early hint of H-feature detectors, but not to such an overwhelming degree as we see in this work. Critically, their analysis focused on a classifier trained on high-dimensional, *sparse* vectors, rather than focusing on *context embeddings* as we do. By using these sparse vectors, their model was unable to generalize across similar contexts. Additionally, their model did not make use of *collapsed dependencies*, making features like "such" much weaker signals of entailment and therefore less dominant during analysis.

Among these remarkable lists, the LEDS and TM14 data sets stand out for having much fewer "such as" patterns compared to BLESS and Medical. The reason for this is explained by the construction of the data sets: since LEDS contains the same words used as both positive and negative examples, the classifier has a hard time picking out clear signal. The TM14 data set, however, does not contain any such negative examples.

We hypothesize the TM14 data set contains too many diverse and mutually exclusive forms of lexical entailment, like instrument-goal (e.g. "honey" $\rightarrow$ "sweetness"). To test this, we retrained the model with only hypernymy as positive examples, and all other relations as negative. We find that "such as" type patterns become top features, but also some interesting data specific features, like "retailer of [clothes]". Examining the data shows it contains many consumer goods, like "beverage" or "clothes", which explains these features.

5 Proposed Model

As we saw in the previous section, Concat only acts as a sort of H-feature detector for whether H is a prototypical hypernym, but does not actually infer the relationship between H and w. Nonetheless, this is powerful behavior which should still be used in combination with the insights of other models like Ksim and Asym. To this end, we propose a novel model which exploits Concat's H-feature detector behavior, extends its modeling power, and adds two other types of evidence proposed in the literature: overall similarity, and distributional inclusion.

Our model works through an iterative procedure similar to Principal Component Analysis (PCA). Each iteration repeatedly trains a Concat classifier under the assumption that it acts as an H-feature detector, and then explicitly discards this information from the distributional vectors. By training a new H-feature detector on these modified distributional vectors, we can find *additional* features indicative of entailment which were missed by the first classifier. The entire procedure is iteratively repeated similar

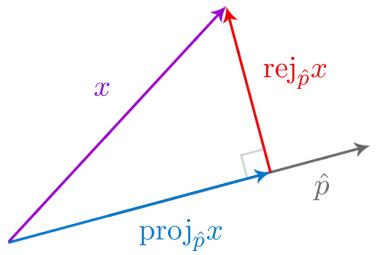


Figure 1: A vector $\hat{p}$ is used to break x into two orthogonal components, its projection and the rejection over $\hat{p}$.

to how in Principal Component Analysis, the second principal component is computed after the first principal component has been removed from the data.

The main insight is that after training some H-feature detector using Concat, we can *remove* this prototype from the distributional vectors through the use of *vector projection*. Formally, the vector projection of x onto a vector $\hat{p}$, $\text{proj}_{\hat{p}}(x)$ finds the *component* of x which is in the direction of $\hat{p}$,

$$\text{proj}_{\hat{p}}(x) = \left(\frac{x^\top \hat{p}}{\|\hat{p}\|}\right)\hat{p}.$$

Figure 1 gives a geometric illustration of the vector projection. If x forms the hypotenuse of a right triangle, $\text{proj}_{\hat{p}}(x)$ forms a leg of the triangle. This also gives rise to the *vector rejection*, which is the vector forming the third leg of the triangle. The vector rejection is orthogonal to the projection, and intuitively, is the original vector after the projection has been removed:

$$\text{rej}_{\hat{p}}(x) = x - \text{proj}_{\hat{p}}(x).$$

Using the vector rejection, we take a learned H-feature detector $\hat{p}$, and discard these features from each of the word vectors. That is, for every data point $\langle H, w\rangle$, we replace it by its vector rejection and rescale it to unit magnitude:

$$H_{i+1} = \text{rej}_{\hat{p}}(H)/\|\text{rej}_{\hat{p}}(H)\|$$
$$w_{i+1} = \text{rej}_{\hat{p}}(w)/\|\text{rej}_{\hat{p}}(w)\|$$

A new classifier trained on the $\langle H_{i+1}, w_{i+1}\rangle$ data *must* now learn a different decision plane than $\hat{p}$, as $\hat{p}$ is no longer present in any data points. This repetition of the procedure is roughly analogous to learning the second principal component of the data; we wish to classify the pairs without using any information learned from the previous iteration.

This second classifier *must* perform strictly worse than the original, otherwise the first classifier would have learned this second hyperplane. Nonetheless, it will be able to learn *new* H-feature detectors which the original classifier was unable to capture. By repeating this process, we can find several H-feature detectors, $\hat{p}_1, \ldots, \hat{p}_n$. Although the first, $\hat{p}_1$ is the best possible *single* H-feature detector, each additional H-feature detector increases the model's representational power (albeit with diminishing returns).

This procedure alone does not address the main concern of Levy et al. (2015): that these linear classifiers never actually model any connection between H and w. To address this, we explicitly *compare* H and w by extracting additional information about how H and w interact with respect to each of the H-feature detectors. This additional information is then used to train one final classifier which makes the final prediction.

Concretely, in each iteration i of the procedure, we generate a four-valued feature vector F_i, based on the H-feature detector $\hat{p}_i$. Each feature vector contains (1) the similarity of H_i and w_i (before projection); (2) the feature $\hat{p}_i$ applied to H_i; (3) the H-feature detector $\hat{p}_i$ applied to w_i; and (4) the difference of 2 and 3.

$$F_i(\langle H_i, w_i\rangle, \hat{p}_i) = \langle H_i^\top w_i, H_i^\top \hat{p}_i, w_i^\top \hat{p}_i, (H_i - w_i)^\top \hat{p}_i\rangle$$

These four "meta"-features capture all the benefits of the H-feature detector (slots 2 and 3), while still addressing Concat's issues with similarity arguments (slot 1) *and* distributional inclusion (slot 4). The final feature's relation to the DIH comes from the observation of Roller et al. (2014) that the vector difference intuitively captures whether the hypernym *includes* the hyponym.

The union of all the feature vectors $F_1, \ldots, F_n$ from repeated iteration form a $4n$-dimensional feature vector which we use as input to one final classifier which makes the ultimate decision. This classifier is trained on the same training data as each of the individual H-feature detectors, so our iterative procedure acts *only* as a method of feature extraction.

For our final classifier, we use an SVM with an RBF-kernel, though decision trees and other nonlinear classifiers also perform reasonably well. The nonlinear final classifier can be understood as doing a form of logical reasoning about the four slots: "animal" is a hypernym of "cat" because (1) they are similar words where (2) animal looks like a hypernym, but (3) cat does not, and (4) some "animal" contexts are not good "cat" contexts.

6 Experimental Setup and Evaluation

In our experiments, we use a variation of 20-fold cross validation which accounts for lexical overlap. To simplify explanation, we first explain how we generate splits for training/testing, and then afterwards introduce validation methodology.

We first pool all the words from the antecedent (LHS) side of the data into a set, and split these lexical items into 20 distinct cross-validation folds. For each fold F_i, we then use all pairs (w, H) where $w \in F_i$ as the test set pairs. That is, if "car" is in the test set fold, then "car $\rightarrow$ vehicle" and "car $\nrightarrow$ truck" will appear as test set pairs. The training set will then be *every pair* which does not contain *any* overlap with the test set; e.g. the training set will be all pairs which do not contain "car", "truck" or "vehicle" as either the antecedent or consequent. This ensures that both (1) there is zero lexical overlap between training and testing and (2) every pair is used as an item in a test fold exactly once. One quirk of this setup is that all test sets are approximately the same size, but training sizes vary dramatically.

This setup differs from those of previous works like Kruszewski et al. (2015) and Levy et al. (2015), who both use single, fixed train/test/val sets without lexical overlap. We find our setup has several advantages over fixed sets. First, we find there can be considerable variance if the train/test set is regenerated with a different random seed, indicating that multiple trials are necessary. Second, fixed setups consistently discard roughly half the data as ineligible for either training or test, as lexical items appear in many pairs. Our CV-like setup allows us to evaluate performance over every item in the data set exactly once, making a much more efficient and representative use of the original data set.

Our performance metric is F1 score. This is more representative than accuracy, as most of the data sets are heavily unbalanced. We report the mean F1 scores across all cross validation folds.

Model	LEDS	BLESS	Medical	TM14
Linear Models				
Cosine	.787	.208	.168	.676
Concat	.794	.612	.218	.693
Diff	.805	.440	.195	.665
Asym	.865	.510	.210	.671
Concat+Diff	.801	.604	.224	.703
Concat+Asym	.843	**.631**	.240	.701
Nonlinear Models				
RBF	.779	.574	.215	.705
Ksim	.893	.488	.224	**.707**
Our model	**.901**	**.631**	**.260**	.697

Table 3: Mean F1 scores for each model and data set.

6.1 Hyperparameter Optimization

In order to handle hyperparameter selection, we actually generate the test set using fold i, and use fold $i-1$ as a validation set (removing pairs which would overlap with test), and the remaining 18 folds as training (removing pairs which would overlap with test *or* validation). We select hyperparameters using grid search. For all models, we optimize over the regularization parameter $C \in \{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$, and for our proposed model, the number of iterations $n \in \{1, \ldots, 6\}$. All other hyperparameters are left as defaults provided by Scikit-Learn (Pedregosa et al., 2011), except for using balanced class weights. Without balanced class weights, several of the baseline models learn degenerate functions (e.g. always guess non-entailing).

7 Results

We compare our proposed model to several existing and alternative baselines from the literature. Namely, we include a baseline Cosine classifier, which only learns a threshold which maximizes F1 score on the training set; three linear models of prior work, Concat, Diff and Asym; and the RBF and Ksim models found to be successful in Kruszewski et al. (2015) and Levy et al. (2015). We also include two additional novel baselines, Concat+Diff and Concat+Asym, which add a notion of Distributional Inclusion into the Concat baseline, but are still linear models. We cannot include baselines like

Model	LEDS	BLESS	Medical	TM14
No Similarity	.099	.061	.034	.003
No Detectors	-.008	.136	.018	.028
No Inclusion	.010	.031	.014	.001

Table 4: Absolute decrease in mean F1 on the development sets with the different feature types ablated. Higher numbers indicate greater feature importance.

Ksim+Asym, because Ksim is based on a custom SVM kernel which is not amenable to combinations.

Table 3 the results across all four data sets for all of the listed models. Our proposed model improves significantly[4] over Concat in the LEDS, BLESS and Medical data sets, indicating the benefits of combining these aspects of similarity and distributional inclusion with the H-feature detectors of Concat. The Concat+Asym classifier also improves over the Concat baseline, further emphasizing these benefits. Our model performs approximately the same as Ksim on the LEDS and TM14 data sets (no significant difference), while significantly outperforming it on BLESS and Medical data sets.

7.1 Ablation Experiments

In order to evaluate how important each of the various F features are to the model, we also performed an ablation experiment where the classifier is *not* given the similarity (slot 1), prototype H-feature detectors (slots 2 and 3) or the inclusion features (slot 4). To evaluate the importance of these features, we fix the regularization parameter at $C = 1$, and train all ablated classifiers on each training fold with number of iterations $n = 1, \ldots, 6$. Table 4 shows the decrease (absolute difference) in performance between the full and ablated models on the development sets, so higher numbers indicate greater feature importance.

We find the similarity feature is extremely important in the LEDS, BLESS and Medical data sets, therefore reinforcing the findings of Levy et al. (2015). The similarity feature is especially important in the LEDS and BLESS data sets, where negative examples include many random pairs. The detector features are moderately important for the Medical and TM14 data sets, and critically important on BLESS, where we found the strongest evidence of Hearst patterns in the H-feature detectors. Surprisingly, the detector features are moderately *detrimental* on the LEDS data set, though this can also be understood in the data set's construction: since the negative examples are randomly shuffled positive examples, the *same* detector signal will appear in both positive and negative examples. Finally, we find the model performs somewhat robustly without the inclusion feature, but still is moderately impactful on three of the four data sets, lending further evidence to the Distributional Inclusion Hypothesis. In general, we find all three components are valuable sources of information for identifying hypernymy and lexical entailment.

[4]Bootstrap test, $p < .01$.

7.2 Analysis by Number of Iterations

In order to evaluate how the iterative feature extraction affects model performance, we fix the regularization parameter at $C = 1$, and train our model fixing the number of iterations to $n = \{1, \ldots, 6\}$. We then measure the mean F1 score across the development folds and compare to a baseline which uses only one iteration. Figure 2 shows these results across all four data sets, with the 0 line set at performance of the $n = 1$ baseline. Models above 0 benefit from the additional iterations, while models below do not.

In the figure, we see that the iterative procedure moderately improves performance LEDS, while greatly improving the scores of BLESS and TM14, but on the medical data set, additional iterations actually hurt performance. The differing curves indicate that the optimal number of iterations is very data set specific, and provides differing amounts of improvement, and therefore should be tuned carefully. The LEDS and BLESS curves indicate a sort of "sweet spot" behavior, where further iterations degrade performance.

To gain some additional insight into what is captured by the various iterations of the feature extraction procedure, we repeat the procedure from Section 4: we train our model on the entire BLESS data set using a fixed four iterations and regularization parameter. For each iteration, we compare its learned H-feature detector to the context embeddings, and report the most similar contexts for each iteration in Table 5.

The first iteration is identical to the one in Ta-

Figure 2: Performance of model on development folds by number of iterations. Plots show the improvement (absolute difference) in mean F1 over the model fixed at one iteration.

Iteration 1	Iteration 2	Iteration 3	Iteration 4
nmod:such_as+submarine	nmod:including+animal	amod+free-swimming	advcl+crown
nmod:such_as+ship	nmod:including+snail	nmod:including^{-1}+thing	advcl+victorious
nmod:such_as+seal	nmod:including+insect	nsubj^{-1}+scarcer	nsubj+eaters
nmod:such_as+plane	nmod:such_as+crustacean	nsubj^{-1}+pupate	nsubj+kaine
nmod:such_as+rack	nmod:such_as+mollusc	nmod:such_as+mollusc	nmod:at+finale
nmod:such_as+rope	nmod:such_as+insect	nmod:of^{-1}+value	nsubj+gowen
nmod:such_as+box	nmod:such_as+animal	nmod:as^{-1}+exhibit	nsubj+pillman

Table 5: Most similar contexts to the H-feature detector for each iteration of the PCA-like procedure. This model was trained on all data of BLESS. The first and second iterations contain clear Hearst patterns, while the third and fourth contain some data-specific and non-obvious signals.

ble 2, as expected. The second iteration includes many H-features not picked up by the first iteration, mostly those of the form "X including Y". The third iteration picks up some data set specific signal, like "free-swimming [animal]" and "value of [computer]", and so on. By the fourth iteration, the features no longer exhibit any obvious Hearst patterns, perhaps exceeding the sweet spot we observed in Figure 2. Nonetheless, we see how multiple iterations of the procedure allows our model to capture many more useful features than a single Concat classifier on its own.

8 Conclusion

We considered the task of detecting lexical entailment using distributional vectors of word meaning. Motivated by the fact that the Concat classifier acts as a strong baseline in the literature, we proposed a novel interpretation of the model's hyperplane. We found the Concat classifier overwhelmingly acted as a feature detector which automatically identifies Hearst Patterns in the distributional vectors.

We proposed a novel model that embraces these H-feature detectors fully, and extends their modeling power through an iterative procedure similar to Principal Component Analysis. In each iteration of the procedure, an H-feature detector is learned, and then removed from the data, allowing us to identify several different kinds of Hearst Patterns in the data. Our final model combines these H-feature detectors with measurements of general similarity and Distributional Inclusion, in order to integrate the strengths of different models in prior work. Our model matches or exceeds the performance of prior work, both on hypernymy detection and general lexical entailment.

Acknowledgments

The authors would like to thank I. Beltagy, Vered Shwartz, Subhashini Venugopalan, and the reviewers for their helpful comments and suggestions. This research was supported by the NSF grant IIS 1523637. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results.

References

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 2012 Conference of the European Chapter of the Association for Computational Linguists*, pages 23–32, Avignon, France.

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

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 2014 Annual Meeting of the Association for Computational Linguistics*, pages 1199–1209, Baltimore, Maryland.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 1992 Conference on Computational Linguistics*, pages 539–545, Nantes, France.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389, 10.

Germán Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving Boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *The First Joint Conference on Lexical and Computational Semantics*, pages 75–79, Montréal, Canada.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for Open IE propositions. In *Proceedings of the 2014 Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the First Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 1025–1036, Dublin, Ireland.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, Paris, France.

Enrico Santus. 2013. SLQS: An entropy measure. Master's thesis, University of Pisa.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*, pages 1297–1304.

Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1671–1682, Berlin, Germany, August.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 2004 International Conference on Computational Linguistics*, pages 1015–1021, Geneva, Switzerland.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 2249–2259, Dublin, Ireland.

Maayan Zhitomirsky-Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 2005 Annual Meeting of the Association for Computational Linguistics*, pages 107–114, Ann Arbor, Michigan.

SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity

Daniela Gerz[1], **Ivan Vulić**[1], **Felix Hill**[1], **Roi Reichart**[2], and **Anna Korhonen**[1]

[1]Language Technology Lab, DTAL, University of Cambridge
[2]Faculty of Industrial Engineering and Management, Technion, IIT
[1]{dsg40,iv250,fh295,alk23}@cam.ac.uk
[2]roiri@ie.technion.ac.il

Abstract

Verbs play a critical role in the meaning of sentences, but these ubiquitous words have received little attention in recent distributional semantics research. We introduce SimVerb-3500, an evaluation resource that provides human ratings for the similarity of 3,500 verb pairs. SimVerb-3500 covers all normed verb types from the USF free-association database, providing at least three examples for every VerbNet class. This broad coverage facilitates detailed analyses of how syntactic and semantic phenomena together influence human understanding of verb meaning. Further, with significantly larger development and test sets than existing benchmarks, SimVerb-3500 enables more robust evaluation of representation learning architectures and promotes the development of methods tailored to verbs. We hope that SimVerb-3500 will enable a richer understanding of the diversity and complexity of verb semantics and guide the development of systems that can effectively represent and interpret this meaning.

1 Introduction

Verbs are famously both complex and variable. They express the semantics of an event as well the relational information among participants in that event, and they display a rich range of syntactic and semantic behaviour (Jackendoff, 1972; Gruber, 1976; Levin, 1993). Verbs play a key role at almost every level of linguistic analysis. Information related to their predicate argument structure can benefit many NLP tasks (e.g. parsing, semantic role labelling, information extraction) and applications (e.g. machine translation, text mining) as well as research on human language acquisition and processing (Korhonen, 2010). Precise methods for representing and understanding verb semantics will undoubtedly be necessary for machines to interpret the meaning of sentences with similar accuracy to humans.

Numerous algorithms for acquiring word representations from text and/or more structured knowledge bases have been developed in recent years (Mikolov et al., 2013; Pennington et al., 2014; Faruqui et al., 2015). These representations (or *embeddings*) typically contain powerful features that are applicable to many language applications (Collobert and Weston, 2008; Turian et al., 2010). Nevertheless, the predominant approaches to distributed representation learning apply a single learning algorithm and representational form for all words in a vocabulary. This is despite evidence that applying different learning algorithms to word types such as nouns, adjectives and verbs can significantly increase the ultimate usefulness of representations (Schwartz et al., 2015).

One factor behind the lack of more nuanced word representation learning methods is the scarcity of satisfactory ways to evaluate or analyse representations of particular word types. Resources such as MEN (Bruni et al., 2014), Rare Words (Luong et al., 2013) and SimLex-999 (Hill et al., 2015) focus either on words from a single class or small samples of different word types, with automatic approaches already reaching or surpassing the inter-annotator agreement ceiling. Consequently, for word classes such as *verbs*, whose semantics is critical for language understanding, it is practically impossible to achieve statistically robust analyses and comparisons between different

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2173–2182,
Austin, Texas, November 1-5, 2016.

representation learning architectures.

To overcome this barrier to verb semantics research, we introduce *SimVerb-3500* – an extensive intrinsic evaluation resource that is unprecedented in both size and coverage. SimVerb-3500 includes 827 verb types from the University of South Florida Free Association Norms (USF) (Nelson et al., 2004), and at least 3 member verbs from each of the 101 top-level VerbNet classes (Kipper et al., 2008). This coverage enables researchers to better understand the complex diversity of syntactic-semantic verb behaviours, and provides direct links to other established semantic resources such as WordNet (Miller, 1995) and PropBank (Palmer et al., 2005). Moreover, the large standardised development and test sets in SimVerb-3500 allow for principled tuning of hyperparameters, a critical aspect of achieving strong performance with the latest representation learning architectures.

In § 2, we discuss previous evaluation resources targeting verb similarity. We present the new SimVerb-3500 data set along with our design choices and the pair selection process in § 3, while the annotation process is detailed in § 4. In § 5 we report the performance of a diverse range of popular representation learning architectures, together with benchmark performance on existing evaluation sets. In § 6, we show how SimVerb-3500 enables a variety of new linguistic analyses, which were previously impossible due to the lack of coverage and scale in existing resources.

2 Related Work

A natural way to evaluate representation quality is by judging the similarity of representations assigned to similar words. The most popular evaluation sets at present consist of word pairs with similarity ratings produced by human annotators.[1] Nevertheless, we find that all available datasets of this kind are insufficient for judging verb similarity due to their small size or narrow coverage of verbs.

In particular, a number of word pair evaluation sets are prominent in the distributional semantics literature.

Representative examples include RG-65 (Rubenstein and Goodenough, 1965) and WordSim-353 (Finkelstein et al., 2002; Agirre et al., 2009) which are small (65 and 353 word pairs, respectively). Larger evaluation sets such as the Rare Words evaluation set (Luong et al., 2013) (2034 word pairs) and the evaluations sets from Silberer and Lapata (2014) are dominated by noun pairs and the former also focuses on low-frequency phenomena. Therefore, these datasets do not provide a representative sample of verbs (Hill et al., 2015).

Two datasets that do focus on verb pairs to some extent are the data set of Baker et al. (2014) and Simlex-999 (Hill et al., 2015). These datasets, however, still contain a limited number of verb pairs (134 and 222, respectively), making them unrepresentative of the rich variety of verb semantic phenomena.

In this paper we provide a remedy for this problem by presenting a more comprehensive and representative verb pair evaluation resource.

3 The SimVerb-3500 Data Set

In this section, we discuss the design principles behind SimVerb-3500. We first demonstrate that a new evaluation resource for verb similarity is a necessity. We then describe how the final verb pairs were selected with the goal to be representative, that is, to guarantee a wide coverage of two standard semantic resources: USF and VerbNet.

3.1 Design Motivation

Hill et al. (2015) argue that comprehensive high-quality evaluation resources have to satisfy the following three criteria: *(C1) Representative* (the resource covers the full range of concepts occurring in natural language); *(C2) Clearly defined* (it clearly defines the annotated relation, e.g., similarity); (C3) *Consistent and reliable* (untrained native speakers must be able to quantify the target relation consistently relying on simple instructions).

Building on the same annotation guidelines as Simlex-999 that explicitly targets similarity, we ensure that criteria C2 and C3 are satisfied. However, even SimLex, as the most extensive evaluation resource for verb similarity available at present, is still of limited size, spanning only 222 verb pairs and 170

[1] In some existing evaluation sets pairs are scored for relatedness which has some overlap with similarity. SimVerb-3500 focuses on similarity as this is a more focused semantic relation that seems to yield a higher agreement between human annotators. For a broader discussion see (Hill et al., 2015).

distinct verb lemmas in total. Given that 39 out of the 101 top-level VerbNet classes are not represented at all in SimLex, while 20 classes have only one member verb,[2] one may conclude that the criterion C1 is not at all satisfied with current resources.

There is another fundamental limitation of all current verb similarity evaluation resources: automatic approaches have reached or surpassed the inter-annotator agreement ceiling. For instance, while the average pairwise correlation between annotators on SL-222 is Spearman's ρ correlation of 0.717, the best performing automatic system reaches $\rho = 0.727$ (Mrkšić et al., 2016). SimVerb-3500 does not inherit this anomaly (see Tab. 2) and demonstrates that there still exists an evident gap between the human and system performance.

In order to satisfy C1-C3, the new SimVerb-3500 evaluation set contains similarity ratings for 3,500 *verb pairs*, containing 827 verb types in total and 3 member verbs for each top-level VerbNet class. The rating scale goes from 0 (not similar at all) to 10 (synonymous). We employed the SimLex-999 annotation guidelines. In particular, we instructed annotators to give low ratings to antonyms, and to distinguish between similarity and relatedness. Pairs that are related but not similar (e.g., *to snore / to snooze*, *to walk / to crawl*) thus have a fairly low rating. Several example pairs are provided in Tab. 1.

3.2 Choice of Verb Pairs and Coverage

To ensure a wide coverage of a variety of syntactico-semantic phenomena (C1), the choice of verb pairs is steered by two standard semantic resources available online: (1) the USF norms data set[3] (Nelson et al., 2004), and (2) the VerbNet verb lexicon[4] (Kipper et al., 2004; Kipper et al., 2008).

The **USF** norms data set (further USF) is the largest database of free association collected for English. It was generated by presenting human subjects with one of $5,000$ cue concepts and asking them to write the first word coming to mind that is associated with that concept. Each cue concept c was normed in this way by over 10 participants, resulting in a set of associates a for each cue, for a total of over $72,000$ (c, a) pairs. For each such pair, the proportion of participants who produced associate a when presented with cue c can be used as a proxy for the strength of association between the two words.

Pair	Rating
to reply / to respond	9.79
to snooze / to nap	8.80
to cook / to bake	7.80
to participate / to join	5.64
to snore / to snooze	4.15
to walk / to crawl	2.32
to stay / to leave	0.17
to snooze / to happen	0.00

Table 1: Example verb pairs from SimVerb-3500.

The norming process guarantees that two words in a pair have a degree of semantic association which correlates well with semantic relatedness and similarity. Sampling from the USF set ensures that both related but non-similar pairs (e.g., *to run / to sweat*) as well as similar pairs (e.g., *to reply / to respond*) are represented in the final list of pairs. Further, the rich annotations of the output USF data (e.g., concreteness scores, association strength) can be directly combined with the SimVerb-3500 similarity scores to yield additional analyses and insight.

VerbNet (**VN**) is the largest online verb lexicon currently available for English. It is hierarchical, domain-independent, and broad-coverage. VN is organised into verb classes extending the classes from Levin (1993) through further refinement to achieve syntactic and semantic coherence among class members. According to the official VerbNet guidelines,[5] "Verb Classes are numbered according to shared semantics and syntax, and classes which share a top-level number (9-109) have corresponding semantic relationships." For instance, all verbs from the top-level Class 9 are labelled "Verbs of Putting", all verbs from Class 30 are labelled "Verbs of Perception", while Class 39 contains "Verbs of Ingesting".

Among others, three basic types of information are covered in VN: (1) verb subcategorization frames (SCFs), which describe the syntactic realization of the predicate-argument structure (e.g. *The window broke*), (2) selectional preferences (SPs), which capture the semantic preferences verbs have for their

[2]Note that verbs in VerbNet are soft clustered, and one verb type may be associated with more than one class. When computing coverage, we assume that such verbs attribute to counts of all their associated classes.

[3]http://w3.usf.edu/FreeAssociation/

[4]http://verbs.colorado.edu/verb-index/

[5]http://verbs.colorado.edu/verb-index/VerbNet_Guidelines.pdf

arguments (e.g. *a breakable physical object* broke) and (3) lexical-semantic verb classes (VCs) which provide a shared level of abstraction for verbs similar in their (morpho-)syntactic and semantic properties (e.g. *BREAK verbs*, sharing the VN class 45.1, and the top-level VN class 45).[6] The basic overview of the VerbNet structure already suggests that measuring verb similarity is far from trivial as it revolves around a complex interplay between various semantic and syntactic properties.

The wide coverage of VN in SimVerb-3500 assures the wide coverage of distinct verb groups/classes and their related linguistic phenomena. Finally, VerbNet enables further connections of SimVerb-3500 to other important lexical resources such as FrameNet (Baker et al., 1998), WordNet (Miller, 1995), and PropBank (Palmer et al., 2005) through the sets of mappings created by the SemLink project initiative (Loper et al., 2007).[7]

Sampling Procedure We next sketch the complete sampling procedure which resulted in the final set of 3500 distinct verb pairs finally annotated in a crowdsourcing study (§ 4).

(Step 1) We extracted all possible verb pairs from USF based on the associated POS tags available as part of USF annotations. To ensure that semantic association between verbs in a pair is not accidental, we then discarded all such USF pairs that had been associated by 2 or less participants in USF.

(Step 2) We then manually cleaned and simplified the list of pairs by removing all pairs with multi-word verbs (e.g., *quit / give up*), all pairs that contained the non-infinitive form of a verb (e.g., *accomplished / finished*, *hidden / find*), removing all pairs containing at least one auxiliary verb (e.g., *must / to see, must / to be*). The first two steps resulted in 3,072 USF-based verb pairs.

(Step 3) After this stage, we noticed that several top-level VN classes are not part of the extracted set. For instance, 5 VN classes did not have any member verbs included, 22 VN classes had only 1 verb, and 6 VN classes had 2 verbs included in the current set.

We resolved the VerbNet coverage issue by sampling from such 'under-represented' VN classes directly. Note that this step is not related to USF at all. For each such class we sampled additional verb types until the class was represented by 3 or 4 member verbs (chosen randomly).[8] Following that, we sampled at least 2 verb pairs for each previously 'under-represented' VN class by pairing 2 member verbs from each such class. This procedure resulted in 81 additional pairs, now 3,153 in total.

(Step 4) Finally, to complement this set with a sample of entirely unassociated pairs, we followed the SimLex-999 setup. We paired up the verbs from the 3,153 associated pairs at random. From these random parings, we excluded those that coincidentally occurred elsewhere in USF (and therefore had a degree of association). We sampled the remaining 347 pairs from this resulting set of unassociated pairs.

(Output) The final SimVerb-3500 data set contains 3,500 verb pairs in total, covering all associated verb pairs from USF, and (almost) all top-level VerbNet classes. All pairs were manually checked post-hoc by the authors plus 2 additional native English speakers to verify that the final data set does not contain unknown or invalid verb types.

Frequency Statistics The 3,500 pairs consist of 827 distinct verbs. 29 top-level VN classes are represented by 3 member verbs, while the three most represented classes cover 79, 85, and 93 member verbs. 40 verbs are not members of any VN class.

We performed an initial frequency analysis of SimVerb-3500 relying on the BNC counts available online (Kilgarriff, 1997).[9] After ranking all BNC verbs according to their frequency, we divided the list into quartiles: Q1 (most frequent verbs in BNC) - Q4 (least frequent verbs in BNC). Out of the 827 SimVerb-3500 verb types, 677 are contained in Q1, 122 in Q2, 18 in Q3, 4 in Q4 (*to enroll, to hitchhike, to implode, to whelp*), while 6 verbs are not covered in the BNC list. 2,818 verb pairs contain Q1 verbs, while there are 43 verb pairs with both verbs not in Q1. Further empirical analyses are provided in § 6.[10]

[6] https://verbs.colorado.edu/verb-index/vn/break-45.1.php

[7] https://verbs.colorado.edu/semlink/

[8] The following three VN classes are exceptions: (1) Class 56, consisting of words that are dominantly tagged as nouns, but can be used as verbs exceptionally (e.g., *holiday, summer, honeymoon*); (2) Class 91, consisting of 2 verbs (*count, matter*); (3) Class 93, consisting of 2 single word verbs (*adopt, assume*).

[9] https://www.kilgarriff.co.uk/bnc-readme.html

[10] Annotations such as VerbNet class membership, relations between WordNet synsets of each verb, and frequency statistics are available as supplementary material.

4 Word Pair Scoring

We employ the Prolific Academic (PA) crowdsourcing platform,[11] an online marketplace very similar to Amazon Mechanical Turk and to CrowdFlower.

4.1 Survey Structure

Following the SimLex-999 annotation guidelines, we had each of the 3500 verb pairs rated by at least 10 annotators. To distribute the workload, we divided the 3500 pairs into 70 tranches, with 79 pairs each. Out of the 79 pairs, 50 are unique to one tranche, while 20 manually chosen pairs are in all tranches to ensure consistency. The remaining 9 are duplicate pairs displayed to the same participant multiple times to detect inconsistent annotations.

Participants see 7-8 pairs per page. Pairs are rated on a scale of 0-6 by moving a slider. The first page shows 7 pairs, 5 unique ones and 2 from the consistency set. The following pages are structured the same but display one extra pair from the previous page. Participants are explicitly asked to give these duplicate pairs the same rating. We use them as quality control so that we can identify and exclude participants giving several inconsistent answers.

Checkpoint Questions The survey contains three control questions in which participants are asked to select the most similar pair out of three choices. For instance, the first checkpoint is: *Which of these pairs of words is the *most* similar? 1. to run / to jog 2. to run / to walk 3. to jog / to sweat.* One checkpoint occurs right after the instructions and the other two later in the survey. The purpose is to check that annotators have understood the guidelines and to have another quality control measure for ensuring that they are paying attention throughout the survey. If just one of the checkpoint questions is answered incorrectly, the survey ends immediately and all scores from the annotator in question are discarded.

Participants 843 raters participated in the study, producing over 65,000 ratings. Unlike other crowdsourcing platforms, PA collects and stores detailed demographic information from the participants upfront. This information was used to carefully select the pool of eligible participants. We restricted the pool to native English speakers with a 90% approval rate (maximum rate on PA), of age 18-50, born and currently residing in the US (45% out of 843 raters), UK (53%), or Ireland (2%). 54% of the raters were female and 46% male, with the average age of 30. Participants took 8 minutes on average to complete the survey containing 79 questions.

[11] https://prolific.ac/ (We chose PA for logistic reasons.)

4.2 Post-Processing

We excluded ratings of annotators who *(a)* answered one of the checkpoint questions incorrectly (75% of exclusions); *(b)* did not give equal ratings to duplicate pairs; *(c)* showed suspicious rating patterns (e.g., randomly alternating between two ratings or using one single rating throughout). The final acceptance rate was 84%. We then calculated the average of all ratings from the accepted raters (≥ 10) for each pair. The score was finally scaled linearly from the 0-6 to the 0-10 interval as in (Hill et al., 2015).

5 Analysis

Inter-Annotator Agreement We employ two measures. **IAA-1 (pairwise)** computes the average pairwise Spearman's ρ correlation between any two raters – a common choice in previous data collection in distributional semantics (Padó et al., 2007; Reisinger and Mooney, 2010a; Silberer and Lapata, 2014; Hill et al., 2015).

A complementary measure would smooth individual annotator effects. For this aim, our **IAA-2 (mean)** measure compares the average correlation of a human rater with the average of all the other raters. SimVerb-3500 obtains $\rho = 0.84$ (IAA-1) and $\rho = 0.86$ (IAA-2), a very good agreement compared to other benchmarks (see Tab. 2).

Vector Space Models We compare the performance of prominent representation models on SimVerb-3500. We include: (1) unsupervised models that learn from distributional information in text, including the skip-gram negative-sampling model (*SGNS*) with various contexts (*BOW = bag of words; DEPS = dependency contexts*) as in Levy and Goldberg (2014), the symmetric-pattern based vectors by Schwartz et al. (2015), and count-based PMI-weighted vectors (Baroni et al., 2014); (2) Models that rely on linguistic hand-crafted resources or curated knowledge bases. Here, we use sparse binary vectors built from linguistic resources (*Non-*

Eval set	IAA-1	IAA-2	ALL	TEXT
WSIM (203)	0.67	0.65	0.79 SGNS-BOW	0.79 SGNS-BOW
SIMLEX (999)	0.67	0.78	0.74 Paragram+CF	0.56 SymPat+SGNS
SL-222 (222)	0.72	-	0.73 Paragram+CF	0.58 SymPat
SIMVERB (3500)	0.84	0.86	0.63 Paragram+CF	0.36 SGNS-DEPS

Table 2: An overview of word similarity evaluation benchmarks. ALL is the current best reported score on each data set across all models (including the models that exploit curated knowledge bases and hand-crafted lexical resources, see supplementary material). TEXT denotes the best reported score for a model that learns solely on the basis of distributional information. All scores are Spearman's ρ correlations.

Distributional, (Faruqui and Dyer, 2015)), and vectors fine-tuned to a paraphrase database (*Paragram*, (Wieting et al., 2015)) further refined using linguistic constraints (*Paragram+CF*, (Mrkšić et al., 2016)). Descriptions of these models are in the supplementary material.

Comparison to SimLex-999 (SL-222) 170 pairs from SL-222 also appear in SimVerb-3500. The correlation between the two data sets calculated on the shared pairs is $\rho = 0.91$. This proves, as expected, that the ratings are consistent across the two data sets.

Tab. 3 shows a comparison of models' performance on SimVerb-3500 against SL-222. Since the number of evaluation pairs may influence the results, we ideally want to compare sets of equal size for a fair comparison. Picking one random subset of 222 pairs would bias the results towards the selected pairs, and even using 10-fold cross-validation we found variations up to 0.05 depending on which subsets were used. Therefore, we employ a 2-level 10-fold cross-validation where new random subsets are picked in each iteration of each model. The numbers reported as *CV-222* are averages of these ten 10-fold cross-validation runs. The reported results come very close to the correlation on the full data set for all models.

Most models perform much better on SL-222, especially those employing additional databases or linguistic resources. The performance of the best scoring Paragram+CF model is even on par with the IAA-1 of 0.72. The same model obtains the highest score on SV-3500 ($\rho = 0.628$), with a clear gap to IAA-1 of 0.84. We attribute these differences in performance largely to SimVerb-3500 being a more extensive and diverse resource in terms of verb pairs.

Development Set A common problem in scored word pair datasets is the lack of a standard split to development and test sets. Previous works often optimise models on the entire dataset, which leads to overfitting (Faruqui et al., 2016) or use custom splits, e.g., 10-fold cross-validation (Schwartz et al., 2015), which make results incomparable with others. The lack of standard splits stems mostly from small size and poor coverage – issues which we have solved with SimVerb-3500.

Our development set contains 500 pairs, selected to ensure a broad coverage in terms of similarity ranges (i.e., non-similar and highly similar pairs, as well as pairs of medium similarity are represented) and top-level VN classes (each class is represented by at least 1 member verb). The test set includes the remaining 3,000 verb pairs. The performances of representation learning architectures on the dev and test sets are reported in Tab. 3. The ranking of models is identical on the test and the full SV-3500 set, with slight differences in ranking on the development set.

6 Evaluating Subsets

The large coverage and scale of SimVerb-3500 enables model evaluation based on selected criteria. In this section, we showcase a few example analyses.

Frequency In the first analysis, we select pairs based on their lemma frequency in the BNC corpus and form three groups, with 390-490 pairs in each group (Fig. 1). The results from Fig. 1 suggest that the performance of all models improves as the frequency of the verbs in the pair increases, with much steeper curves for the purely distributional models (e.g., SGNS and SymPat). The non-distributional non data-driven model of Faruqui and Dyer (2015) is only slightly affected by frequency.

WordNet Synsets Intuitively, representations for verbs with more diverse usage patterns are more difficult to learn with statistical models. To examine this hypothesis, we resort to WordNet (Miller, 1995), where different semantic usages of words are listed as so-called *synsets*. Fig. 2 shows a clear downward trend for all models, confirming that polysemous

Model	SV-3500	CV-222	SL-222	DEV-500	TEST-3000
SGNS-BOW-PW (d=300)	0.274	0.279	0.328	0.333	0.265
SGNS-DEPS-PW (d=300)	0.313	0.314	0.390	0.401	0.304
SGNS-UDEPS-PW (d=300)	0.259	0.262	0.347	0.313	0.250
SGNS-BOW-8B (d=500)	0.348	0.343	0.307	0.378	0.350
SGNS-DEPS-8B (d=500)	0.356	0.347	0.385	0.389	0.351
SYMPAT-8B (d=500)	0.328	0.336	0.544	0.276	0.347
COUNT-SVD (d=500)	0.196	0.200	0.059	0.259	0.186
NON-DISTRIBUTIONAL	0.596	0.596	0.689	0.632	0.600
PARAGRAM (d=25)	0.418	0.432	0.531	0.443	0.433
PARAGRAM (d=300)	0.540	0.528	0.590	0.525	0.537
PARAGRAM+CF (d=300)	0.628	0.625	0.727	0.611	0.624

Table 3: Evaluation of state-of-the representation learning models on the full SimVerb-3500 set (SV-3500), the Simlex-999 verb subset containing 222 pairs (SL-222), cross-validated subsets of 222 pairs from SV-3500 (CV-222), and the SimVerb-3500 development (DEV-500) and test set (TEST-3000).

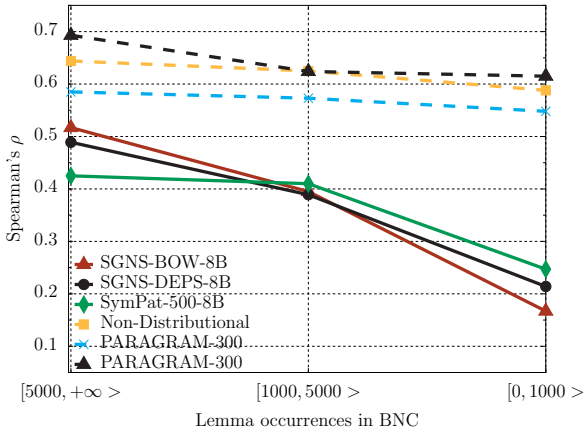


Figure 1: Subset-based evaluation, where subsets are created based on the frequency of verb lemmas in the BNC corpus. Each of the three frequency groups contains 390-490 verb pairs. To be included in each group it is required that both verbs in a pair are contained in the same frequency interval (x axis).

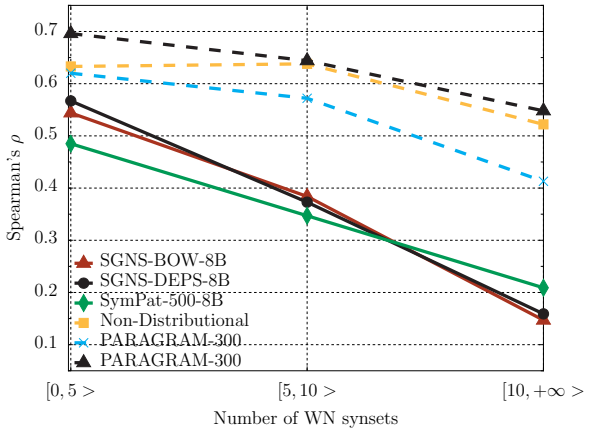


Figure 2: Subset-based evaluation, where subsets are created based on the number of synsets in WordNet (x axis). To be included in each subset it is required that both verbs in a pair have the number of synsets in the same interval.

verbs are more difficult for current verb representation models. Nevertheless, approaches which use additional information beyond corpus co-occurrence are again more robust. Their performance only drops substantially for verbs with more than 10 synsets, while the performance of other models deteriorates already when tackling verbs with more than 5 synsets.

VerbNet Classes Another analysis enabled by SimVerb-3500 is investigating the connection between VerbNet classes and human similarity judgments. We find that verbs in the same top-level VerbNet class are often not assigned high similarity score. Out of 1378 pairs where verbs share the top-level VerbNet class, 603 have a score lower than 5. Tab. 4 reports scores per VerbNet class. When a verb belongs to multiple classes, we count it for each class (see Footnote 2). We run the analysis on the five largest VN classes, each with more than 100 pairs with paired verbs belonging to the same class.

The results indicate clear differences between classes (e.g., Class 31 vs Class 51), and suggest that further developments in verb representation learning should also focus on constructing specialised representations at the finer-grained level of VN classes.

Lexical Relations SimVerb-3500 contains relation annotations (e.g., *antonyms*, *synonyms*, *hyper-/hyponyms*, *no relation*) for all pairs extracted automatically from WordNet. Evaluating per-relation subsets, we observe that some models draw their strength from good performance across different re-

Model	#13	#31	#37	#45	#51
SGNS-BOW-8B	0.210	0.308	0.352	0.270	0.170
SGNS-DEPS-8B	0.289	0.270	0.306	0.238	0.225
SYMPAT-8B (d=500)	0.171	0.320	0.143	0.195	0.113
NON-DISTR	0.571	0.483	0.372	0.501	0.499
PARAGRAM (d=300)	0.571	0.504	0.567	0.531	0.387
PARAGRAM+CF	0.735	0.575	0.666	0.622	0.614

Table 4: Spearman's ρ correlation between human judgments and model's cosine similarity by VerbNet Class. We chose classes #13 *Verbs of Change of Possession*, #31 *Verbs of Psychological State*, #37 *Verbs of Communication*, #45 *Verbs of Change of State*, and #51 *Verbs of Motion* as examples. All are large classes with more than 100 pairs each, and the frequencies of member verbs are distributed in a similar way.

Model	NR	SYN	HYP
SGNS-BOW-PW (d=300)	0.096	0.288	0.292
SGNS-DEPS-PW (d=300)	0.132	0.290	0.336
SGNS-BOW-8B (d=500)	0.292	0.273	0.338
SGNS-DEPS-8B (d=500)	0.157	0.323	0.378
SYMPAT-8B-DENSE (d=300)	0.225	0.182	0.265
SYMPAT-8B-DENSE (d=500)	0.248	0.260	0.251
NON-DISTRIBUTIONAL	0.126	0.379	0.488
PARAGRAM (d=300)	0.254	0.356	0.439
PARAGRAM+CF (d=300)	0.250	0.417	0.475

Table 5: Spearman's ρ correlation between human judgments and model's cosine similarity based on pair relation type. Relations are based on WordNet, and included in the dataset. The classes are of different size, 373 pairs with no relation (*NR*), 306 synonym (*SYN*) pairs, and 800 hyper/hyponym (*HYP*) pairs. Frequencies of member verbs are distributed in a similar way.

lations. Others have low performance on these pairs, but do very well on synonyms and hyper-/hyponyms. Selected results of this analysis are in Tab. 5.[12]

Human Agreement Motivated by the varying performance of computational models regarding frequency and ambiguous words with many synsets, we analyse what disagreement effects may be captured in human ratings. We therefore compute the average standard deviation of ratings per subset: $avgstdd(S) = \frac{1}{n}\sum_{p \in S} \sigma(r_p)$, where *S* is one subset of pairs, *n* is the number of pairs in this subset, *p* is one pair, and r_p are all human ratings for this pair.

[12] Evaluation based on Spearman's ρ may be problematic with certain categories, e.g., with antonyms. It evaluates pairs according to their ranking; for antonyms the ranking is arbitrary - every antonym pair should have a very low rating, hence they are not included in Tab. 5. A similar effect occurs with highly ranked synonyms, but to a much lesser degree than with antonyms.

While the standard deviation of ratings is diverse for individual pairs, overall the average standard deviations per subset are almost identical. For both the frequency and the WordNet synset analyses it is around $\approx$1.3 across all subsets, and with only little difference for the subsets based on VerbNet. The only subsets where we found significant variations is the grouping by relations, where ratings tend to be more similar especially on antonyms (*0.86*) and pairs with no relation (*0.92*), much less similar on synonyms (*1.34*) and all other relations ($\approx$*1.4*). These findings suggest that humans are much less influenced by frequency or polysemy in their understanding of verb semantics compared to computational models.

7 Conclusions

SimVerb-3500 is a verb similarity resource for analysis and evaluation that will be of use to researchers involved in understanding how humans or machines represent the meaning of verbs, and, by extension, scenes, events and full sentences. The size and coverage of syntactico-semantic phenomena in SimVerb-3500 makes it possible to compare the strengths and weaknesses of various representation models via statistically robust analyses on specific word classes.

To demonstrate the utility of SimVerb-3500, we conducted a selection of analyses with existing representation-learning models. One clear conclusion is that distributional models trained on raw text (e.g. SGNS) perform very poorly on low frequency and highly polysemous verbs. This degradation in performance can be partially mitigated by focusing models on more principled distributional contexts, such as those defined by symmetric patterns. More generally, the finding suggests that, in order to model the diverse spectrum of verb semantics, we may require algorithms that are better suited to fast learning from few examples (Lake et al., 2011), and have some flexibility with respect to sense-level distinctions (Reisinger and Mooney, 2010b; Vilnis and McCallum, 2015). In future work we aim to apply such methods to the task of verb acquisition.

Beyond the preliminary conclusions from these initial analyses, the benefit of SimLex-3500 will become clear as researchers use it to probe the relationship between architectures, algorithms and representation quality for a wide range of verb classes. Better under-

standing of how to represent the full diversity of verbs should in turn yield improved methods for encoding and interpreting the facts, propositions, relations and events that constitute much of the important information in language.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909).

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL-HLT*, pages 19–27.

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

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *EMNLP*, pages 278–289.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *ACL*, pages 464–469.

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

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *CoRR*, abs/1605.02276.

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

Jeffrey Gruber. 1976. *Lexical structure in syntax and semantics*. North-Holland Pub. Co.

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

Ray S. Jackendoff. 1972. *Semantic interpretation in generative grammar*. MIT Press.

Adam Kilgarriff. 1997. Putting frequencies in the dictionary. *International Journal of Lexicography*, 10(2):135–155.

Karin Kipper, Benjamin Snyder, and Martha Palmer. 2004. Extending a verb-lexicon using a semantically annotated corpus. In *LREC*, pages 1557–1560.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resource and Evaluation*, 42(1):21–40.

Anna Korhonen. 2010. Automatic lexical classification: bridging research and practice. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 368(1924):3621–3632.

Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. One shot learning of simple visual concepts. In *CogSci*.

Beth Levin. 1993. *English verb classes and alternation, A preliminary investigation*. The University of Chicago Press.

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

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between PropBank and VerbNet. In *IWCS*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR: Workshop Papers*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*, pages 142–148.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Sebastian Padó, Ulrike Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *EMNLP-CoNLL*, pages 400–409.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

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

Joseph Reisinger and Raymond J. Mooney. 2010a. A mixture model with sharing for lexical semantics. In *EMNLP*, pages 1173–1182.

Joseph Reisinger and Raymond J Mooney. 2010b. Multi-prototype vector-space models of word meaning. In *NAACL-HTL*, pages 109–117.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

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

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*, pages 721–732.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Luke Vilnis and Andrew McCallum. 2015. Word representations via Gaussian embedding. *ICLR*.

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

POLY: Mining Relational Paraphrases from Multilingual Sentences

Adam Grycner
Max-Planck Institute for Informatics
Saarland Informatics Campus
Building E1.4, 66123
Saarbrücken, Germany
agrycner@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute for Informatics
Saarland Informatics Campus
Building E1.4, 66123
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

Abstract

Language resources that systematically organize paraphrases for binary relations are of great value for various NLP tasks and have recently been advanced in projects like PATTY, WiseNet and DEFIE. This paper presents a new method for building such a resource and the resource itself, called POLY. Starting with a very large collection of multilingual sentences parsed into triples of phrases, our method clusters relational phrases using probabilistic measures. We judiciously leverage fine-grained semantic typing of relational arguments for identifying synonymous phrases. The evaluation of POLY shows significant improvements in precision and recall over the prior works on PATTY and DEFIE. An extrinsic use case demonstrates the benefits of POLY for question answering.

1 Introduction

Motivation. Information extraction from text typically yields relational triples: a binary relation along with its two arguments. Often the relation is expressed by a verb phrase, and the two arguments are named entities. We refer to the surface form of the relation in a triple as a *relational phrase*. Repositories of relational phrases are an asset for a variety of tasks, including information extraction, textual entailment, and question answering.

This paper presents a new method for systematically organizing a large set of such phrases. We aim to construct equivalence classes of synonymous phrases, analogously to how WordNet organizes unary predicates as noun-centric synsets (aka. semantic types). For example, the following relational phrases should be in the same equivalence class: *sings in, is vocalist in, voice in* denoting a relation between a musician and a song.

State of the Art and its Limitations. Starting with the seminal work on DIRT (Lin and Pantel, 2001), there have been various attempts on building comprehensive resources for relational phrases. Recent works include PATTY (Nakashole et al., 2012), WiseNet (Moro and Navigli, 2012) and DEFIE (Bovi et al., 2015). Out of these DEFIE is the cleanest resource. However, the equivalence classes tend to be small, prioritizing precision over recall. On the other hand, PPDB (Ganitkevitch et al., 2013) offers the largest repository of paraphrases. However, the paraphrases are not relation-centric and they are not semantically typed. So it misses out on the opportunity of using types to distinguish identical phrases with different semantics, for example, *performance in* with argument types *musician* and *song* versus *performance in* with types *athlete* and *competition*.

Our Approach. We start with a large collection of relational triples, obtained by shallow information extraction. Specifically, we use the collection of Faruqui and Kumar (2015), obtained by combining the OLLIE tool with Google Translate and projecting multilingual sentences back to English. Note that the task addressed in that work is relational triple extraction, which is orthogonal to our problem of organizing the relational phrases in these triples into synonymy sets.

We canonicalize the subject and object arguments

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2183–2192,
Austin, Texas, November 1-5, 2016.

of triples by applying named entity disambiguation and word sense disambiguation wherever possible. Using a knowledge base of entity types, we can then infer prevalent type signatures for relational phrases. Finally, based on a suite of judiciously devised probabilistic distance measures, we cluster phrases in a type-compatible way using a graph-cut technique. The resulting repository contains ca. 1 Million relational phrases, organized into ca. 160,000 clusters.

Contribution. Our salient contributions are: i) a novel method for constructing a large repository of relational phrases, based on judicious clustering and type filtering; ii) a new linguistic resource, coined POLY, of relational phrases with semantic typing, organized in equivalence classes; iii) an intrinsic evaluation of POLY, demonstrating its high quality in comparison to PATTY and DEFIE; iv) an extrinsic evaluation of POLY, demonstrating its benefits for question answering. The POLY resource is publicly available [1].

2 Method Overview

Our approach consists of two stages: *relational phrase typing* and *relational phrase clustering*. In Section 3, we explain how we infer semantic types of the arguments of a relational phrase. In Section 4, we present the model for computing synonyms of relational phrases (i.e., paraphrases) and organizing them into clusters.

A major asset for our approach is a large corpus of multilingual sentences from the work of Faruqui and Kumar (2015). That dataset contains sentences from Wikipedia articles in many languages. Each sentence has been processed by an Open Information Extraction method (Banko et al., 2007), specifically the OLLIE tool (Mausam et al., 2012), which produces a triple of surface phrases that correspond to a relational phrase candidate and its two arguments (subject and object). Each non-English sentence has been translated into English using Google Translate, thus leveraging the rich statistics that Google has obtained from all kinds of parallel multilingual texts. Altogether, the data from Faruqui and Kumar (2015) provides 135 million triples in 61 languages and in English (from the translations of the corresponding sentences). This is the noisy input to our method. Figure 1 shows two Spanish sentences, the extracted triples of Spanish phrases, the sentences' translations to English, and the extracted triples of English phrases.

The figure shows that identical phrases in the foreign language - "fue filmado por" - may be translated into different English phrases: "was shot by" vs. "was filmed by", depending on the context in the respective sentences. This is the main insight that our method builds on. The two resulting English phrases have a certain likelihood of being paraphrases of the same relation. However, this is an uncertain hypotheses only, given the ambiguity of language, the noise induced by machine translation and the potential errors of the triple extraction. Therefore, our method needs to de-noise these input phrases and quantify to what extent the the relational phrases are indeed synonymous. We discuss this in Sections 3 and 4.

3 Relation Typing

This section explains how we assign semantic types to relational phrases. For example, the relational phrase *wrote* could be typed as *<author> wrote <paper>*, as one candidate. The typing helps us to disambiguate the meaning of the relational phrase and later find correct synonyms. The relational phrase *shot* could have synonyms *directed* or *killed with a gun*. However, they represent different senses of the phrase *shot*. With semantic typing, we can separate these two meanings and determine that *<person> shot <person>* is a synonym of *<person> killed with a gun <person>*, whereas *<director> shot <movie>* is a synonym of *<director> directed <movie>*.

Relation typing has the following steps: argument extraction, argument disambiguation, argument typing and type filtering. The output is a set of candidate types for the left and right arguments of each English relational phrase.

3.1 Argument Extraction

For the typing of a relational phrase, we have to determine words in the left and right arguments that give cues for semantic types. To this end, we identify named entities, whose types can be looked up in a knowledge base, and the head words of common

[1] www.mpi-inf.mpg.de/yago-naga/poly/

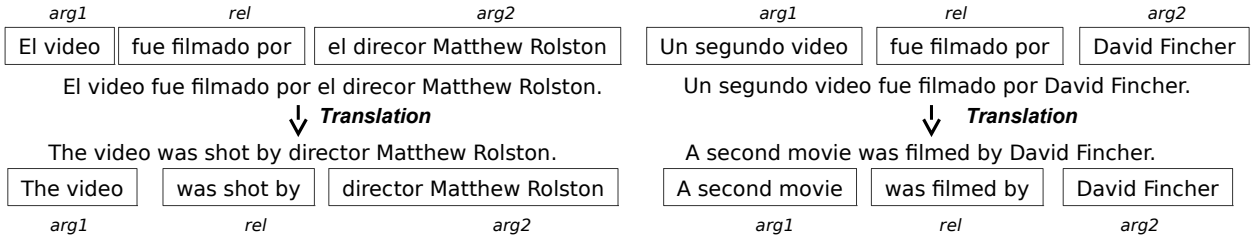


Figure 1: Multilingual input sentences and triples

noun phrases. As output, we produce a ranked list of entity mentions and common nouns.

To create this ranking, we perform POS tagging and noun phrase chunking using Stanford CoreNLP (Manning et al., 2014) and Apache OpenNLP [2]. For head noun extraction, we use the YAGO Javatools[3] and a set of manually crafted regular expressions. Since the input sentences result from machine translation, we could not use dependency parsing, because sentences are often ungrammatical.

Finally, we extract all noun phrases which contain the same head noun. These noun phrases are then sorted according to their lengths.

For example, for input phrase *contemporary British director who also created "Inception"*, our method would yield *contemporary British director*, *British director*, *director* in decreasing order.

3.2 Argument Disambiguation

The second step is responsible for the disambiguation of the noun phrase and named entity candidates. We use the YAGO3 knowledge base (Mahdisoltani et al., 2015) for named entities, and WordNet (Fellbaum, 1998) for noun phrases. We proceed in the ranking order of the phrases from the first step.

Candidate senses are looked up in YAGO3 and WordNet, respectively, and each candidate is scored. The scores are based on:

- Frequency count prior: This is the number of Wikipedia incoming links for named entities in YAGO3, or the frequency count of noun phrase senses in WordNet.
- Wikipedia prior: We increase scores of YAGO3 entities whose URL strings (i.e., Wikipedia titles) occur in the Wikipedia page from which the triple was extracted.
- Translation prior: We boost the scores of senses whose translations occur in the original input sentence. For example, the word *stage* is disambiguated as *opera stage* rather than *phase*, because the original German sentence contains the word *Bühne* (German word for a concert stage) and not *Phase*. The translations of word senses are obtained from Universal WordNet (de Melo and Weikum, 2009).

We prefer WordNet noun phrases over YAGO3 named entities since noun phrases have lower type ambiguity (fewer possible types). The final score of a sense s is:

$$score(\mathrm{s}) = \alpha freq(\mathrm{s}) + \beta wiki(\mathrm{s}) + \gamma trans(\mathrm{s}) \quad (1)$$

where $freq(\mathrm{s})$ is the frequency count of s, and $wiki(\mathrm{s})$ and $trans(\mathrm{s})$ equal maximal frequency count if the Wikipedia prior and Translation prior conditions hold (and otherwise set to 0). α, β, γ are tunable hyper-parameters (set using withheld data).

Finally, from the list of candidates, we generate a disambiguated argument: either a WordNet synset or a YAGO3 entity identifier.

3.3 Argument Typing

In the third step of relation typing, we assign candidate types to the disambiguated arguments. To this end, we query YAGO3 for semantic types (incl. transitive hypernyms) for a given YAGO3 or WordNet identifier.

The type system used in POLY consists of a subset of the WordNet noun hierarchy. We restrict ourselves to 734 types, chosen semi-automatically as follows. We selected the 1000 most frequent WordNet types in YAGO3 (incl. transitive hypernyms). Redundant and non-informative types were filtered out by the following technique: all types were organized into a directed acyclic graph (DAG), and

[2] opennlp.apache.org/

[3] mpi-inf.mpg.de/yago-naga/javatools/

we removed a type when the frequency count of some of its children was higher than 80% of the parent's count. For example, we removed type *trainer* since more than 80% of trainers in YAGO3 are also *coaches*. In addition, we manually removed a few non-informative types (e.g. *expressive style*).

As output, we obtain lists of semantic types for the two arguments of each relational phrase.

3.4 Type Filtering

In the last step, we filter types one more time. This time we filter candidate types separately for each distinct relational phrase, in order to choose the most suitable specific type signature for each phrase. This choice is made by type tree pruning.

For each relational phrase, we aggregate all types of the left arguments and all types of the right arguments, summing up their their frequency counts. This information is organized into a DAG, based on type hypernymy. Then we prune types as follows (similarly to Section 3.3): i) remove a parent type when the relative frequency count of one of the children types is larger than 80% of the parent's count; ii) remove a child type when its relative frequency count is smaller than 20% of the parent's count.

For each of the two arguments of the relational phrase we allow only those types which are left after the pruning. The final output is a set of relational phrases where each has a set of likely type signatures (i.e., pairs of types for the relation's arguments).

4 Relation Clustering

The second stage of POLY addresses the relation clustering. The algorithm takes semantically typed relational phrases as input, quantifies the semantic similarity between relational phrases, and organizes them into clusters of synonyms. The key insight that our approach hinges on is that synonymous phrases have similar translations in a different language. In our setting, two English phrases are semantically similar if they were translated from the same relational phrases in a foreign language and their argument types agree (see Figure 1 for an example). Similarities between English phrases are cast into edge weights of a graph with phrases as nodes. This graph is then partitioned to obtain clusters.

4.1 Probabilistic Similarity Measures

The phrase similarities in POLY are based on probabilistic measures. We use the notation:

- F: a set of relational phrases from a foreign language F
- E: a set of translations of relational phrases from language F to English
- $c(f, e)$: no. of times of translating relational phrase $f \in F$ into relational phrase $e \in E$
- $c(f)$, $c(e)$: frequency counts for relational phrase $f \in F$ and its translation $e \in E$
- $p(e|f) = \frac{c(f,e)}{c(f)}$: (estimator for the) probability of translating $f \in F$ into $e \in E$
- $p(f|e) = \frac{c(f,e)}{c(e)}$: (estimator for the) probability of $e \in E$ being a translation of $f \in F$

We define:

$$p(e_1|e_2) = \sum_f p(e_1|f) * p(f|e_2) \qquad (2)$$

as the probability of generating relational phrase $e_1 \in E$ from phrase $e_2 \in E$. Finally we define:

$$support(e_1, e_2) = \sum_{f \in F} c(f, e_1) * c(f, e_2) \qquad (3)$$

$$confidence(e_1, e_2) = \frac{2}{\frac{1}{p(e_1|e_2)} + \frac{1}{p(e_2|e_1)}} \qquad (4)$$

Confidence is the final similarity measure used in POLY. We use the harmonic mean in Equation 4 to dampen similarity scores that have big differences in their probabilities in Equation 2. Typically, pairs e_1, e_2 with such wide gaps in their probabilities come from subsumptions, not synonymous phrases. Finally, we compute the support and confidence for every pair of English relational phrases which have a common source phrase of translation. We prune phrase pairs with low support (below a threshold), and rank the remaining pairs by confidence.

4.2 Graph Clustering

To compute clusters of relational phrases, we use modularity-based graph partitioning. Specifically, we use the partitioning algorithm of Blondel et al. (2008). The resulting clusters (i.e., subgraphs) are

Cluster of relational phrases
<location> is the heart of <location>
<location> is situated in <location>
<location> is enclosed by <location>
<location> is located amidst <location>
<location> is surrounded by <location>

Table 1: Example of a cluster of relational phrases

then ranked by their weighted graph density multiplied by the graph size (Equation 5). The example of a cluster is shown in Table 1.

$$\frac{\sum_{(e_i,e_j)\in E} sim(e_i,e_j)}{|V| * |V-1|} * |V| \quad (5)$$

5 Evaluation

For the experimental evaluation, we primarily chose triples from the German language (and their English translations). With about 23 million triples, German is the language with the largest number of extractions in the dataset, and there are about 2.5 million distinct relational phrases from the German-to-English translation. The POLY method is implemented using Apache Spark, so it scales out to handle such large inputs.

After applying the relation typing algorithm, we obtain around 10 million typed relational phrases. If we ignored the semantic types, we would have about 950,000 distinct phrases. On this input data, POLY detected 1,401,599 pairs of synonyms. The synonyms were organized into 158,725 clusters.

In the following, we present both an intrinsic evaluation and an extrinsic use case. For the intrinsic evaluation, we asked human annotators to judge whether two typed relational phrases are synonymous or not. We also studied source languages other than German. In addition, we compared POLY against PATTY (Nakashole et al., 2012) and DEFIE (Bovi et al., 2015) on the relation paraphrasing task. For the extrinsic evaluation, we considered a simple question answering system and studied to what extent similarities between typed relational phrases can contribute to answering more questions.

5.1 Precision of Synonyms

To assess the precision of the discovered synonymy among relational phrases (i.e., clusters of paraphrases), we sampled POLY's output. We assessed the 250 pairs of synonyms with the highest similarity scores. We also assessed a sample of 250 pairs of synonyms, randomly drawn from POLY's output.

	Precision	Range
Top 250	0.91	$0.87 - 0.94$
Random	0.83	$0.78 - 0.87$

Table 2: Precision of synonym pairs in POLY

These pairs of synonyms were shown to several human annotators to check their correctness. Relational phrases were presented by showing the semantic types, the textual representation of the relational phrase and sample sentences where the phrase was found. The annotators were asked whether two relational phrases have the same meaning or not. They could also abstain.

The results of this evaluation are shown in Table 2 with (lower bounds and upper bounds of) the 0.95-confidence Wilson score intervals (Brown et al., 2001). This evaluation task had good inter-annotator agreement, with Fleiss' Kappa around 0.6. Table 3 shows anecdotal examples of synonymous pairs of relational phrases.

These results show that POLY's quality is comparable with state-of-the-art baselines resources. WiseNet (Moro and Navigli, 2012) is reported to have precision of 0.85 for 30,000 clusters. This is also the only prior work where the precision of synonymy of semantically typed relational phrases was evaluated. The other systems did not report that measure. However, they performed the evaluation of subsumption, entailment or hypernymy relationships which are related to synonymy. Subsumptions in PATTY have precision of 0.83 for top 100 and 0.75 for a random sample. Hypernyms in RELLY are reported to have precision of 0.87 for top 100 and 0.78 for a random sample. DEFIE performed separate evaluations for hypernyms generated directly from WordNet (precision 0.87) and hypernyms obtained through a substring generalization algorithm (precision 0.9).

Typical errors in the paraphrase discovery of POLY come from incorrect translations or extraction errors. For example, *heard* and *belongs to* were clustered together because they were translated from the

Id	Relation phrase	Synonymous Relational Phrase
1	<location> is surrounded by <region>	<location> is the heart of <region>
2	<artifact> is reminiscent of <time_period>	<artifact> recalls <time_period>
3	<painter> was a participant in <show>	<painter> has participated in <show>
4	<group> maintains a partnership with <district>	<group> has partnered with <district>
5	<movie> was shot at <location>	<movie> was filmed in <location>
6	<person> was shot by <group>	<person> was shot dead by <group>
7	<movie> was shot by <film_director>	<movie> was directed by <film_director>

Table 3: Examples of synonyms of semantically typed relational phrases

same semantically ambiguous German word *gehört*. An example for extraction errors is that *took* and *participated in* were clustered together because *took* was incorrectly extracted from a sentence with the phrase *took part in*. Other errors are caused by swapped order of arguments in a triple (i.e., mistakes in detecting passive form) and incorrect argument disambiguation.

5.2 Comparison to Competitors

To compare POLY with the closest competitors PATTY and DEFIE, we designed an experiment along the lines of the evaluation of Information Retrieval systems (e.g. TREC benchmarks). First, we randomly chose 100 semantically typed relational phrases with at least three words (to focus on the more interesting multi-word case, rather than single verbs). These relational phrases had to occur in all three resources. For every relational phrase we retrieved synonyms from all of the systems, forming a pool of candidates. Next, to remove minor syntactic variations of the same phrase, the relational phrases were lemmatized. In addition, we removed all leading prepositions, modal verbs, and adverbs.

We manually evaluated the correctness of the remaining paraphrase candidates for each of the 100 phrases. Precision was computed as the ratio of the correct synonyms by one system to the number of all synonyms provided by that system. Recall was computed as the ratio of the number of correct synonyms by one system to the number of all correct synonyms in the candidate pool from all three systems.

The results are presented in Table 4. All results are macro-averaged over the 100 sampled phrases. We performed a paired t-test for precision and recall of POLY against each of the systems and obtained p-values below 0.05. POLY and DEFIE offer much higher diversity of synonyms than PATTY. However, DEFIE's synonyms often do not fit the semantic type signature of the given relational phrase and are thus incorrect. For example, *was assumed by* was found to be a synonym of *<group> was acquired by <group>*. PATTY, on the other hand, has higher recall due to its variety of prepositions attached to relational phrases; however, these also include spurious phrases, leading to lower precision. For example, *succeeded in* was found to be a synonym of *<person> was succeeded by <leader>*. Overall, POLY achieves much higher precision and recall than both of these baselines.

	Precision	Recall	F1
PATTY	0.63	0.32	0.42
DEFIE	0.66	0.32	0.44
POLY	**0.79**	**0.46**	**0.58**

Table 4: Comparison to the competitors

5.3 Ablation Study

To evaluate the influence of different components, we performed an ablation study. We consider versions of POLY where Wikipedia prior and Translation prior (Section 3.2) are disregarded (*− disambiguation*), where the type system (Section 3.3) was limited to the 100 most frequent YAGO types (*Type system 100*) or to the 5 top-level types from the YAGO hierarchy (*Type system 5*), or where the type filtering parameter (Section 3.4) was set to 70% or 90% (*Type filtering 0.7/0.9*). The evaluation was done on random samples of 250 pairs of synonyms.

Table 5 shows the results with the 0.95-confidence Wilson score intervals. Without our argument disambiguation techniques, the precision drops heavily. When weakening the type system, our tech-

	Precision	Coverage
POLY	0.83	1,401,599
− disambiguation	0.66 ± 0.06	1,279,941
Type system 100	0.76 ± 0.05	858,053
Type system 5	0.62 ± 0.06	236,804
Type filtering 0.7	0.81 ± 0.05	192,117
Type filtering 0.9	0.73 ± 0.05	2,061,257

Table 5: Ablation Study

	Top 250	Random 250
French	0.93 ± 0.03	0.85 ± 0.04
Hindi	0.86 ± 0.05	0.71 ± 0.05
Russian	0.85 ± 0.05	0.77 ± 0.05

Table 6: Precision of synonyms (other languages)

niques for argument typing and type filtering are penalized, resulting in lower precision. So we see that all components of the POLY architecture are essential for achieving high-quality output. Lowering the type-filtering threshold yields results with comparable precision. However, increasing the threshold results in a worse noise filtering procedure.

5.4 Evaluation with Other Languages

In addition to paraphrases derived from German, we evaluated the relational phrase synonymy derived from a few other languages with lower numbers of extractions. We chose French, Hindi, and Russian (cf. (Faruqui and Kumar, 2015)). The results are presented in Table 6, again with the 0.95-confidence Wilson score intervals.

Synonyms derived from French have similar quality as those from German. This is plausible as one would assume that French and German have similar quality in translation to English. Synonyms derived from Russian and Hindi have lower precision due to the lower translation quality. The precision for Hindi is lower, as the Hindi input corpus has much fewer sentences than for the other languages.

5.5 Extrinsic Evaluation: Question Answering

As an extrinsic use case for the POLY resource, we constructed a simple Question Answering (QA) system over knowledge graphs such as Freebase, and determined the number of questions for which the system can find a correct answer. We followed the approach presented by Fader et al. (2014). The system consists of question parsing, query rewriting and database look-up stages. We disregard the stage of ranking answer candidates, and merely test whether the system could return the right answer (i.e., would return with the perfect ranking).

In the question parsing stage, we use 10 high-precision parsing operators by Fader et al. (2014), which map questions (e.g., *Who invented papyrus?*) to knowledge graph queries (e.g., *(?x, invented, papyrus)*). Additionally, we map question words to semantic types. For example, the word *who* is mapped to *person*, *where* to *location*, *when* to *abstract entity* and the rest of the question words are mapped to type *entity*.

We harness synonyms and hyponyms of relational phrases to paraphrase the predicate of the query. The paraphrases must be compatible with the semantic type of the question word. In the end, we use the original query, as well as found paraphrases, to query a database of subject, predicate, object triples. As the knowledge graph for this experiment we used the union of collections: a triples database from OpenIE (Fader et al., 2011), Freebase (Bollacker et al., 2008), Probase (Wu et al., 2012) and NELL (Carlson et al., 2010). In total, this knowledge graph contained more than 900 Million triples.

We compared six systems for paraphrasing semantically typed relational phrases:

- **Basic**: no paraphrasing at all, merely using the originally generated query.
- **DEFIE**: using the taxonomy of relational phrases by Bovi et al. (2015).
- **PATTY**: using the taxonomy of relational phrases by Nakashole et al. (2012).
- **RELLY**: using the subset of the PATTY taxonomy with additional entailment relationships between phrases (Grycner et al., 2015).
- **POLY_DE**: using synonyms of relational phrases derived from the German language.
- **POLY_ALL**: using synonyms of relational phrases derived from the 61 languages.

Since DEFIE's relational phrases are represented by BabelNet (Navigli and Ponzetto, 2012) word sense identifiers, we generated all possible lemmas for

each identifier.

We ran the paraphrase-enhanced QA system for three benchmark sets of questions:

- **TREC**: the set of questions used for the evaluation of information retrieval QA systems (Voorhees and Tice, 2000)
- **WikiAnswers**: a random subset of questions from WikiAnswers (Fader et al., 2013).
- **WebQuestions**: the set of questions about Freebase entities (Berant et al., 2013).

From these question sets, we kept only those questions which can be parsed by one of the 10 question parsing templates and have a correct answer in the gold-standard ground truth. In total, we executed 451 questions for TREC, 516 for WikiAnswers and 1979 for WebQuestions.

For every question, each paraphrasing system generates a set of answers. We measured for how many questions we could obtain at least one correct answer. Table 7 shows the results.

The best results were obtained by **POLY_ALL**. We performed a paired t-test for the results of POLY_DE and POLY_ALL against all other systems. The differences between POLY_ALL and the other systems are statistically significant with p-value below 0.05.

Additionally, we evaluated paraphrasing systems which consist of combination of all of the described datasets and all of the described datasets without POLY. The difference between these two versions suggest that POLY contains many paraphrases which are available in none of the competing resources.

	TREC	WikiAnswers	WebQuestions
Basic	193	144	365
DEFIE	197	147	394
RELLY	208	150	424
PATTY	213	155	475
POLY_DE	232	163	477
POLY_ALL	**238**	**173**	**530**
All	246	176	562
All / POLY	218	157	494
Questions	451	516	1979

Table 7: Number of questions with correct answer.

6 Related Work

Knowledge bases (KBs) contribute to many NLP tasks, including Word Sense Disambiguation (Moro et al., 2014), Named Entity Disambiguation (Hoffart et al., 2011), Question Answering (Fader et al., 2014), and Textual Entailment (Sha et al., 2015). Widely used KBs are DBpedia (Lehmann et al., 2015), Freebase (Bollacker et al., 2008), YAGO (Mahdisoltani et al., 2015), Wikidata (Vrandecic and Krötzsch, 2014) and the Google Knowledge Vault (Dong et al., 2014). KBs have rich information about named entities, but are pretty sparse on relations. In the latter regard, manually created resources such as WordNet (Fellbaum, 1998), VerbNet (Kipper et al., 2008) or FrameNet (Baker et al., 1998) are much richer, but still face the limitation of labor-intensive input and human curation.

The paradigm of Open Information Extraction (OIE) was developed to overcome the weak coverage of relations in automatically constructed KBs. OIE methods process natural language texts to produce triples of surface forms for the arguments and relational phrase of binary relations. The first large-scale approach along these lines, TextRunner (Banko et al., 2007), was later improved by ReVerb (Fader et al., 2011) and OLLIE (Mausam et al., 2012). The focus of these methods has been on verbal phrases as relations, and there is little effort to determine lexical synonymy among them.

The first notable effort to build up a resource for relational paraphrases is DIRT (Lin and Pantel, 2001), based on Harris' Distributional Hypothesis to cluster syntactic patterns. RESOLVER (Yates and Etzioni, 2009) introduced a probabilistic relational model for predicting synonymy. Yao et al. (2012) incorporated latent topic models to resolve the ambiguity of relational phrases. Other probabilistic approaches employed matrix factorization for finding entailments between relations (Riedel et al., 2013; Petroni et al., 2015) or used probabilistic graphical models to find clusters of relations (Grycner et al., 2014). All of these approaches rely on the co-occurrence of the arguments of the relation.

Recent endeavors to construct large repositories of relational paraphrases are PATTY, WiseNet and DEFIE. PATTY (Nakashole et al., 2012) devised a sequence mining algorithm to extract relational

phrases with semantic type signatures, and organized them into synonymy sets and hypernymy hierarchies. WiseNet (Moro and Navigli, 2012) tapped Wikipedia categories for a similar way of organizing relational paraphrases. DEFIE (Bovi et al., 2015) went even further and used word sense disambiguation, anchored in WordNet, to group phrases with the same meanings.

Translation models have previously been used for paraphrase detection. Barzilay and McKeown (2001) utilized multiple English translations of the same source text for paraphrase extraction. Bannard and Callison-Burch (2005) used the bilingual pivoting method on parallel corpora for the same task. Similar methods were performed at a much bigger scale by the Paraphrase Database (PPDB) project (Pavlick et al., 2015). Unlike POLY, the focus of these projects was not on paraphrases of binary relations. Moreover, POLY considers the semantic type signatures of relations, which is missing in PPDB.

Research on OIE for languages other than English has received little attention. Kim et al. (2011) uses Korean-English parallel corpora for cross-lingual projection. Gamallo et al. (2012) developed an OIE system for Spanish and Portuguese using rules over shallow dependency parsing. The recent work of Faruqui and Kumar (2015) extracted relational phrases from Wikipedia in 61 languages using cross-lingual projection. Lewis and Steedman (2013) clustered semantically equivalent English and French phrases, based on the arguments of relations.

7 Conclusions

We presented POLY, a method for clustering semantically typed English relational phrases using a multilingual corpus, resulting in a repository of semantically typed paraphrases with high coverage and precision. Future work includes jointly processing all 61 languages in the corpus, rather than considering them pairwise, to build a resource for all languages. The POLY resource is publicly available at `www.mpi-inf.mpg.de/yago-naga/poly/`.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *ACL*.

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

Colin J. Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. *TACL*, 3:529–543.

Lawrence D. Brown, T. Tony Cai, and Anirban Dasgupta. 2001. Interval estimation for a binomial proportion. *Statistical Science*, 16:101–133.

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

Gerard de Melo and Gerhard Weikum. 2009. Towards a universal wordnet by learning from combined evidence. In *CIKM*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *NAACL*.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, ROBUS-UNSUP '12, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL*.

Adam Grycner, Gerhard Weikum, Jay Pujara, James Foulds, and Lise Getoor. 2014. A unified probabilistic approach for semantic clustering of relational phrases. In *AKBC '14: Proceedings of the 2014 Workshop on Automated Knowledge Base Construction*.

Adam Grycner, Gerhard Weikum, Jay Pujara, James R. Foulds, and Lise Getoor. 2015. RELLY: Inferring hypernym relationships between relational phrases. In *EMNLP*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2011. A cross-lingual annotation projection-based self-supervision approach for open information extraction. In *IJCNLP*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42:21–40.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Mike Lewis and Mark Steedman. 2013. Unsupervised induction of cross-lingual semantic relations. In *EMNLP*.

Dekang Lin and Patrick Pantel. 2001. DIRT @SBT@discovery of inference rules from text. In *KDD*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP*.

Andrea Moro and Roberto Navigli. 2012. WiseNet: building a wikipedia-based semantic network with ontologized relations. In *CIKM*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP*.

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

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

Fabio Petroni, Luciano Del Corro, and Rainer Gemulla. 2015. CORE: Context-aware open relation extraction with factorization machines. In *EMNLP*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2015. Recognizing textual entailment using probabilistic inference. In *EMNLP*.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *SIGIR*.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *ACL*.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34(1).

Exploiting Sentence Similarities for Better Alignments

Tao Li and **Vivek Srikumar**
University of Utah
{tli,svivek}@cs.utah.edu

Abstract

We study the problem of jointly aligning sentence constituents and predicting their similarities. While extensive sentence similarity data exists, manually generating reference alignments and labeling the similarities of the aligned chunks is comparatively onerous. This prompts the natural question of whether we can exploit easy-to-create sentence level data to train better aligners. In this paper, we present a model that learns to jointly align constituents of two sentences and also predict their similarities. By taking advantage of both sentence and constituent level data, we show that our model achieves state-of-the-art performance at predicting alignments and constituent similarities.

1 Introduction

The problem of discovering semantic relationships between two sentences has given birth to several NLP tasks over the years. Textual entailment (Dagan et al., 2013, *inter alia*) asks about the truth of a hypothesis sentence given another sentence (or more generally a paragraph). Paraphrase identification (Dolan et al., 2004, *inter alia*) asks whether two sentences have the same meaning. Foregoing the binary entailment and paraphrase decisions, the semantic textual similarity (STS) task (Agirre et al., 2012) asks for a numeric measure of semantic equivalence between two sentences. All three tasks have attracted much interest in the form of shared tasks.

While various approaches have been proposed to predict these sentence relationships, a commonly employed strategy (Das and Smith, 2009; Chang et al., 2010a) is to postulate an alignment between constituents of the sentences and use this alignment to make the final prediction (a binary decision or a numeric similarity score). The implicit assumption in such approaches is that better constituent alignments can lead to better identification of semantic relationships between sentences.

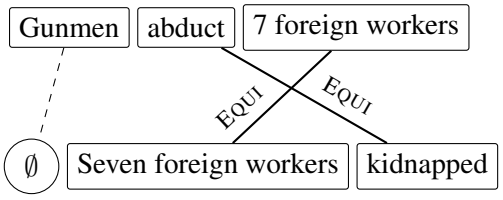


Figure 1: Example constituent alignment. The solid lines represent aligned constituents (here, both labeled equivalent). The chunk *Gunmen* is unaligned.

Constituent alignments serve two purposes. First, they act as an intermediate representation for predicting the final output. Second, the alignments help interpret (and debug) decisions made by the overall system. For example, the alignment between the sentences in Figure 1 can not only be useful to determine the equivalence of the two sentences, but also help reason about the predictions.

The importance of this intermediate representation led to the creation of the *interpretable* semantic textual similarity task (Agirre et al., 2015a) that focuses on predicting chunk-level alignments and similarities. However, while extensive resources exist for sentence-level relationships, human annotated chunk-aligned data is comparatively smaller.

In this paper, we address the following question: can we use sentence-level resources to better pre-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2193–2203,
Austin, Texas, November 1-5, 2016.

dict constituent alignments and similarities? To answer this question, we focus on the semantic textual similarity (STS) task and its interpretable variant. We propose a joint model that aligns constituents and integrates the information across the aligned edges to predict both constituent and sentence level similarity. The key advantage of modeling these two problems jointly is that, during training, the sentence-level information can provide feedback to the constituent-level predictions.

We evaluate our model on the SemEval-2016 task of interpretable STS. We show that even without the sentence information, our joint model that uses constituent alignments and similarities forms a strong baseline. Further, our easily extensible joint model can incorporate sentence-level similarity judgments to produce alignments and chunk similarities that are comparable to the best results in the shared task.

In summary, the contributions of this paper are:

1. We present the first joint model for predicting constituent alignments and similarities. Our model can naturally take advantage of the much larger sentence-level annotations.

2. We evaluate our model on the SemEval-2016 task of interpretable semantic similarity and show state-of-the-art results.

2 Problem Definition

In this section, we will introduce the notation used in the paper using the sentences in Figure 1 as a running example. The input to the problem is a pair of sentences, denoted by $\mathbf{x}$. We will assume that the sentences are chunked (Tjong Kim Sang and Buchholz, 2000) into constituents. We denote the chunks using subscripts. Thus, the input $\mathbf{x}$ consists of two sequences of chunks $\mathbf{s} = (s_1, s_2, \cdots)$ and $\mathbf{t} = (t_1, t_2, \cdots)$ respectively. In our running example, we have $\mathbf{s} =$ (*Gunmen*, *abduct*, *seven foreign workers*) and $\mathbf{t} =$ (*Seven foreign workers*, *kidnapped*).

The output consists of three components:

1. **Alignment**: The alignment between a pair of chunks is a labeled, undirected edge that explains the relation that exists between them. The labels can be one of EQUI (semantically equivalent), OPPO (opposite meaning in context), SPE1, SPE2 (the chunk from $\mathbf{s}$ is more specific than the one from $\mathbf{t}$ and vice versa), SIMI (similar meaning, but none of the previous ones) or REL (related, but none of the above)[1]. In Figure 1, we see two EQUI edges. A chunk from either sentence can be unaligned, as in the case of the chunk *Gunmen*.

   We will use $\mathbf{y}$ to denote the alignment for an input $\mathbf{x}$. The alignment $\mathbf{y}$ consists of a sequence of triples of the form (s_i, t_j, l). Here, s_i and t_j denote a pair of chunks that are aligned with a label l. For brevity, we will include unaligned chunks into this format using a special null chunk and label to indicate that a chunk is unaligned. Thus, the alignment for our running example contain the triple $(\textit{Gunmen}, \emptyset, \emptyset)$.

2. **Chunk similarity**: Every aligned chunk is associated with a relatedness score between zero and five, denoting the range from unrelated to equivalent. Note that even chunks labeled OPPO can be assigned a high score because the polarity is captured by the label rather than the score. We will denote the chunk similarities using $\mathbf{z}$, comprising of numeric $z_{i,j,l}$ for elements of the corresponding alignment $\mathbf{y}$. For an unaligned chunk, the corresponding similarity z is fixed to zero.

3. **Sentence similarity**: The pair of sentences is associated with a scalar score from zero to five, to be interpreted as above. We will use r to denote the sentence similarity for an input $\mathbf{x}$.

Thus, the prediction problem is the following: Given a pair of chunked sentences $\mathbf{x} = (\mathbf{s}, \mathbf{t})$, predict the alignment $\mathbf{y}$, the alignment similarities $\mathbf{z}$ and the sentence similarity r. Note that this problem definition integrates the canonical semantic textual similarity task (only predicting r) and its interpretable variant (predicting both $\mathbf{y}$ and $\mathbf{z}$) into a single task.

[1] We refer the reader to the guidelines of the task (Agirre et al., 2015a) for further details on these labels. Also, for simplicity, in this paper, we ignore the factuality and polarity tags from the interpretable task.

3 Predicting Alignments and Similarities

This section describes our model for predicting alignments, alignment scores, and the sentence similarity scores for a given pair of sentences. We will assume that learning is complete and we have all the scoring functions we need and defer discussing the parameterization and learning to Section 4.

We frame the problem of inference as an instance of an integer linear program (ILP). We will first see the scoring functions and the ILP formulation in Section 3.1. Then, in Section 3.2, we will see how we can directly read off the similarity scores at both chunk and sentence level from the alignment.

3.1 Alignment via Integer Linear Programs

We have two kinds of 0-1 inference variables to represent labeled aligned chunks and unaligned chunks.

We will use the inference variables $\mathbb{1}_{i,j,l}$ to denote the decision that chunks s_i and t_j are aligned with a label l. To allow chunks to be unaligned, the variables $\mathbb{1}_{i,0}$ and $\mathbb{1}_{0,j}$ denote the decisions that s_i and t_j are unaligned respectively.

Every inference decision is scored by the trained model. Thus, we have $\text{score}(i,j,l)$, $\text{score}(i,0)$ and $\text{score}(0,j)$ for the three kinds of inference variables respectively. All scores are of the form $A\left(\mathbf{w}^T\Phi\left(\cdot,\mathbf{s},\mathbf{t}\right)\right)$, where $\mathbf{w}$ is a weight vector that is learned, $\Phi\left(\cdot,\mathbf{s},\mathbf{t}\right)$ is a feature function whose arguments include the constituents and labels in question, and A is a sigmoidal activation function that flattens the scores to the range $[0,5]$. In all our experiments, we used the function $A(x)=\frac{5}{1+e^{-x}}$.

The goal of inference is to find the assignment to the inference variables that maximizes total score. That is, we seek to solve

$$\begin{aligned}\arg\max_{\mathbb{1}\in\mathcal{C}} \quad & \sum_{i,j,l}\text{score}(i,j,l)\mathbb{1}_{i,j,l} \\ & +\sum_{i}\text{score}(i,0)\mathbb{1}_{i,0} \\ & +\sum_{j}\text{score}(0,j)\mathbb{1}_{0,j} \qquad (1)\end{aligned}$$

Here $\mathbb{1}$ represents all the inference variables together and $\mathcal{C}$ denotes the set of all valid assignments to the variables, defined by the following set of constraints:

1. A pair of chunks can have *at most* one label.
2. Either a chunk can be unaligned or it should participate in a labeled alignment with exactly one chunk of the other sentence.

We can convert these constraints into linear inequalities over the inference variables using standard techniques for ILP inference (Roth and Yih, 2004)[2]. Note that, by construction, there is a one-to-one mapping from an assignment to the inference variables $\mathbb{1}$ and the alignment $\mathbf{y}$. In the rest of the paper, we use these two symbols interchangeably, using $\mathbb{1}$ referring details of inference and $\mathbf{y}$ referring to the alignment as a sequence of labeled edges.

3.2 From Alignments to Similarities

To complete the prediction, we need to compute the numeric chunk and sentence similarities given the alignment $\mathbf{y}$. In each case, we make modeling assumptions about how the alignments and similarities are related, as described below.

Chunk similarities To predict the chunk similarities, we assume that the label-specific chunk similarities of aligned chunks *are* the best edge-weights for the corresponding inference variables. That is, for a pair of chunks (s_i,t_j) that are aligned with a label l, the chunk pair similarity $z_{i,j,l}$ is the coefficient associated with the corresponding inference variable. If the alignment edge indicates an unaligned chunk, then the corresponding score is zero. That is,

$$z_{i,j,l}=\begin{cases}A\left(\mathbf{w}^T\Phi\left(s_i,t_j,l,\mathbf{s},\mathbf{t}\right)\right) & \text{if } l\neq\emptyset \\ 0 & \text{if } l=\emptyset.\end{cases} \qquad (2)$$

But can chunk similarities directly be used to find good alignments? To validate this assumption, we performed a pilot experiment on the chunk aligned part of our training dataset. We used the gold standard chunk similarities as scores of the inference variables in the integer program in Eq. 1, with the variables associated with unaligned chunks being scored zero. We found that this experiment gives a near-perfect typed alignment F-score of 0.9875.

[2]While it may be possible to find the score maximizing alignment in the presence of these constraints using dynamic programming (say, a variant of the Kuhn-Munkres algorithm), we model inference as an ILP to allow us the flexibility to explore more sophisticated output interactions in the future.

The slight disparity is because the inference only allows 1-to-1 matches between chunks (constraint 2), which does not hold in a small number of examples.

Sentence similarities Given the aligned chunks $\mathbf{y}$, the similarity between the sentences $\mathbf{s}$ and $\mathbf{t}$ (*i.e.*, in our notation, r) is the weighted average of the chunk similarities (*i.e.*, $z_{i,j,l}$). Formally,

$$r = \frac{1}{|\mathbf{y}|} \sum_{(s_i,t_j,l)\in\mathbf{y}} \alpha_l z_{i,j,l}. \quad (3)$$

Note that the weights α_l depend only on the labels associated with the alignment edge and are designed to capture the polarity and strength of the label. Eq. 3 bridges sentence similarities and chunk similarities. During learning, this provides the feedback from sentence similarities to chunk similarities. The values of the α's can be learned or fixed before learning commences. To simplify our model, we choose the latter approach . Section 5 gives more details.

Features To complete the description of the model, we now describe the features that define the scoring functions. We use standard features from the STS literature (Karumuri et al., 2015; Agirre et al., 2015b; Banjade et al., 2015).

For a pair of chunks, we extract the following similarity features: (1) Absolute cosine similarities of GloVe embeddings (Pennington et al., 2014) of head words, (2) WordNet based Resnik (Resnik, 1995), Leacock (Leacock and Chodorow, 1998) and Lin (Lin, 1998) similarities of head words, (3) Jaccard similarity of content words and lemmas. In addition, we also add indicators for: (1) the part of speech tags of the pair of head words, (2) the pair of head words being present in the lexical large section of the Paraphrase Database (Ganitkevitch et al., 2013), (3) a chunk being longer than the other while both are not named entity chunks, (4) a chunk having more content words than the other, (5) contents of one chunk being a part of the other, (6) having the same named entity type or numeric words, (7) sharing synonyms or antonyms, (8) sharing conjunctions or prepositions, (9) the existence of unigram/bigram/trigram overlap, (10) if only one chunk has a negation, and (11) a chunk having extra content words that are also present in the other sentence.

For a chunk being unaligned, we conjoin an indicator that the chunk is unaligned with the part of speech tag of its head word.

3.3 Discussion

In the model proposed above, by predicting the alignment, we will be able to deterministically calculate both chunk and sentence level similarities. This is in contrast to other approaches for the STS task, which first align constituents and then extract features from alignments to predict similarities in a pipelined fashion. The joint prediction of alignment and similarities allows us to address the primary motivation of the paper, namely using the abundant sentence level data to train the aligner and scorer.

The crucial assumption that drives the joint model is that the *same* set of parameters that can discover a good alignment can also predict similarities. This assumption – similar to the one made by Chang et al. (2010b) – and the associated model described above, imply that the goal of learning is to find parameters that drive the inference towards good alignments and similarities.

4 Learning the Alignment Model

Under the proposed model, the alignment directly predicts the chunk and sentence similarities as well. We utilize two datasets to learn the model:

1. The **alignment dataset** D_A consists of fully annotated aligned chunks and respective chunk similarity scores.

2. The **sentence dataset** D_S that consists of pairs of sentences where each pair is labeled with a numeric similarity score between zero and five.

The goal of learning is to use these two datasets to train the model parameters. Note that unlike standard multi-task learning problems, the two tasks in our case are tightly coupled both in terms of their definition and via the model described in Section 3.

We define three types of loss functions corresponding to the three components of the final output (*i.e.*, alignment, chunk similarity and sentence similarity). Naturally, for each kind of loss, we assume that we have the corresponding ground truth. We will denote ground truth similarity scores and alignments using asterisks. Also, the loss functions

defined below depend on the weight vector $\mathbf{w}$, but this is not shown to simplify notation.

1. The **alignment loss** L_a is a structured loss function that penalizes alignments that are far away from the ground truth. We used the structured hinge loss (Taskar et al., 2004; Tsochantaridis et al., 2005) for this purpose.

$$L_a(\mathbf{s}, \mathbf{t}, \mathbf{y}^*) = \max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{s}, \mathbf{t}, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}^*) - \mathbf{w}^T \Phi(\mathbf{s}, \mathbf{t}, \mathbf{y}^*).$$

   Here, Δ refers to the Hamming distance between the alignments.

2. The **chunk score loss** L_c is designed to penalize errors in predicted chunk level similarities. To account for cases where chunk boundaries may be incorrect, we define this loss as the sum of squared errors of token similarities. However, neither our output nor the gold standard similarities are at the granularity of tokens. Thus, to compute the loss, we project the chunk scores $z_{i,j,l}$ for an aligned chunk pair (s_i, t_j, l) to the tokens that constitute the chunks by equally partitioning the scores among all possible internal alignments. In other words, for a token w_i in the chunk s_i and token w_j in chunk s_j, we define token similarity scores as

$$z(w_i, w_j, l) = \frac{z_{i,j,l}}{N(s_i, t_j)}$$

   Here, the normalizing function N is the product of the number of tokens in the chunks[3]. Note that this definition of the token similarity scores applies to both predicted and gold standard similarities. Unaligned tokens are associated with a zero score.

   We can now define the loss for a token pair $(w_i, w_j) \in (\mathbf{s}, \mathbf{t})$ and a label l as the squared error of their token similarity scores:

$$l(w_i, w_j, l) = \left(z(w_i, w_j, l) - z^*(w_i, w_j, l)\right)^2$$

   The chunk loss score L_c for a sentence pair is the sum of all the losses over all pairs of tokens and labels.

$$L_c(\mathbf{s}, \mathbf{t}, \mathbf{y}, \mathbf{y}^*, \mathbf{z}, \mathbf{z}^*) = \sum_{w_i, w_j, l} l(w_i, w_j, l)$$

3. The **sentence similarity loss** L_s provides feedback to the aligner by penalizing alignments that are far away from the ground truth in their similarity assessments. For a pair of sentences $(\mathbf{s}, \mathbf{t})$, given the ground truth sentence similarity r^* and the predicted sentence similarity r (using Equation (3)), the sentence similarity loss is the squared error:

$$L_s(\mathbf{s}, \mathbf{t}, r^*) = (r - r^*)^2.$$

Our learning objective is the weighted combination of the above three components and a ℓ_2 regularizer on the weight vector. The importance of each type of loss is controlled by a corresponding hyperparameter: λ_a, λ_c and λ_s respectively.

Learning algorithm We have two scenarios to consider: with only alignment dataset D_A, and with both D_A and sentence dataset D_S. Note that even if we train only on the alignment dataset D_A, our learning objective is not convex because the activation function is sigmoidal (in Section 3.1).

In both cases, we use stochastic gradient descent with minibatch updates as the optimizer. In the first scenario, we simply perform the optimization using the alignment and the chunk score losses. We found by preliminary experiments on training data that initializing the weights to one performed best.

Algorithm 1 Learning alignments and similarities, given alignment dataset D_A and sentence similarity dataset D_S. See the text for more details.

1: Initialize all weights to one.
2: $\mathbf{w}^0 \leftarrow SGD(D_A)$: Train an initial model
3: Use $\mathbf{w}^0$ to predict alignments on examples in D_S. Call this $\widehat{D_S}$.
4: $\mathbf{w} \leftarrow SGD(D_A \cup \widehat{D_S})$: Train on both sets of examples.
5: **return** $\mathbf{w}$

When we have both D_A and D_S (Algorithm 1), we first initialize the model on the alignment data

[3] Following the official evaluation of the interpretable STS task, we also experimented with the $\max(|s_i|, |t_j|)$ for the normalizer, but we found via cross validation that the product performs better.

only. Using this initial model, we hypothesize alignments on all examples in D_S to get fully labeled examples. Then, we optimize the full objective (all three loss terms) on the combined dataset. Because our goal is to study the impact on the chunk level predictions, in the full model, the sentence loss does not play a part on examples from D_A.

5 Experiments and Results

The primary research question we seek to answer via experiments is: Can we better predict chunk alignments and similarities by taking advantage of sentence level similarity data?

Datasets We used the training and test data from the 2016 SemEval shared tasks of predicting semantic textual similarity (Agirre et al., 2016a) and interpretable STS (Agirre et al., 2016b), that is, tasks 1 and 2 respectively. For our experiments, we used the headlines and images sections of the data. The data for the interpretable STS task, consisting of manually aligned and scored chunks, provides the alignment datasets for training (D_A). The headlines section of the training data consists for 756 sentence pairs, while the images section consists for 750 sentence pairs. The data for the STS task acts as our sentence level training dataset (D_S). For the headlines section, we used the 2013 headlines test set consisting of 750 sentence pairs with gold sentence similarity scores. For the images section, we used the 2014 images test set consisting of 750 examples. We evaluated our models on the official Task 2 test set, consisting of 375 sentence pairs for both the headlines and images sections. In all experiments, we used gold standard chunk boundaries if they are available (*i.e.*, for D_A).

Pre-processing We pre-processed the sentences with parts of speech using the Stanford CoreNLP toolkit (Manning et al., 2014). Since our setting assumes that we have the chunks as input, we used the Illinois shallow parser (Clarke et al., 2012) to extract chunks from D_S. We post-processed the predicted chunks to correct for errors using the following steps: 1. Split on punctuation; 2. Split on verbs in NP; 3. Split on nouns in VP; 4. Merge PP+NP into PP; 5. Merge VP+PRT into VP if the PRT chunk is not a preposition or a subordinating conjunction; 6. Merge SBAR+NP into SBAR; and 7. Create new contiguous chunks using tokens that are marked as being outside a chunk by the shallow parser. We found that using the above post-processing rules, improved the F1 of chunk accuracy from 0.7865 to 0.8130. We also found via cross-validation that this post-processing improved overall alignment accuracy. The reader may refer to other STS resources (Karumuri et al., 2015) for further improvements along this direction.

Experimental setup We performed stochastic gradient descent for 200 epochs in our experiments, with a mini-batch size of 20. We determined the three λ's using cross-validation, with different hyperparameters for examples from D_A and D_S. Table 1 lists the best hyperparameter values. For performing inference, we used the Gurobi optimizer[4].

Setting	$\lambda_a, \lambda_c, \lambda_s$
headlines, D_A	100, 0.01, N/A
headlines, D_S	0.5, 1, 50
images, D_A	100, 0.01, N/A
images, D_S	5, 2.5, 50

Table 1: Hyperparameters for the various settings, chosen by cross-validation. The alignment dataset do not have a λ associated with the sentence loss.

As noted in Section 3.1, the parameter α_l combines chunk scores into sentence scores. To find these hyper-parameters, we used a set of 426 sentences from the from the headlines training data that had both sentence and chunk annotation. We simplified the search by assuming that α_{Equi} is always 1.0 and all labels other than OPPO have the same α. Using grid search over $[-1, 1]$ in increments of 0.1, we selected α's that gave us the highest Pearson correlation for sentence level similarities. The best α's (with a Pearson correlation of 0.7635) were:

$$\alpha_l = \begin{cases} 1, & l = \text{EQUI}, \\ -1, & l = \text{OPPO}, \\ 0.7, & \text{otherwise} \end{cases}$$

Results Following the official evaluation for the SemEval task, we evaluate both alignments and their

[4] http://www.gurobi.com/

Setting	untyped		typed	
	ali	score	ali	score
Baseline	0.8462	0.7610	0.5462	0.5461
Rank 1	0.8194	0.7865	0.7031	**0.6960**
D_A	**0.9257**	0.8377	**0.7350**	0.6776
$D_A + D_S$	0.9235	**0.8591**	0.7281	0.6948

(a) Headlines results

Setting	untyped		typed	
	ali	score	ali	score
Baseline	0.8556	0.7456	0.4799	0.4799
Rank 1	**0.8922**	**0.8408**	0.6867	0.6708
D_A	0.8689	0.7905	0.6933	0.6411
$D_A + D_S$	0.8738	0.8193	**0.7011**	**0.6769**

(b) Imags results

Table 2: F-score for headlines and images datasets. These tables show the result of our systems, baseline and top-ranked systems. D_A is our strong baseline trained on interpretable STS dataset; $D_A + D_S$ is trained on interpretable STS as well as STS dataset. The rank 1 system on headlines is Inspire (Kazmi and Schüller, 2016) and UWB (Konopik et al., 2016) on images. Bold are the best scores.

corresponding similarity scores. The typed alignment evaluation (denoted by **typed ali** in the results table) measures F1 over the alignment edges where the types need to match, but scores are ignored. The typed similarity evaluation (denoted by **typed score**) is the more stringent evaluation that measures F1 of the alignment edge labels, but penalizes them if the similarity scores do not match. The **untyped** versions of alignment and scored alignment evaluations ignore alignment labels. These metrics, based on Melamed (1997), are tailored for the interpretable STS task[5]. We refer the reader to the guidelines of the task for further details. We report both scores in Table 2. We also list the performance of the baseline system (Sultan et al., 2014a) and the top ranked systems from the 2016 shared task for each dataset[6].

By comparing the rows labeled D_A and $D_A + D_S$ in Table 2 (a) and Table 2 (b), we see that in both the headlines and the images datasets, adding sentence level information improves the untyped score, lifting the stricter typed score F1. On the headlines dataset, incorporating sentence-level information degrades both the untyped and typed alignment quality because we cross-validated on the typed score metric.

The typed score metric is the combination of untyped alignment, untyped score and typed alignment. From the row $D_A + D_S$ in Table 2(a), we observe that the typed score F1 is slightly behind that of rank 1 system while all other three metrics are significantly better, indicating that we need to improve our modeling of the intersection of the three aspects. However, this does not apply to images dataset where the improvement on the typed score F1 comes from the typed alignment.

Further, we see that even our base model that only depends on the alignment data offers strong alignment F1 scores. This validates the utility of jointly modeling alignments and chunk similarities. Adding sentence data to this already strong system leads to performance that is comparable to or better than the state-of-the-art systems. Indeed, our final results would have been ranked first on the images task and a close second on the headlines task in the official standings.

The most significant feedback coming from sentence-level information is with respect to the chunk similarity scores. While we observed slight change in the unscored alignment performance, for both the headlines and the images datasets, we saw improvements in both scored precision and recall when sentence level data was used.

6 Analysis and Discussion

In this section, first, we report the results of manual error analysis. Then, we study the ability of our model to handle data from different domains.

6.1 Error Analysis

To perform a manual error analysis, we selected 40 examples from the development set of the headlines section. We classified the errors made by the full model trained on the alignment and sentence datasets. Below, we report the four most significant types of errors:

1. **Contextual implication**: Chunks that are meant to be aligned are not synonyms by them-

[5]In the SemEval 2016 shared task, the typed score is the metric used for system ranking.

[6]http://alt.qcri.org/semeval2016/task2/

selves but are implied by the context. For instance, *Israeli forces* and *security forces* might be equivalent in certain contexts. Out of the 16 instances of EQUI being misclassified as SPE, eight were caused by the features' inability to ascertain contextual implications. This also accounted for four out of the 15 failures to identify alignments.

2. **Semantic phrase understanding**: These are the cases where our lexical resources failed, *e.g.*, *ablaze* and *left burning*. This accounted for ten of the 15 chunk alignment failures and nine of the 21 labeling errors. Among these, some errors (four alignment failures and four labeling errors) were much simpler than others that could be handled with relatively simple features (e.g. *family reunions* ↔ *family unions*).

3. **Preposition semantics**: The inability to account for preposition semantics accounts for three of the 16 cases where EQUI is mistaken as a SPE. Some examples include *at 91* ↔ *aged 91* and *catch fire* ↔ *after fire*.

4. **Underestimated** EQUI **score**: Ten out of 14 cases of score underestimation happened on EQUI label.

Our analysis suggests that we need better contextual features and phrasal features to make further gains in aligning constituents.

6.2 Does the text domain matter?

In all the experiments in Section 5, we used sentence datasets belonging to the same domain as the alignment dataset (either headlines or images). Given that our model can take advantage of two separate datasets, a natural question to ask is how the domain of the sentence dataset influences overall alignment performance. Additionally, we can also ask how well the trained classifiers perform on out-of-domain data. We performed a series of experiments to explore these two questions. Table 3 summarizes the results of these experiments.

The columns labeled Train and Test of the table show the training and test sets used. Each dataset can be either the headlines section (denoted by hdln), or the images section (img) or not used ($\emptyset$). The last two columns report performance on the test set. The rows 1 and 5 in the table correspond to the in-domain settings and match the results of typed alignment and score in Table 2.

Id	Train		Test	Typed F1	
	D_A	D_S		ali	score
1.	hdln	$\emptyset$	hdln	0.7350	0.6776
2.		img		0.6826	0.6347
3.		$\emptyset$	img	0.6547	0.5989
4.		img		0.6161	0.5854
5.	img	$\emptyset$	img	0.6933	0.6411
6.		hdln		0.7033	0.6793
7.		$\emptyset$	hdln	0.6702	0.6274
8.		hdln		0.6672	0.6445

Table 3: F-score for the domain adaptation experiments. This table shows the performance of training on different dataset combinations.

When the headlines data is tested on the images section, we see that there is the usual domain adaptation problem (row 3 vs row 1) and using target images sentence data does not help (row 4 vs row 3). In contrast, even though there is a domain adaptation problem when we compare the rows 5 and 7, we see that once again, using headlines sentence data improves the predicted scores (row 7 vs row 8). This observation can be explained by the fact that the images sentences are relatively simpler and headlines dataset can provide richer features in comparison, thus allowing for stronger feedback from sentences to constituents.

The next question concerns how the domain of the sentence dataset D_S influences alignment and similarity performance. To answer this, we can compare the results in every pair of rows (*i.e.*, 1 vs 2, 3 vs 4, etc.) We see that when the sentence data from the image data is used in conjunction to the headlines chunk data, it invariably makes the classifiers worse. In contrast, the opposite trend is observed when the headlines sentence data augments the images chunk data. This can once again be explained by relatively simpler sentence constructions in the images set, suggesting that we can leverage linguistically complex corpora to improve alignment on simpler ones. Indeed, surprisingly, we obtain marginally better performance on the images set when we use images chunk level data in conjunction

with the headlines sentence data (row 6 vs the row labeled $D_A + D_S$ in the Table 2(b)).

7 Related Work

Aligning words and phrases between pairs of sentences is widely studied in NLP. Machine translation has a rich research history of using alignments (for *e.g.*, (Koehn et al., 2003; Och and Ney, 2003)), going back to the IBM models (Brown et al., 1993). From the learning perspective, the alignments are often treated as latent variables during learning, as in this work where we treated alignments in the sentence level training examples as latent variables. Our work is also conceptually related to (Ganchev et al., 2008), which asked whether improved alignment error implied better translation.

Outside of machine translation, alignments are employed either explicitly or implicitly for recognizing textual entailment (Brockett, 2007; Chang et al., 2010a) and paraphrase recognition (Das and Smith, 2009; Chang et al., 2010a). Additionally, alignments are explored in multiple ways (tokens, phrases, parse trees and dependency graphs) as a foundation for natural logic inference (Chambers et al., 2007; MacCartney and Manning, 2007; MacCartney et al., 2008). Our proposed aligner can be used to aid such applications.

For predicting sentence similarities, in both variants of the task, word or chunk alignments have extensively been used (Sultan et al., 2015; Sultan et al., 2014a; Sultan et al., 2014b; Hänig et al., 2015; Karumuri et al., 2015; Agirre et al., 2015b; Banjade et al., 2015, and others). In contrast to these systems, we proposed a model that is trained jointly to predict alignments, chunk similarities and sentence similarities. To our knowledge, this is the first approach that combines sentence-level similarity data with fine grained alignments to train a chunk aligner.

8 Conclusion

In this paper, we presented the first joint framework for aligning sentence constituents and predicting constituent and sentence similarities. We showed that our predictive model can be trained using both aligned constituent data *and* sentence similarity data. Our jointly trained model achieves state-of-the-art performance on the task of predicting interpretable sentence similarities.

Acknowledgments

The authors wish to thank the anonymous reviewers and the members of the Utah NLP group for their valuable comments and pointers to references.

References

[Agirre et al.2012] Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics*.

[Agirre et al.2015a] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015a. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Agirre et al.2015b] Eneko Agirre, Aitor Gonzalez-Agirre, Inigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2015b. UBC: Cubes for English Semantic Textual Similarity and Supervised Approaches for Interpretable STS. In *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Agirre et al.2016a] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016a. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation*.

[Agirre et al.2016b] Eneko Agirre, Aitor Gonzalez-Agirre, Inigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2016b. SemEval-2016 Task 2: Interpretable Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation*.

[Banjade et al.2015] Rajendra Banjade, Nobal B Niraula, Nabin Maharjan, Vasile Rus, Dan Stefanescu, Mihai Lintean, and Dipesh Gautam. 2015. NeRoSim: A System for Measuring and Interpreting Semantic Textual Similarity. *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Brockett2007] Chris Brockett. 2007. Aligning the RTE 2006 corpus. Technical Report MSR-TR-2007-77, Microsoft Research.

[Brown et al.1993] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*.

[Chambers et al.2007] Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine De Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning Alignments and Leveraging Natural Logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics.

[Chang et al.2010a] Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010a. Discriminative Learning over Constrained Latent Representations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[Chang et al.2010b] Ming-Wei Chang, Vivek Srikumar, Dan Goldwasser, and Dan Roth. 2010b. Structured Output Learning with Indirect Supervision. In *In Proceedings of the 27th International Conference on Machine Learning*.

[Clarke et al.2012] James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. 2012. An NLP Curator (or: How I Learned to Stop Worrying and Love NLP Pipelines). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

[Dagan et al.2013] Ido Dagan, Dan Roth, Mark Sammons, and Fabio M. Zanzotto. 2013. Recognizing Textual Entailment: Models and Applications. *Synthesis Lectures on Human Language Technologies*.

[Das and Smith2009] Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*.

[Dolan et al.2004] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*.

[Ganchev et al.2008] Kuzman Ganchev, Joao V Graça, and Ben Taskar. 2008. Better Alignments= Better Translations? *Proceedings of ACL-08: HLT*.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[Hänig et al.2015] Christian Hänig, Robert Remus, and Xose De La Puente. 2015. ExB Themis: Extensive Feature Extraction from Word Alignments for Semantic Textual Similarity. *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Karumuri et al.2015] Sakethram Karumuri, Viswanadh Kumar Reddy Vuggumudi, and Sai Charan Raj Chitirala. 2015. UMDuluth-BlueTeam: SVCSTS-A Multilingual and Chunk Level Semantic Similarity System. *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Kazmi and Schüller2016] Mishal Kazmi and Peter Schüller. 2016. Inspire at SemEval-2016 Task 2: Interpretable Semantic Textual Similarity Alignment based on Answer Set Programming. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, June.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

[Konopik et al.2016] Miloslav Konopik, Ondrej Prazak, David Steinberger, and Tomáš Brychcín. 2016. UWB at SemEval-2016 Task 2: Interpretable Semantic Textual Similarity with Distributional Semantics for Chunks. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, June.

[Leacock and Chodorow1998] Claudia Leacock and Martin Chodorow. 1998. Combining Local Context and WordNet Similarity for Word Sense Identification. *WordNet: An Electronic Lexical Database*.

[Lin1998] Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *In Proceedings of the 15th International Conference on Machine Learning*.

[MacCartney and Manning2007] Bill MacCartney and Christopher D Manning. 2007. Natural Logic for Textual Inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

[MacCartney et al.2008] Bill MacCartney, Michel Galley, and Christopher D Manning. 2008. A Phrase-Based Alignment Model for Natural Language Inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

[Manning et al.2014] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

[Melamed1997] Dan Melamed. 1997. Manual Annotation of Translational Equivalence: The Blinker Project.

Technical report, Institute for Research in Cognitive Science, Philadelphia.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics, Volume 29, Number 1, March 2003*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

[Resnik1995] Philip Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*.

[Roth and Yih2004] Dan Roth and Wen-Tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*.

[Sultan et al.2014a] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014a. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association of Computational Linguistics*.

[Sultan et al.2014b] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014b. DLS@CU: Sentence similarity from word alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation*.

[Sultan et al.2015] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Taskar et al.2004] Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems 16*.

[Tjong Kim Sang and Buchholz2000] Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

[Tsochantaridis et al.2005] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research, Volume 6*.

Bi-directional Attention with Agreement for Dependency Parsing

Hao Cheng **Hao Fang**
University of Washington
{chenghao,hfang}@uw.edu

Xiaodong He **Jianfeng Gao** **Li Deng**
Microsoft Research
{xiaohe,jfgao,deng}@microsoft.com

Abstract

We develop a novel bi-directional attention model for dependency parsing, which learns to agree on headword predictions from the forward and backward parsing directions. The parsing procedure for each direction is formulated as sequentially querying the memory component that stores continuous headword embeddings. The proposed parser makes use of *soft* headword embeddings, allowing the model to implicitly capture high-order parsing history without dramatically increasing the computational complexity. We conduct experiments on English, Chinese, and 12 other languages from the CoNLL 2006 shared task, showing that the proposed model achieves state-of-the-art unlabeled attachment scores on 6 languages.[1]

1 Introduction

Recently, several neural network models have been developed for efficiently accessing long-term memory and discovering dependencies in sequential data. The memory network framework has been studied in the context of question answering and language modeling (Weston et al., 2015; Sukhbaatar et al., 2015), whereas the neural attention model under the encoder-decoder framework has been applied to machine translation (Bahdanau et al., 2015) and constituency parsing (Vinyals et al., 2015b). Both frameworks learn the latent alignment between the source and target sequences, and the mechanism of attention over the *encoder* can be viewed as a soft operation on the *memory*. Although already used in the encoder for capturing global context information (Bahdanau et al., 2015), the bi-directional recurrent neural network (RNN) has yet to be employed in the decoder. Bi-directional decoding is expected to be advantageous over the previously developed uni-directional counterpart, because the former exploits richer contextual information. Intuitively, we can use two separate uni-directional RNNs where each one constructs its respective attended encoder context vectors for computing RNN hidden states. However, the drawback of this approach is that the decoder would often produce different alignments resulting in discrepancies for the forward and backward directions. In this paper, we design a training objective function to enforce attention agreement between both directions, inspired by the alignment-by-agreement idea from Liang et al. (2006). Specifically, we develop a dependency parser (BiAtt-DP) using a bi-directional attention model based on the memory network. Given that the golden alignment is observed for dependency parsing in the training stage, we further derive a simple and interpretable approximation for the agreement objective, which makes a natural connection between the latent and observed alignment cases.

The proposed BiAtt-DP parses a sentence in a linear order via sequentially querying the memory component that stores continuous embeddings for all headwords. In other words, we consider all possible arcs during the parsing. This formulation is adopted by graph-based parsers such as the MSTParser (McDonald et al., 2005). The consideration

[1] Our software and models are available at https://github.com/hao-cheng/biattdp.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2204–2214,
Austin, Texas, November 1-5, 2016.

of all possible arcs makes the proposed BiAtt-DP different from many recently developed neural dependency parsers (Chen and Manning, 2014; Weiss et al., 2015; Alberti et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015), which use a transition-based algorithm by modeling the parsing procedure as a sequence of actions on buffers. Moreover, unlike most graph-based parsers which may suffer from high computational complexity when utilizing high-order parsing history (McDonald and Pereira, 2006), the proposed BiAtt-DP can implicitly inject such information into the model while keeping the computational complexity in the order of $\mathcal{O}(n^2)$ for a sentence with n words. This is achieved by feeding the RNN in the query component with a *soft* headword embedding, which is computed as the probability-weighted sum of all headword embeddings in the memory component.

To the best of our knowledge, this is the first attempt to apply memory network models to graph-based dependency parsing. Moreover, it is the first extension of neural attention models from uni-direction to multi-direction by enforcing agreement on alignments. Experiments on English, Chinese, and 12 languages from the CoNLL 2006 shared task show the BiAtt-DP can achieve competitive parsing accuracy with several state-of-the-art parsers. Furthermore, our model achieves the highest unlabeled attachment score (UAS) on Chinese, Czech, Dutch, German, Spanish and Turkish.

2 A MemNet-based Dependency Parser

The proposed parser first encodes each word in a sentence by continuous embeddings using a bi-directional RNN, and then performs two types of operations, *i.e.* 1) headword predictions based on bi-directional parsing history and 2) the relation prediction conditioned on the current modifier and its *predicted* headword both in the embedding space. In the following, we first present how the token embeddings are constructed. Then, the key components of the proposed parser, *i.e.* the memory component and the query component, are discussed in detail. Lastly, we describe the parsing algorithm using a bi-directional attention model with agreement.

2.1 Token Embeddings

In the proposed BiAtt-DP, the memory and query components share the same token embeddings. We use the notion of additive token embedding as in (Botha and Blunsom, 2014) to utilize the available information about the token, e.g., its word form, lemma, part-of-speech (POS) tag, and morphological features. Specifically, the token embedding is computed as

$$\mathbf{E}^{\text{form}}\mathbf{e}_i^{\text{form}} + \mathbf{E}^{\text{pos}}\mathbf{e}_i^{\text{pos}} + \mathbf{E}^{\text{lemma}}\mathbf{e}_i^{\text{lemma}} + \cdots,$$

where $\mathbf{e}_i$'s are one-hot encoding vectors for the i-th word, and $\mathbf{E}$'s are parameters to be learned that store the continuous embeddings for corresponding feature. Note those one-hot encoding vectors have different dimensions, depending on individual vocabulary sizes, and all $\mathbf{E}$'s have the same first dimension but different second dimension. The additive token embeddings allow us to easily integrate a variety of information. Moreover, we only need to make a single decision on the dimensionality of the token embedding, rather than a combination of decisions on word embeddings and POS tag embeddings as in concatenated token embeddings used by Chen and Manning (2014), Dyer et al. (2015) and Weiss et al. (2015). It reduces the number of model parameters to be tuned, especially when lots of different features are used. In our experiments, the word form and fine-grained POS tag are always used, whereas other features are used depending on their availability in the dataset. All singleton words, lemmas, and POS tags are replaced by special tokens.

The additive token embeddings are transformed into another space before they are used by the memory and query components, *i.e.*

$$\mathbf{x}_i = \text{LReL}\left[\mathbf{P}\left(\mathbf{E}^{\text{form}}\mathbf{e}_i^{\text{form}} + \cdots\right)\right],$$

where $\mathbf{P}$ is the projection matrix and is shared by the memory and query components as well. The activation function of this projection layer is the leaky rectified linear (LReL) function (Mass et al., 2013) with 0.1 as the slope of the negative part. In the remaining part of the paper, we refer to $\mathbf{x}_i \in \mathbb{R}^p$ as the token embedding for word at position i. Note the subscript i is substituted by j and t for the memory and query components, respectively.

2.2 Components

As shown in Figure 1, the proposed BiAtt-DP has three components, *i.e.* a memory component, a left-to-right query component, and a right-to-left query component. Given a sentence of length n, the parser first uses a bi-directional RNN to construct $n+1$ headword embeddings, $\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_n \in \mathbb{R}^e$, with $\mathbf{m}_0$ reserved for the `ROOT` symbol. Each query component is an uni-directional attention model. In a query component, a sequence of n modifier embeddings $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ are constructed recursively by conditioning on all headword embeddings. To address the vanishing gradient issue in RNNs, we use the gated recurrent unit (GRU) proposed by Cho et al. (2014), where an update gate and a reset gate are employed to control the information flow. We replace the hyperbolic tangent function in GRU with the LReL function, which is faster to compute and achieves better parsing accuracy in our preliminary studies. In the following, we refer to headword and modifier embeddings as memory and query vectors, respectively.

Memory Component: The proposed BiAtt-DP uses a bi-directional RNN to obtain the memory vectors. At time step j, the current hidden state vector $\mathbf{h}^l_j \in \mathbb{R}^{e/2}$ (or $\mathbf{h}^r_j \in \mathbb{R}^{e/2}$) is computed as a non-linear transformation based on the current input vector $\mathbf{x}_j$ and the previous hidden state vector $\mathbf{h}^l_{j-1}$ (or $\mathbf{h}^r_{j+1}$), *i.e.* $\mathbf{h}^l_j = \mathrm{GRU}(\mathbf{h}^l_{j-1}, \mathbf{x}_j)$ (or $\mathbf{h}^r_j = \mathrm{GRU}(\mathbf{h}^r_{j+1}, \mathbf{x}_j)$). Ideally, the recursive nature of the RNN allows it to capture all context information from one-side, and a bi-directional RNN can thus capture context information from both sides. We concatenate the hidden layers of the left-to-right RNN and the right-to-left RNN for the word at position j as the memory vector $\mathbf{m}_j = \left[\mathbf{h}^l_j; \mathbf{h}^r_j\right]$. These memory vectors are expected to encode the words and their context information in the headword space.

Query Component: For each query component, we use a single-directional RNN with GRU to obtain the query vectors $\mathbf{q}_j$'s, which are the hidden state vectors of the RNN. Each $\mathbf{q}_t$ is used to query the memory component, returning association scores $s_{t,j}$'s between the word at position t and the headword at position j for $j \in \{0, \cdots, n\}$, *i.e.*

$$s_{t,j} = \mathbf{v}^T \phi\left(\mathbf{C}\mathbf{m}_j + \mathbf{D}\mathbf{q}_t\right), \tag{1}$$

where $\phi(\cdot)$ is the element-wise hyperbolic tangent function, and $\mathbf{C} \in \mathbb{R}^{h \times e}$, $\mathbf{D} \in \mathbb{R}^{h \times d}$ and $\mathbf{v} \in \mathbb{R}^h$ are model parameters. Then, we can obtain probabilities (aka attention weights), $a_{t,0}, \cdots, a_{t,n}$, over all headwords in the sentence by normalizing $s_{t,j}$'s, using a softmax function

$$\mathbf{a}_t = \mathrm{softmax}(\mathbf{s}_t). \tag{2}$$

The *soft* headword embedding is then defined as $\tilde{\mathbf{m}}_t = \sum_{j=1}^n a_{t,j}\mathbf{m}_j$. At each time step t, the

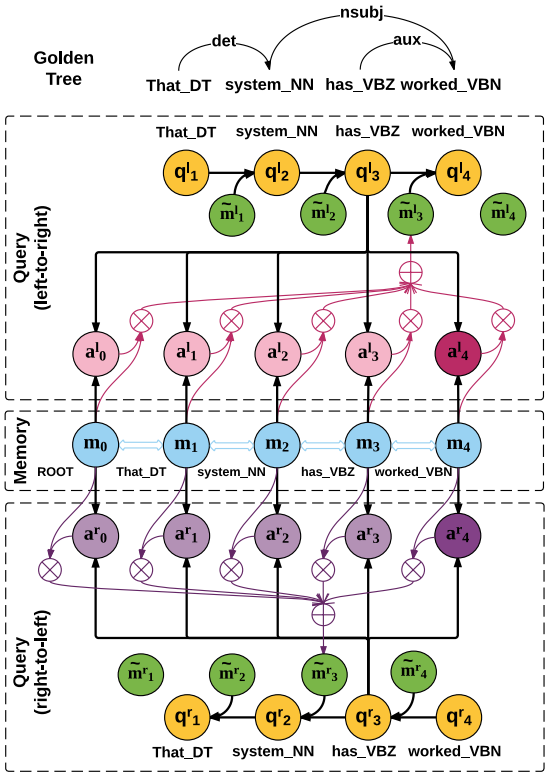


Figure 1: The structure of the BiAtt-DP. The figure only illustrates the parsing process at the time step for `has`. Blue and yellow circles are memory and query vectors, respectively. Red and purple circles represent headword probabilities predicted from corresponding query components. Green circles represent soft headword embeddings. Black arrowed lines are connections carrying weight matrices. $\otimes$ and $\oplus$ indicate element-wise multiplication and addition, respectively. For simplicity, we ignore the token embedding $\mathbf{x}_t$ connected to the RNN hidden layers $\mathbf{m}_j$, $\mathbf{q}^l_t$ and $\mathbf{q}^r_t$.

RNN takes the *soft* headword embedding $\tilde{\mathbf{m}}^l_{t-1}$ or $\tilde{\mathbf{m}}^r_{t+1}$ as the input, in addition to the token embedding $\mathbf{x}_t$. Formally, for the forward case, the $\mathbf{q}_t$ can be computed as $\mathbf{q}_t = \mathrm{GRU}\left(\mathbf{q}_{t-1}, [\tilde{\mathbf{m}}_t; \mathbf{x}_t]\right)$. Although the RNN is able to capture long-span context information to some extent, the local context may very easily dominate the hidden state. Therefore, this additional soft headword embedding allows the model to access long-span context information in a different channel. On the other hand, by recursively feeding both the query vector and the soft headword embedding into the RNN, the model *implicitly* captures high-order parsing history information, which can potentially improve the parsing accuracy (Yamada and Matsumoto, 2003; McDonald and Pereira, 2006). However, for a graph-based dependency parser, utilizing parsing history features is computationally expensive. For example, an k-th order MSTParser (McDonald and Pereira, 2006) has $\mathcal{O}(n^{k+1})$ complexity for a sentence of n words. In contrast, the BiAtt-DP *implicitly* captures high-order parsing history while keeping the complexity in the order of $\mathcal{O}(n^2)$, *i.e.* for each direction. we compute $n(n+1)$ pair-wise probabilities $a_{t,j}$ for $t = 1, \cdots, n$ and $j = 0, \cdots, n$.

In this paper, we choose to use soft headword embeddings rather than making hard decisions on headwords. In the latter case, beam search may potentially improve the parsing accuracy at the cost of higher computational complexity, *i.e.* $\mathcal{O}(Bn^2)$ with a beam width of B. When using soft headword embeddings, there is no need to perform beam search. Moreover, it is straightforward to incorporate parsing history from both directions by using two query components at the cost of $\mathcal{O}(2n^2)$, which cannot be easily achieved when using beam search. The parsing decision can be made directly based on attention weights from the two query components or further rescored by the maximum spanning tree (MST) search algorithm.

2.3 Parsing by Attention with Agreement

For the bi-directional attention model, the underlying probability distributions $\mathbf{a}^l_t$ and $\mathbf{a}^r_t$ may not agree with each other. In order to encourage the agreement, we use the mathematically convenient metric, *i.e.* the squared Hellinger distance $H^2\left(\mathbf{a}^l_t || \mathbf{a}^r_t\right)$, for quantifying the distance between these two distributions. For dependency parsing, when the golden alignment is known during training, we can derive an upper bound on the latent agreement objective as

$$H^2(\mathbf{a}^l_t, \mathbf{a}^r_t) \leq 2\sqrt{D(\mathbf{g}_t||\mathbf{a}^l_t) + D(\mathbf{g}_t||\mathbf{a}^r_t)},$$

where $D(\cdot||\cdot)$ is the KL-divergence. The complete derivation is provided in the Appendix A. During optimization, we can safely drop the constant scaler and the square root operation in the upper bound, leading to the following loss function

$$D(\mathbf{g}_t||\mathbf{a}^l_t) + D(\mathbf{g}_t||\mathbf{a}^r_t) = 2D(\mathbf{g}_t||\mathbf{a}^l_t \odot \mathbf{a}^r_t), \quad (3)$$

where $\odot$ indicates element-wise multiplication. The resulting loss function is equivalent to the cross-entropy loss, which is widely adopted for training neural networks.

As we can see, the loss function (3) tries to minimize the distance between the golden alignment and the intersection of the two directional attention alignments at every time step. Therefore, during inference, the headword prediction for the word at time step t can be obtained as

$$\underset{j}{\mathrm{argmax}} \quad \log a^l_{t,j} + \log a^r_{t,j},$$

seeking for agreement between both query components. This parsing procedure is also similar to the exhaustive left-to-right modifier-first search algorithm described in (Covington, 2001), but it is enhanced by an additional right-to-left search with the agreement enforcement. Alternatively, we can treat $(\log a^l_{t,j} + \log a^r_{t,j})$ as a score of the corresponding arc and then search for the MST to form a dependency parse tree, as proposed in (McDonald et al., 2005). The MST search is achieved via the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which can be implemented in $\mathcal{O}(n^2)$ for dense graphs according to Tarjan (1977). In practice, the MST search slows down the parsing speed by 6–10%. However, it forces the parser to produce a valid tree, and we observe a slight improvement on parsing accuracy in most cases.

After obtaining each modifier and its *soft* header embeddings, we use a single-layer perceptron to predict the head-modifier relation, *i.e.*

$$\mathbf{y}_t = \mathrm{softmax}\left(\mathbf{U}\left[\tilde{\mathbf{m}}^l_t;\, \tilde{\mathbf{m}}^r_t\right] + \mathbf{W}\left[\mathbf{q}^l_t;\, \mathbf{q}^r_t\right]\right), \quad (4)$$

where $y_{t,1}, \cdots, y_{t,m}$ are the probabilities of m possible relations, and $\mathbf{U} \in \mathbb{R}^{m \times 2e}$ and $\mathbf{W} \in \mathbb{R}^{m \times 2d}$ are model parameters.

3 Model Learning

For the t-th word (modifier) w_t in a sentence of length n, let H_t^l and H_t^r denote random variables representing the predicted headword from forward (left-to-right) and backward (right-to-left) parsing directions, respectively. Also let R_t denote the random variable representing the dependency relation for w_t. The joint probability of headword and relation predictions can be written as

$$\begin{aligned} & P(R_{1:n}, H_{1:n}^l, H_{1:n}^r | w_{1:n}) \\ = & \prod_{t=1}^{n} P(R_t | w_{1:n}) P(H_t^l | w_{1:n}) P(H_t^r | w_{1:n}) \\ = & \prod_{t=1}^{n} y_{t,R_t}^l \cdot a_{t,H_t^l}^l \cdot a_{t,H_t^r}^r \end{aligned} \quad (5)$$

where at each time step we assume head-modifier relations and headwords from both directions are independent with each other when conditioned on the global knowledge of the whole sentence. Note that the long-span context and high-order parsing history information are injected when we model $P(H_t^l | w_{1:n})$, $P(H_t^r | w_{1:n})$ and $P(R_t | w_{1:n})$, as discussed in Section 2.2.

As discussed in Section 2.3, the model can be trained by encouraging attention agreement between two query components. From (5), we observe that it is equivalent to maximizing the log-likelihood of the golden dependency tree (or minimizing the cross-entropy) for each training sentence, *i.e.*

$$\sum_{t=1}^{n} \left(\log y_{t,\text{relation}_t} + \log a_{t,\text{head}_t}^l + \log a_{t,\text{head}_t}^r \right),$$

where $a_{t,j}$ and $y_{t,r}$ are defined in (2) and (4), respectively, and $\texttt{relation}_t$ and $\texttt{head}_t$ are golden relation and headword labels, respectively. The gradients are computed via the back-propagation algorithm (Rumelhart et al., 1986). Errors of $\mathbf{y}_t$ come from the arc labels, whereas there are two source of errors for $\mathbf{a}_t$, one from the headword labels and the other back-propagated from errors of $\mathbf{y}_t$. We use stochastic gradient descent with the Adam algorithm proposed in (Kingma and Ba, 2015). The learning rate is halved at each iteration once the log-likelihood of the dev set decreases. The whole training procedure terminates when the log-likelihood decreases for the second time. All learning parameters except bias terms are initialized randomly according to the Gaussian distribution $\mathcal{N}(0, 10^{-2})$. In our experiments, we tune the initial learning rate with a step size of 0.0002, and choose the best one based on the log-likelihood on the dev set at the first epoch. Empirically, the selected initial learning rates fall in the range of $[0.0004, 0.0010]$ for hidden layer size $[128, 320]$, and tend to be larger when using a smaller hidden layer size, *i.e.* $[0.0016, 0.0034]$ for hidden layer size around 80. The training data are randomly shuffled at every epoch.

4 Experiments

In this section, we present the parsing accuracy of the proposed BiAtt-DP on 14 languages. We report both UAS and labeled attachment score (LAS), obtained by the CoNLL-X eval.pl script[2] which ignores punctuation symbols. The headword predictions are made through the MST search, which slightly improves both UAS and LAS (less than 0.3% absolutely). Overall, the proposed BiAtt-DP achieves competitive parsing accuracy on all languages as state-of-the-art parsers, and obtains better UAS in 6 languages. We also show the impact of using POS tags and pre-trained word embeddings. Moreover, different variants of the full model are compared in this section.

4.1 Data

We work on the English Treebank-3 (PTB) dataset (Marcus et al., 1999), the Chinese Treebank-5.1 (CTB) dataset (Palmer et al., 2005), and 12 other languages from the CoNLL 2006 shared task (Buchholz and Marsi, 2006). For PTB and CTB datasets, we use exactly the same setup as in (Chen and Manning, 2014; Dyer et al., 2015). Specifically, we convert the English and Chinese data using the Stanford parser v3.3.0 (de Marneffe et al., 2006) and the Penn2Malt tool (Zhang and Clark, 2008), respectively.

For English, POS tags are obtained using the Stanford POS tagger v3.3.0 (Toutanova et al., 2003),

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

whereas for Chinese, we use gold segmentation and POS tags. When constructing the token embeddings for English and Chinese, both the word form and the POS tag are used. We also initialize $\mathbf{E}^{\text{form}}$ by pre-trained word embeddings[3].

For the 12 other languages, we randomly hold out 5% of the training data as the dev set. In addition to the word form and find-grained POS tags, we use extra features such as lemmas, coarse-grained POS tags, and morphemes when they are available in the dataset. No pre-trained word embeddings are used for these 12 languages.

4.2 Model Configurations

The hidden layer size is kept the same across all RNNs in the proposed BiAtt-DP. We also require the dimension of the token embeddings to be the same as the hidden layer size. Note that we concatenate the hidden layers of two RNNs for constructing $\mathbf{m}_j$, and thus we have $e = 2d$. The weight matrices $\mathbf{C}$ and $\mathbf{D}$ respectively project vectors $\mathbf{m}_j$ and $\mathbf{q}_t$ to the same dimension h, which is equivalent to d. For English and Chinese, since the dimension of pre-trained word embeddings are 300, we use $300 \times h$ as the dimension of embedding parameters $\mathbf{E}$'s. For the 12 other languages, we use square matrices for the embedding parameters $\mathbf{E}$'s. For all languages, We tune the hidden layer size and choose one according to UAS on the dev set. The selected hidden layer sizes for these languages are: 368 (English), 114 (Chinese), 128 (Arabic), 160 (Bulgarian), 224 (Czech), 176 (Danish), 220 (Dutch), 200 (German), 128 (Japanese), 168 (Portuguese), 128 (Slovene), 144 (Spanish), 176 (Swedish), and 128 (Turkish).

4.3 Results

We first compare our parser with state-of-the-art neural transition-based dependency parsers on PTB and CTB. For English, we also compare with state-of-the-art graph-based dependency parsers. The results are shown in Table 1 and Table 2, respectively. It can be seen that the BiAtt-DP outperforms all other graph-based parsers on PTB. Compared with the transition-based parsers, it achieves better accuracy than Chen and Manning (2014), which uses a feed-forward neural network, and Dyer et al. (2015), which uses three stack LSTM networks. Compared with the integrated parsing and tagging models, the BiAtt-DP outperforms Bohnet and Nivre (2012) but has a small gap to Alberti et al. (2015). On CTB, it achieves best UAS and similar LAS. This may be caused by that the relation vocabulary size is relatively smaller than the average sentence length, which biases the joint objective to be more sensitive to UAS. The parsing speed is around 50–60 sents/sec measured on a desktop with Intel Core i7 CPU @ 3.33GHz using single thread.

Next, in Table 3 we show the parsing accuracy of the proposed BiAtt-DP on 12 languages in the CoNLL 2006 shared task, including comparison with state-of-the-art parsers. Specifically, we show UAS of the 3rd-order RBGParser as reported in (Lei et al., 2014) since it also uses low-dimensional continuous embeddings. However, there are several major differences between the RBGParser and the BiAtt-DP. First, in (Lei et al., 2014), the low-dimensional continuous embeddings are derived

Type	Method	UAS	LAS
Trans.	C&M (2014)	91.8_	89.6_
	Dyer et al. (2015)	93.2_	90.9_
	B&N (2012)†	93.33	91.22
	Alberti et al. (2015)†	94.23	92.41
	Weiss et al. (2015)†	94.26	92.41
	Andor et al. (2016)*	**94.41**	**92.55**
Graph	Bohnet (2010)†	92.88	90.71
	Martins et al. (2013)†	92.89	90.55
	Z&M (2014)†	93.22	91.02
	BiAtt-DP	**94.10**	**91.49**

Table 1: Parsing accuracy on PTB test set. Our parser uses the same POS tagger as C&M (2014) and Dyer et al. (2015), whereas other parsers use a different POS tagger. Results with † and $*$ are provided in (Alberti et al., 2015) and (Andor et al., 2016), respectively.

	Dev		Test	
	UAS	LAS	UAS	LAS
C&M (2014)	84.0	82.4	83.9	82.4
Dyer et al. (2015)	87.2	**85.9**	87.2	**85.7**
BiAtt-DP	**87.7**	85.3	**88.1**	**85.7**

Table 2: Parsing accuracy on CTB dev and test sets.

[3]For English, we use the dependency-based word embeddings at `https://goo.gl/tWke3I` (Levy and Goldberg, 2014). For Chinese, we pre-train 192-dimension skip-gram embeddings (Mikolov et al., 2013) on Chinese Gigawords (Graff et al., 2005).

Language	BiAtt-DP		RBGParser	Best Published		Crossed	Uncrossed	%Crossed
Arabic	80.34	[68.58]	79.95	**81.12**	(Ma11)	17.24	80.71	0.58
Bulgarian	93.96	[89.55]	93.50	**94.02**	(Zh14)	79.59	94.10	0.98
Czech	**91.16**	[85.14]	90.50	90.32	(Ma13)	81.62	91.63	4.68
Danish	91.56	[85.53]	91.39	**92.00**	(Zh13)	73.33	91.89	1.80
Dutch	**87.15**	[82.41]	86.41	86.19	(Ma13)	82.82	87.66	10.48
German	**92.71**	[89.80]	91.97	92.41	(Ma13)	85.93	92.90	2.70
Japanese	93.44	[90.67]	93.71	**93.72**	(Ma11)	48.67	94.48	2.26
Portuguese	92.77	[88.44]	91.92	**93.03**	(Ko10)	73.02	93.28	2.52
Slovene	86.01	[75.90]	86.24	**86.95**	(Ma11)	60.11	86.99	3.66
Spanish	**88.74**	[84.03]	88.00	87.98	(Zh14)	50.00	88.77	0.08
Swedish	90.50	[84.05]	91.00	**91.85**	(Zh14)	45.16	90.78	0.62
Turkish	**78.43**	[66.16]	76.84	77.55	(Ko10)	38.85	79.71	3.13

Table 3: UAS on 12 languages in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). We also report corresponding LAS in squared brackets. The results of the 3rd-order RBGParser are reported in (Lei et al., 2014). Best published results on the same dataset in terms of UAS among (Pitler and McDonald, 2015), (Zhang and McDonald, 2014), (Zhang et al., 2013), (Zhang and McDonald, 2012), (Rush and Petrov, 2012), (Martins et al., 2013), (Martins et al., 2010), and (Koo et al., 2010). To study the effectiveness of the parser in dealing with non-projectivity, we follow (Pitler and McDonald, 2015), to compute the recall of crossed and uncrossed arcs in the gold tree, as well as the percentage of crossed arcs.

from low-rank tensors. Second, the RBGParser uses combined scoring of arcs by including traditional features from the MSTParser (McDonald and Pereira, 2006) / TurboParser (Martins et al., 2013). Third, the RBGParser employs a third-order parsing algorithm based on (Zhang et al., 2014), although it also implements a first-order parsing algorithm, which achieves lower UAS in general. In Table 3, we show that the proposed BiAtt-DP outperforms the RBGParser in most languages except Japanese, Slovene, and Swedish.

It can be observed from Table 3 that the BiAtt-DP has highly competitive parsing accuracy as state-of-the-art parsers. Moreover, it achieves best UAS for 5 out of 12 languages. For the remaining seven languages, the UAS gaps between the BiAtt-DP and state-of-the-art parsers are within 1.0%, except Swedish. An arguably fair comparison for the BiAtt-DP is the MSTParser (McDonald and Pereira, 2006), since the BiAtt-DP replaces the scoring function for arcs but uses exactly the same search algorithm. Due to the space limit, we refer readers to (Lei et al., 2014) for results of the MSTParsers (also shown in Appendix B). The BiAtt-DP consistently outperforms both parser by up to 5% absolute UAS score.

Finally, following (Pitler and McDonald, 2015), we also analyze the performance of the BiAtt-DP on both crossed and uncrossed arcs. Since the BiAtt-DP uses a graph-based non-projective parsing algorithm, it is interesting to evaluate the performance on crossed arcs, which result in the non-projectivity of the dependency tree. The last three columns of Table 3 show the recall of crossed arcs, that of uncrossed arcs, and the percentage of crossed arcs in the test set. Pitler and McDonald (2015) reported numbers on the same data for Dutch, German, Portuguese, and Slovene as in this paper. For these four languages, the BiAtt-DP achieves better UAS than that reported in (Pitler and McDonald, 2015). More importantly, we observe that the improvement on recall of crossed arcs (around 10–18% absolutely) is much more significant than that of uncrossed arcs (around 1–3% absolutely), which indicates the effectiveness of the BiAtt-DP in parsing languages with non-projective trees.

4.4 Ablative Study

Here we try to study the impact of using pre-trained word embeddings, POS tags, as well as the bi-directional query components on our model. First of all, we start from our best model (Model 1 in Table 4) on English, which uses 300 as the token embedding dimension and 368 as the hidden layer size. We keep those model parameter dimensions unchanged and analyze different factors by comparing the parsing accuracy on PTB dev set.

No.	INIT	POS	L2R	R2L	UAS	LAS
1	✓	✓	✓	✓	93.99	91.32
2		✓	✓	✓	93.36	90.42
3			✓	✓	91.87	87.85
4		✓		✓	92.64	89.66
5		✓	✓		92.47	89.47
6		✓	✓†	✓†	93.03	90.06

Table 4: Parsing accuracy on PTB dev set for different variants of the full model. `INIT` refers to using pre-trained word embddings to initialize $\mathbf{E}^{\text{form}}$. `POS` refers to using POS tags in token embeddings. `L2R` and `R2L` respectively indicate whether to use the left-to-right and right-to-left query components. † means the query component drops soft headword embeddings when constructing RNN hidden states.

The results are summarized in Table 4. Comparing Models 1–3, it can be observed that without using pre-trained word embeddings, both UAS and LAS drop by 0.6%, and without using POS tags in token embeddings, the numbers further drop by 1.6% in UAS and around 2.6% in LAS. In terms of query components, using single query component (Models 4–5) degrades UAS by 0.7–0.9% and LAS by around 1.0%, compared with Model 2. For Model 6, the soft headword embedding is only used for arc label predictions but not fed into the next hidden state, which is around 0.3% worse than Model 2. This supports the hypothesis about the usefulness of the parsing history information. We also implement a variant of Model 6 which produces one $\mathbf{a}_t$ instead two by using both $\mathbf{q}_t^l$ and $\mathbf{q}_t^r$ in (1). It gets 92.44% UAS and 89.26% LAS, indicating that naively applying a bi-directional RNN may not be enough.

5 Related Work

Neural Dependency Parsing: Recently developed neural dependency parsers are mostly transition-based models, which read words sequentially from a buffer into a stack and incrementally build a parse tree by predicting a sequence of transitions (Yamada and Matsumoto, 2003; Nivre, 2003; Nivre, 2004). A feed-forward neural network is used in (Chen and Manning, 2014), where they represent the current state with 18 selected elements such as the top words on the stack and buffer. Each element is encoded by concatenated embeddings of words, POS tags, and arc labels. Their dependency parser achieves improvement on both accuracy and parsing speed. Weiss et al. (2015) improve the parser using semi-supervised structured learning and unlabeled data. The model is extended to integrate parsing and tagging in (Alberti et al., 2015). On the other hand, Dyer et al. (2015) develop the stack LSTM architecture, which uses three LSTMs to respectively model the sequences of buffer states, stack states, and actions. Unlike the transition-based formulation, the proposed BiAtt-DP directly predicts the headword and the dependency relation at each time step. Specifically, there is no explicit representation of actions or headwords in our model. The model learns to retrieve the most relevant information from the input memory to make decisions on headwords and head-modifier relations.

Graph-based Dependency Parsing: In addition to the transition-based parsers, another line of research in dependency parsing uses graph-based models. Graph-based parser usually build a dependency tree from a directed graph and learns to scoring the possible arcs. Due to this nature, non-projective parsing can be done straightforwardly by most graph-based dependency parsers. The MSTParser (McDonald et al., 2005) and the TurboParser (Martins et al., 2010) are two examples of graph-based parsers. The MSTParser formulates the parsing as searching for the MST, whereas the TurboParser performs approximate variational inference over a factor graph. The RBGParser proposed in (Lei et al., 2014) can also be viewed as a graph-based parser, which scores arcs using low-dimensional continuous features derived from low-rank tensors as well as features used by MSTParser/TurboParser. It also employs a sampler-based algorithm for parsing (Zhang et al., 2014).

Neural Attention Model: The proposed BiAtt-DP is closely related to the memory network (Sukhbaatar et al., 2015) for question answering, as well as the neural attention models for machine translation (Bahdanau et al., 2015) and constituency parsing (Vinyals et al., 2015b). The way we query the memory component and obtain the soft headword embeddings is essentially the attention mechanism. However, different from the above studies where the alignment information is latent, in dependency parsing, the arc between the modifier and

headword is known during training. Thus, we can utilize these labels for attention weights. The similar idea is employed by the pointer network in (Vinyals et al., 2015a), which is used to solve three different combinatorial optimization problems.

6 Conclusion

In this paper, we develop a bi-directional attention model by encouraging agreement between the latent attention alignments. Through a simple and interpretable approximation, we make the connection between latent and observed alignments for training the model. We apply the bi-directional attention model incorporating the agreement objective during training to the proposed memory-network-based dependency parser. The resulting parser is able to implicitly capture the high-order parsing history without suffering from issue of high computational complexity for graph-based dependency parsing.

We have carried out empirical studies over 14 languages. The parsing accuracy of the proposed model is highly competitive with state-of-the-art dependency parsers. For English, the proposed BiAtt-DP outperforms all graph-based parsers. It also achieves state-of-the-art performance in 6 languages in terms of UAS, demonstrating the effectiveness of the proposed mechanism of bi-directional attention with agreement and its use in dependency parsing.

A Upper Bound on $H^2(\mathbf{p}, \mathbf{q})$

Here, we use the following definition of squared Hellinger distance for countable space

$$H^2(\mathbf{p}, \mathbf{q}) = \frac{1}{2} \sum_i (\sqrt{\mathbf{p}_i} - \sqrt{\mathbf{q}_i})^2$$

where $\mathbf{p}, \mathbf{q} \in \Delta^k$ are two k-simplexes. Introducing $\mathbf{g} \in \Delta^k$, the squared Hellinger distance can be upper bounded as

$$\begin{aligned} H^2(\mathbf{p}, \mathbf{q}) &\leq \sqrt{2} H(\mathbf{p}, \mathbf{q}) && (6) \\ &\leq \sqrt{2} \left[H(\mathbf{p}, \mathbf{g}) + H(\mathbf{q}, \mathbf{g}) \right] && (7) \\ &\leq 2\sqrt{H^2(\mathbf{p}, \mathbf{g}) + H^2(\mathbf{q}, \mathbf{g})} && (8) \end{aligned}$$

where (6), (7) and (8) follow the inequalities between the ℓ_1-norm and the ℓ_2-norm, the triangle inequality defined for a metric, and the Cauchy-Schwarz's inequality, respectively. Using the relationship between the KL-divergence and the squared Hellinger distance, (8) can be further bounded by

$$2\sqrt{D(\mathbf{g}||\mathbf{p}) + D(\mathbf{g}||\mathbf{q})}.$$

B UAS Scores of MSTParsers

Language	1st-order		2nd-order	
Arabic	78.30	(2.02)	78.75	(1.57)
Bulgarian	90.98	(3.00)	91.56	(2.42)
Czech	86.18	(4.88)	87.30	(3.76)
Danish	89.84	(1.80)	90.50	(1.14)
Dutch	82.89	(4.54)	84.11	(3.32)
German	89.54	(3.17)	90.14	(2.57)
Japanese	93.38	(0.14)	92.92	(0.60)
Portuguese	89.92	(3.17)	91.08	(2.01)
Slovene	82.09	(4.54)	83.25	(3.38)
Spanish	83.79	(4.59)	84.33	(4.05)
Swedish	88.27	(1.95)	89.05	(1.17)
Turkish	74.81	(3.74)	74.39	(4.16)
Average	85.83	(2.85)	86.45	(2.23)

Table 5: UAS scores of 1st-order and 2-nd order MSTParsers on 12 languages in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). We use the numbers reported in (Lei et al., 2014). Numbers in brackets indicate the absolute improvement of the proposed BiAtt-DP over the MSTParsers.

References

Chris Alberti, David Weiss, Slav Petrov, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 349–359.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1455–1465.

Bernd Bohnet. 2010. Very high accurarcy and fast dependency parsing is not a contradiction. In *Proc. Int. Conf. Computational Linguistics (COLING)*, pages 89–97.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proc. Int. Conf. Machine Learning (ICML)*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. Conf. Computational Natural Language Learning (CoNLL)*, pages 149–164.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 740–750.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahadanau, Fethhi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1724–1734.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescene of a directed graph. *Science Sinica*, 14:1396–1400.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. Annu. ACM Southeast Conf.*, pages 95–102.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. Int. Conf. Language Resources and Evaluation (LREC)*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 334–343.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 718(4):233–240.

David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2005. Chinese Gigaword Second Edition LDC2005T14. Web Download.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1288–1298.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 1381–1391.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 302–308.

Percy Liang, Ben Tasker, and Dan Klein. 2006. Alignment by agreement. In *Proc. Human Language Technology Conf. and Conf. North American Chapter Assoc. for Computational Linguistics (HLT-NAACL)*, pages 104–111.

Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 LDC99T42. Web Download.

Andrè F. T. Martins, Noah A. Smith, and Eric P. Xing. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 34–44.

Andrè F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turing on the turbo: Fast third-order non-projective turbo parsers. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 617–622.

Andrew L. Mass, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. Int. Conf. Machine Learning (ICML)*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. European Chapter Assoc. for Computational Linguistics (EACL)*, pages 81–88.

Ryan McDonald, Fernando Pererira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. Human Language Technology Conf. and Conf. Empirical Methods Natural Language Process. (HLT/EMNLP)*, pages 523–530.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proc. Workshop at Int. Conf. Learning Representations*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. Int. Conf. Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing: Bringing engineering and cognition together. In *Proc. Workshop at ACL*.

Martha Palmer, Fu-Dong Chiou, Nianwen Xue, and Tsan-Kuang Lee. 2005. Chinese Treebank 5.0 LDC2005T01. Web Download.

Emily Pitler and Ryan McDonald. 2015. A linear-time translation system for crossing interval trees. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 662–671.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propogating errors. *Nature*, 323(6088):533–536, October.

Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 498–507.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2431–2439.

Robert E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. Human Language Technology Conf. and Conf. North American Chapter Assoc. for Computational Linguistics (HLT-NAACL)*, pages 173–180.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2692–2700.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2755–2763.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 323–333.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machine. In *Proc. Int. Conf. Parsing Technologies (IWPT)*, pages 195–206.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based depdency parsing using beam-search. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 562–571.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proc. Conf. Empirical Methods Natural Language Process. and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 320–331.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 656–661.

Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 908–913.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Golberson. 2014. Steps to excellence: Simple inference with the refined scoring of dependency trees. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 197–207.

Anchoring and Agreement in Syntactic Annotations

Yevgeni Berzak
CSAIL MIT
berzak@mit.edu

Yan Huang
Language Technology Lab
DTAL Cambridge University
yh358@cam.ac.uk

Andrei Barbu
CSAIL MIT
andrei@0xab.com

Anna Korhonen
Language Technology Lab
DTAL Cambridge University
alk23@cam.ac.uk

Boris Katz
CSAIL MIT
boris@mit.edu

Abstract

We present a study on two key characteristics of human syntactic annotations: anchoring and agreement. Anchoring is a well known cognitive bias in human decision making, where judgments are drawn towards pre-existing values. We study the influence of anchoring on a standard approach to creation of syntactic resources where syntactic annotations are obtained via human editing of tagger and parser output. Our experiments demonstrate a clear anchoring effect and reveal unwanted consequences, including overestimation of parsing performance and lower quality of annotations in comparison with human-based annotations. Using sentences from the Penn Treebank WSJ, we also report systematically obtained inter-annotator agreement estimates for English dependency parsing. Our agreement results control for parser bias, and are consequential in that they are *on par* with state of the art parsing performance for English newswire. We discuss the impact of our findings on strategies for future annotation efforts and parser evaluations.[1]

1 Introduction

Research in NLP relies heavily on the availability of human annotations for various linguistic prediction tasks. Such resources are commonly treated as de facto "gold standards" and are used for both training and evaluation of algorithms for automatic annotation. At the same time, human agreement on these annotations provides an indicator for the difficulty of the task, and can be instrumental for estimating upper limits for the performance obtainable by computational methods.

Linguistic gold standards are often constructed using pre-existing annotations, generated by automatic tools. The output of such tools is then manually corrected by human annotators to produce the gold standard. The justification for this annotation methodology was first introduced in a set of experiments on POS tag annotation conducted as part of the Penn Treebank project (Marcus et al., 1993). In this study, the authors concluded that tagger-based annotations are not only much faster to obtain, but also more consistent and of higher quality compared to annotations from scratch. Following the Penn Treebank, syntactic annotation projects for various languages, including German (Brants et al., 2002), French (Abeillé et al., 2003), Arabic (Maamouri et al., 2004) and many others, were annotated using automatic tools as a starting point. Despite the widespread use of this annotation pipeline, there is, to our knowledge, little prior work on syntactic annotation quality and on the reliability of system evaluations on such data.

In this work, we present a systematic study of the influence of automatic tool output on characteristics of annotations created for NLP purposes. Our investigation is motivated by the hypothesis that annotations obtained using such methodologies may be

[1] The experimental data in this study will be made publicly available.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2215–2224,
Austin, Texas, November 1-5, 2016.

subject to the problem of *anchoring*, a well established and robust cognitive bias in which human decisions are affected by pre-existing values (Tversky and Kahneman, 1974). In the presence of anchors, participants reason relative to the existing values, and as a result may provide different solutions from those they would have reported otherwise. Most commonly, anchoring is manifested as an alignment *towards* the given values.

Focusing on the key NLP tasks of POS tagging and dependency parsing, we demonstrate that the standard approach of obtaining annotations via human correction of automatically generated POS tags and dependencies exhibits a clear anchoring effect – a phenomenon we refer to as *parser bias*. Given this evidence, we examine two potential adverse implications of this effect on parser-based gold standards.

First, we show that parser bias entails substantial overestimation of parser performance. In particular, we demonstrate that bias towards the output of a specific tagger-parser pair leads to over-estimation of the performance of these tools relative to other tools. Moreover, we observe general performance gains for automatic tools relative to their performance on human-based gold standards. Second, we study whether parser bias affects the quality of the resulting gold standards. Extending the experimental setup of Marcus et al. (1993), we demonstrate that parser bias may lead to *lower* annotation quality for parser-based annotations compared to human-based annotations.

Furthermore, we conduct an experiment on inter-annotator agreement for POS tagging and dependency parsing which controls for parser bias. Our experiment on a subset of section 23 of the WSJ Penn Treebank yields agreement rates of 95.65 for POS tagging and 94.17 for dependency parsing. This result is significant in light of the state of the art tagging and parsing performance for English newswire. With parsing reaching the level of human agreement, and tagging surpassing it, a more thorough examination of evaluation resources and evaluation methodologies for these tasks is called for.

To summarize, we present the first study to measure and analyze anchoring in the standard parser-based approach to creation of gold standards for POS tagging and dependency parsing in NLP. We conclude that gold standard annotations that are based on editing output of automatic tools can lead to inaccurate figures in system evaluations and lower annotation quality. Our human agreement experiment, which controls for parser bias, yields agreement rates that are comparable to state of the art automatic tagging and dependency parsing performance, highlighting the need for a more extensive investigation of tagger and parser evaluation in NLP.

2 Experimental Setup

2.1 Annotation Tasks

We examine two standard annotation tasks in NLP, POS tagging and dependency parsing. In the POS tagging task, each word in a sentence has to be categorized with a Penn Treebank POS tag (Santorini, 1990) (henceforth POS). The dependency parsing task consists of providing a sentence with a labeled dependency tree using the Universal Dependencies (UD) formalism (De Marneffe et al., 2014), according to version 1 of the UD English guidelines[2]. To perform this task, the annotator is required to specify the head word index (henceforth HIND) and relation label (henceforth REL) of each word in the sentence.

We distinguish between three variants of these tasks, *annotation*, *reviewing* and *ranking*. In the annotation variant, participants are asked to conduct annotation from scratch. In the reviewing variant, they are asked to provide alternative annotations for all annotation tokens with which they disagree. The participants are not informed about the source of the given annotation, which, depending on the experimental condition can be either parser output or human annotation. In the ranking task, the participants rank several annotation options with respect to their quality. Similarly to the review task, the participants are not given the sources of the different annotation options. Participants performing the annotation, reviewing and ranking tasks are referred to as annotators, reviewers and judges, respectively.

2.2 Annotation Format

All annotation tasks are performed using a CoNLL style text-based template, in which each word appears in a separate line. The first two columns of each line contain the word index and the word, re-

[2] http://universaldependencies.org/#en

spectively. The next three columns are designated for annotation of POS, HIND and REL.

In the annotation task, these values have to be specified by the annotator from scratch. In the review task, participants are required to edit pre-annotated values for a given sentence. The sixth column in the review template contains an additional # sign, whose goal is to prevent reviewers from overlooking and passively approving existing annotations. Corrections are specified following this sign in a space separated format, where each of the existing three annotation tokens is either corrected with an alternative annotation value or approved using a * sign. Approval of all three annotation tokens is marked by removing the # sign. The example below presents a fragment from a sentence used for the reviewing task, in which the reviewer approves the annotations of all the words, with the exception of "help", where the POS is corrected from VB to NN and the relation label *xcomp* is replaced with *dobj*.

```
...
5   you         PRP   6    nsubj
6   need        VBP   3    ccomp
7   help        VB    6    xcomp  # NN * dobj
...
```

The format of the ranking task is exemplified below. The annotation options are presented to the participants in a random order. Participants specify the rank of each annotation token following the vertical bar. In this sentence, the label *cop* is preferred over *aux* for the word "be" and *xcomp* is preferred over *advcl* for the word "Common".

```
...
8   it          PRP   10   nsubjpass
9   is          VBZ   10   auxpass
10  planed      VBN   0    root
11  to          TO    15   mark
12  be          VB    15   aux-cop | 2-1
13  in          IN    15   case
14  Wimbledon   NNP   15   compound
15  Common      NNP   10   advcl-xcomp | 2-1
...
```

The participants used basic validation scripts which checked for typos and proper formatting of the annotations, reviews and rankings.

2.3 Evaluation Metrics

We measure both parsing performance and inter-annotator agreement using tagging and parsing evaluation metrics. This choice allows for a direct comparison between parsing and agreement results. In this context, POS refers to tagging accuracy. We utilize the standard metrics Unlabeled Attachment Score (UAS) and Label Accuracy (LA) to measure accuracy of head attachment and dependency labels. We also utilize the standard parsing metric Labeled Attachment Score (LAS), which takes into account both dependency arcs and dependency labels. In all our parsing and agreement experiments, we exclude punctuation tokens from the evaluation.

2.4 Corpora

We use sentences from two publicly available datasets, covering two different genres. The first corpus, used in the experiments in sections 3 and 4, is the First Certificate in English (FCE) Cambridge Learner Corpus (Yannakoudakis et al., 2011). This dataset contains essays authored by upper-intermediate level English learners[3].

The second corpus is the WSJ part of the Penn Treebank (WSJ PTB) (Marcus et al., 1993). Since its release, this dataset has been the most commonly used resource for training and evaluation of English parsers. Our experiment on inter-annotator agreement in section 5 uses a random subset of the sentences in section 23 of the WSJ PTB, which is traditionally reserved for tagging and parsing evaluation.

2.5 Annotators

We recruited five students at MIT as annotators. Three of the students are linguistics majors and two are engineering majors with linguistics minors. Prior to participating in this study, the annotators completed two months of training. During training, the students attended tutorials, and learned the annotation guidelines for PTB POS tags, UD guidelines, as well as guidelines for annotating challenging syntactic structures arising from grammatical errors. The students also annotated individually six

[3]The annotation bias and quality results reported in sections 3 and 4 use the original learner sentences, which contain grammatical errors. These results were replicated on the error corrected versions of the sentences.

practice batches of 20-30 sentences from the English Web Treebank (EWT) (Silveira et al., 2014) and FCE corpora, and resolved annotation disagreements during group meetings.

Following the training period, the students annotated a treebank of learner English (Berzak et al., 2016) over a period of five months, three of which as a full time job. During this time, the students continued attending weekly meetings in which further annotation challenges were discussed and resolved. The annotation was carried out for sentences from the FCE dataset, where both the original and error corrected versions of each sentence were annotated and reviewed. In the course of the annotation project, each annotator completed approximately 800 sentence annotations, and a similar number of sentence reviews. The annotations and reviews were done in the same format used in this study. With respect to our experiments, the extensive experience of our participants and their prior work as a group strengthen our results, as these characteristics reduce the effect of anchoring biases and increase inter-annotator agreement.

3 Parser Bias

Our first experiment is designed to test whether expert human annotators are biased towards POS tags and dependencies generated by automatic tools. We examine the common out-of-domain annotation scenario, where automatic tools are often trained on an existing treebank in one domain, and used to generate initial annotations to speed-up the creation of a gold standard for a new domain. We use the EWT UD corpus as the existing gold standard, and a sample of the FCE dataset as the new corpus.

Procedure

Our experimental procedure, illustrated in figure 1(a) contains a set of 360 sentences (6,979 tokens) from the FCE, for which we generate three gold standards: one based on human annotations and two based on parser outputs. To this end, for each sentence, we assign *at random* four of the participants to the following annotation and review tasks. The fifth participant is left out to perform the quality ranking task described in section 4.

The first participant annotates the sentence from scratch, and a second participant reviews this annotation. The overall agreement of the reviewers with the annotators is 98.24 POS, 97.16 UAS, 96.3 LA and 94.81 LAS. The next two participants review parser outputs. One participant reviews an annotation generated by the Turbo tagger and parser (Martins et al., 2013). The other participant reviews the output of the Stanford tagger (Toutanova et al., 2003) and RBG parser (Lei et al., 2014). The taggers and parsers were trained on the gold annotations of the EWT UD treebank, version 1.1. Both parsers use predicted POS tags for the FCE sentences.

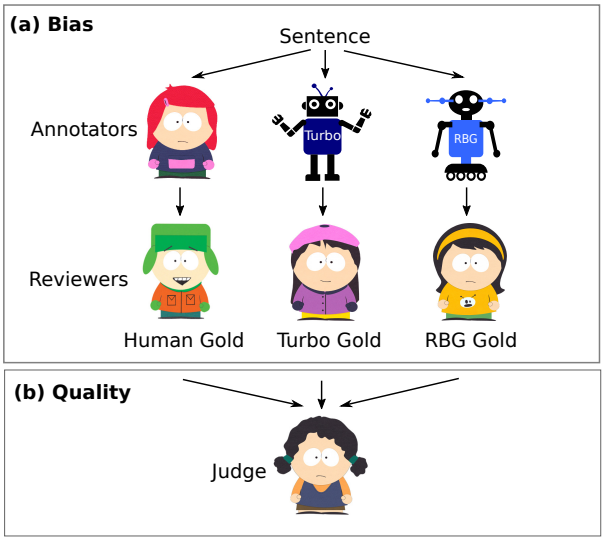


Figure 1: Experimental setup for parser bias (a) and annotation quality (b) on 360 sentences (6,979 tokens) from the FCE. For each sentence, five human annotators are assigned at random to one of three roles: annotation, review or quality assessment. In the bias experiment, presented in section 3, every sentence is annotated by a human, Turbo parser (based on Turbo tagger output) and RBG parser (based on Stanford tagger output). Each annotation is reviewed by a different human participant to produce three gold standards of each sentence: "Human Gold", "Turbo Gold" and "RBG Gold". The fifth annotator performs a quality assessment task described in section 4, which requires to rank the three gold standards in cases of disagreement.

Assigning the reviews to the human annotations yields a human based gold standard for each sentence called "Human Gold". Assigning the reviews to the tagger and parser outputs yields two parser-based gold standards, "Turbo Gold" and "RBG Gold". We chose the Turbo-Turbo and Stanford-RBG tagger-parser pairs as these tools obtain comparable performance on standard evaluation bench-

	Turbo				RBG			
	POS	UAS	LA	LAS	POS	UAS	LA	LAS
Human Gold	95.32	87.29	88.35	82.29	95.59	87.19	88.03	82.05
Turbo Gold	**97.62**	**91.86**	**92.54**	**89.16**	*96.64*	*89.16*	*89.75*	*84.86*
Error Reduction %	49.15	35.96	35.97	38.79	23.81	15.38	14.37	15.65
RBG Gold	96.43	*88.65*	*89.95*	*84.42*	**97.76**	**91.22**	**91.84**	**87.87**
Error Reduction %	23.72	10.7	13.73	12.03	49.21	31.46	31.83	32.42

Table 1: Annotator bias towards taggers and parsers on 360 sentences (6,979 tokens) from the FCE. Tagging and parsing results are reported for the Turbo parser (based on the output of the turbo Tagger) and RBG parser (based on the output of the Stanford tagger) on three gold standards. Human Gold are manual corrections of human annotations. Turbo Gold are manual corrections of the output of Turbo tagger and Turbo parser. RBG Gold are manual corrections of the Stanford tagger and RBG parser. Error reduction rates are reported relative to the results obtained by the two tagger-parser pairs on the Human Gold annotations. Note that (1) The parsers perform equally well on Human Gold. (2) Each parser performs better than the other parser on its own reviews. (3) Each parser performs better on the reviews of the other parser compared to its performance on Human Gold. The differences in (2) and (3) are statistically significant with $p \ll 0.001$ using McNemar's test.

marks, while yielding substantially different annotations due to different training algorithms and feature sets. For our sentences, the agreement between the Turbo tagger and Stanford tagger is 96.97 POS. The agreement between the Turbo parser and RBG parser based on the respective tagger outputs is 90.76 UAS, 91.6 LA and 87.34 LAS.

Parser Specific and Parser Shared Bias

In order to test for parser bias, in table 1 we compare the performance of the Turbo-Turbo and Stanford-RBG tagger-parser pairs on our three gold standards. First, we observe that while these tools perform equally well on Human Gold, each tagger-parser pair performs better than the other on its own reviews. These *parser specific* performance gaps are substantial, with an average of 1.15 POS, 2.63 UAS, 2.34 LA and 3.88 LAS between the two conditions. This result suggests the presence of a bias towards the output of specific tagger-parser combinations. The practical implication of this outcome is that a gold standard created by editing an output of a parser is likely to boost the performance of that parser in evaluations and over-estimate its performance relative to other parsers.

Second, we note that the performance of each of the parsers on the gold standard of the other parser is still higher than its performance on the human gold standard. The average performance gap between these conditions is 1.08 POS, 1.66 UAS, 1.66 LA and 2.47 LAS. This difference suggests an annotation bias towards *shared* aspects in the predictions of taggers and parsers, which differ from the human based annotations. The consequence of this observation is that irrespective of the specific tool that was used to pre-annotate the data, parser-based gold standards are likely to result in higher parsing performance relative to human-based gold standards.

Taken together, the parser specific and parser shared effects lead to a dramatic overall average error reduction of 49.18% POS, 33.71% UAS, 34.9% LA and 35.61% LAS on the parser-based gold standards compared to the human-based gold standard. To the best of our knowledge, these results are the first systematic demonstration of the tendency of the common approach of parser-based creation of gold standards to yield biased annotations and lead to overestimation of tagging and parsing performance.

4 Annotation Quality

In this section we extend our investigation to examine the impact of parser bias on the quality of parser-based gold standards. To this end, we perform a manual comparison between human-based and parser-based gold standards.

Our quality assessment experiment, depicted schematically in figure 1(b), is a ranking task. For each sentence, a randomly chosen judge, who did not annotate or review the given sentence, ranks disagreements between the three gold standards Human Gold, Turbo Gold and RBG Gold, generated in the parser bias experiment in section 3.

Table 2 presents the preference rates of judges

Human Gold Preference %	POS	HIND	REL
Turbo Gold	64.32*	63.96*	61.5*
# disagreements	199	444	439
RBG Gold	56.72	61.38*	57.73*
# disagreements	201	435	440

Table 2: Human preference rates for a human-based gold standard Human Gold over the two parser-based gold standards Turbo Gold and RBG Gold. # disagreements denotes the number of tokens that differ between Human Gold and the respective parser-based gold standard. Statistically significant values for a two-tailed Z test with $p < 0.01$ are marked with *. Note that for both tagger-parser pairs, human judges tend to prefer human-based over parser-based annotations.

for the human-based gold standard over each of the two parser-based gold standards. In all three evaluation categories, human judges tend to prefer the human-based gold standard over both parser-based gold standards. This result demonstrates that the initial reduced quality of the parser outputs compared to human annotations indeed percolates via anchoring to the resulting gold standards.

The analysis of the quality assessment experiment thus far did not distinguish between cases where the two parsers agree and where they disagree. In order to gain further insight into the relation between parser bias and annotation quality, we break down the results reported in table 2 into two cases which relate directly to the *parser specific* and *parser shared* components of the tagging and parsing performance gaps observed in the parser bias results reported in section 3. In the first case, called "parser specific approval", a reviewer approves a parser annotation which disagrees both with the output of the other parser and the Human Gold annotation. In the second case, called "parser shared approval", a reviewer approves a parser output which is shared by both parsers but differs with respect to Human Gold.

Table 3 presents the judge preference rates for the Human-Gold annotations in these two scenarios. We observe that cases in which the parsers disagree are of substantially worse quality compared to human-based annotations. However, in cases of agreement between the parsers, the resulting gold standards do not exhibit a clear disadvantage relative to the Human Gold annotations.

This result highlights the crucial role of parser

Human Gold Preference %	POS	HIND	REL
Turbo specific approval	85.42*	78.69*	80.73*
# disagreements	48	122	109
RBG specific approval	73.81*	77.98*	77.78*
# disagreements	42	109	108
Parser shared approval	51.85	58.49*	51.57
# disagreements	243	424	415

Table 3: Breakdown of the Human preference rates for the human-based gold standard over the parser-based gold standards in table 2, into cases of agreement and disagreement between the two parsers. Parser specific approval are cases in which a parser output approved by the reviewer differs from both the output of the other parser and the Human Gold annotation. Parser shared approval denotes cases where an approved parser output is identical to the output of the other parser but differs from the Human Gold annotation. Statistically significant values for a two-tailed Z test with $p < 0.01$ are marked with *. Note that parser specific approval is substantially more detrimental to the resulting annotation quality compared to parser shared approval.

specific approval in the overall preference of judges towards human-based annotations in table 2. Furthermore, it suggests that annotations on which multiple state of the art parsers agree are of sufficiently high accuracy to be used to save annotation time without substantial impact on the quality of the resulting resource. In section 7 we propose an annotation scheme which leverages this insight.

5 Inter-annotator Agreement

Agreement estimates in NLP are often obtained in annotation setups where both annotators edit the same automatically generated input. However, in such experimental conditions, anchoring can introduce cases of spurious disagreement as well as spurious agreement between annotators due to alignment of one or both participants towards the given input. The initial quality of the provided annotations in combination with the parser bias effect observed in section 3 may influence the resulting agreement estimates. For example, in Marcus et al. (1993) annotators were shown to produce POS tagging agreement of 92.8 on annotation from scratch, compared to 96.5 on reviews of tagger output.

Our goal in this section is to obtain estimates for inter-annotator agreement on POS tagging and dependency parsing that control for parser bias, and

as a result, reflect more accurately human agreement on these tasks. We thus introduce a novel pipeline based on human annotation only, which eliminates parser bias from the agreement measurements. Our experiment extends the human-based annotation study of Marcus et al. (1993) to include also syntactic trees. Importantly, we include an additional review step for the initial annotations, designed to increase the precision of the agreement measurements by reducing the number of errors in the original annotations.

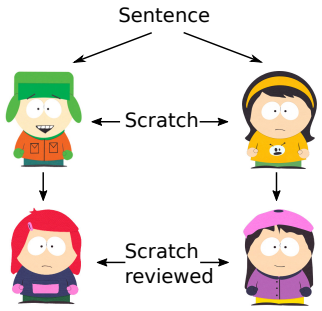


Figure 2: Experimental setup for the inter-annotator agreement experiment. 300 sentences (7,227 tokens) from section 23 of the PTB-WSJ are annotated and reviewed by four participants. The participants are assigned to the following tasks *at random* for each sentence. Two participants annotate the sentence from scratch, and the remaining two participants review one of these annotations each. Agreement is measured on the annotations ("scratch") as well after assigning the review edits ("scratch reviewed").

For this experiment, we use 300 sentences (7,227 tokens) from section 23 of the PTB-WSJ, the standard test set for English parsing in NLP. The experimental setup, depicted graphically in figure 2, includes four participants randomly assigned for each sentence to annotation and review tasks. Two of the participants provide the sentence with annotations from scratch, while the remaining two participants provide reviews. Each reviewer edits one of the annotations independently, allowing for correction of annotation errors while maintaining the independence of the annotation sources. We measure agreement between the initial annotations ("scratch"), as well as the agreement between the reviewed versions of our sentences ("scratch reviewed").

The agreement results for the annotations and the reviews are presented in table 4. The initial agreement rate on POS annotation from scratch is higher than in (Marcus et al., 1993). This difference is likely to arise, at least in part, due to the fact that their experiment was conducted at the beginning of the annotation project, when the annotators had a more limited annotation experience compared to our participants. Overall, we note that the agreement rates from scratch are relatively low. The review round raises the agreement on all the evaluation categories due to elimination of annotation errors present the original annotations.

	POS	UAS	LA	LAS
scratch	94.78	93.07	92.3	88.32
scratch reviewed	**95.65**	**94.17**	**94.04**	**90.33**

Table 4: Inter-annotator agreement on 300 sentences (7,227 tokens) from the PTB-WSJ section 23. "scratch" is agreement on independent annotations from scratch. "scratch reviewed" is agreement on the same sentences after an additional independent review round of the annotations.

Our post-review agreement results are consequential in light of the current state of the art performance on tagging and parsing in NLP. For more than a decade, POS taggers have been achieving over 97% accuracy with the PTB POS tag set on the PTB-WSJ test set. For example, the best model of the Stanford tagger reported in Toutanova et al. (2003) produces an accuracy of 97.24 POS on sections 22-24 of the PTB-WSJ. These accuracies are above the human agreement in our experiment.

With respect to dependency parsing, recent parsers obtain results which are on par or higher than our inter-annotator agreement estimates. For example, Weiss et al. (2015) report 94.26 UAS and Andor et al. (2016) report 94.61 UAS on section 23 of the PTB-WSJ using an automatic conversion of the PTB phrase structure trees to Stanford dependencies (De Marneffe et al., 2006). These results are not fully comparable to ours due to differences in the utilized dependency formalism and the automatic conversion of the annotations. Nonetheless, we believe that the similarities in the tasks and evaluation data are sufficiently strong to indicate that dependency parsing for standard English newswire may be reaching human agreement levels.

6 Related Work

The term "anchoring" was coined in a seminal paper by Tversky and Kahneman (1974), which demonstrated that numerical estimation can be biased by uninformative prior information. Subsequent work across various domains of decision making confirmed the robustness of anchoring using both informative and uninformative anchors (Furnham and Boo, 2011). Pertinent to our study, anchoring biases were also demonstrated when the participants were domain experts, although to a lesser degree than in the early anchoring experiments (Wilson et al., 1996; Mussweiler and Strack, 2000).

Prior work in NLP examined the influence of pre-tagging (Fort and Sagot, 2010) and pre-parsing (Skjærholt, 2013) on human annotations. Our work introduces a systematic study of this topic using a novel experimental framework as well as substantially more sentences and annotators. Differently from these studies, our methodology enables characterizing annotation bias as anchoring and measuring its effect on tagger and parser evaluations.

Our study also extends the POS tagging experiments of Marcus et al. (1993), which compared inter-annotator agreement and annotation quality on manual POS tagging in annotation from scratch and tagger-based review conditions. The first result reported in that study was that tagger-based editing increases inter-annotator agreement compared to annotation from scratch. Our work provides a novel agreement benchmark for POS tagging which reduces annotation errors through a review process while controlling for tagger bias, and obtains agreement measurements for dependency parsing. The second result reported in Marcus et al. (1993) was that tagger-based edits are of higher quality compared to annotations from scratch when evaluated against an additional independent annotation. We modify this experiment by introducing ranking as an alternative mechanism for quality assessment, and adding a review round for human annotations from scratch. Our experiment demonstrates that in this configuration, parser-based annotations are of *lower* quality compared to human-based annotations.

Several estimates of expert inter-annotator agreement for English parsing were previously reported. However, most such evaluations were conducted using annotation setups that can be affected by an anchoring bias (Carroll et al., 1999; Rambow et al., 2002; Silveira et al., 2014). A notable exception is the study of Sampson and Babarczy (2008) who measure agreement on annotation from scratch for English parsing in the SUSANNE framework (Sampson, 1995). The reported results, however, are not directly comparable to ours, due to the use of a substantially different syntactic representation, as well as a different agreement metric. Their study further suggests that despite the high expertise of the annotators, the main source of annotation disagreements was annotation errors. Our work alleviates this issue by using annotation reviews, which reduce the number of erroneous annotations while maintaining the independence of the annotation sources. Experiments on non-expert dependency annotation from scratch were previously reported for French, suggesting low agreement rates (79%) with an expert annotation benchmark (Gerdes, 2013).

7 Discussion

We present a systematic study of the impact of anchoring on POS and dependency annotations used in NLP, demonstrating that annotators exhibit an anchoring bias effect towards the output of automatic annotation tools. This bias leads to an artificial boost of performance figures for the parsers in question and results in lower annotation quality as compared with human-based annotations.

Our analysis demonstrates that despite the adverse effects of parser bias, predictions that are shared across different parsers do not significantly lower the quality of the annotations. This finding gives rise to the following *hybrid annotation* strategy as a potential future alternative to human-based as well as parser-based annotation pipelines. In a hybrid annotation setup, human annotators review annotations on which several parsers agree, and complete the remaining annotations from scratch. Such a strategy would largely maintain the annotation speed-ups of parser-based annotation schemes. At the same time, it is expected to achieve annotation quality comparable to human-based annotation by avoiding parser specific bias, which plays a pivotal role in the reduced quality of single-parser reviewing pipelines.

Further on, we obtain, to the best of our knowl-

edge for the first time, syntactic inter-annotator agreement measurements on WSJ-PTB sentences. Our experimental procedure reduces annotation errors and controls for parser bias. Despite the detailed annotation guidelines, the extensive experience of our annotators, and their prior work as a group, our experiment indicates rather low agreement rates, which are below state of the art tagging performance and on par with state of the art parsing results on this dataset. We note that our results do not necessarily reflect an upper bound on the achievable syntactic inter-annotator agreement for English newswire. Higher agreement rates could in principle be obtained through further annotator training, refinement and revision of annotation guidelines, as well as additional automatic validation tests for the annotations. Nonetheless, we believe that our estimates reliably reflect a realistic scenario of expert syntactic annotation.

The obtained agreement rates call for a more extensive examination of annotator disagreements on parsing and tagging. Recent work in this area has already proposed an analysis of expert annotator disagreements for POS tagging in the absence of annotation guidelines (Plank et al., 2014). Our annotations will enable conducting such studies for annotation with guidelines, and support extending this line of investigation to annotations of syntactic dependencies. As a first step towards this goal, we plan to carry out an in-depth analysis of disagreement in the collected data, characterize the main sources of inconsistent annotation and subsequently formulate further strategies for improving annotation accuracy. We believe that better understanding of human disagreements and their relation to disagreements between humans and parsers will also contribute to advancing evaluation methodologies for POS tagging and syntactic parsing in NLP, an important topic that has received only limited attention thus far (Schwartz et al., 2011; Plank et al., 2015).

Finally, since the release of the Penn Treebank in 1992, it has been serving as the standard benchmark for English parsing evaluation. Over the past few years, improvements in parsing performance on this dataset were obtained in small increments, and are commonly reported without a linguistic analysis of the improved predictions. As dependency parsing performance on English newswire may be reaching human expert agreement, not only new evaluation practices, but also more attention to noisier domains and other languages may be in place.

Acknowledgments

We thank our terrific annotators Sebastian Garza, Jessica Kenney, Lucia Lam, Keiko Sophie Mori and Jing Xian Wang. We are also grateful to Karthik Narasimhan and the anonymous reviewers for valuable feedback on this work. This material is based upon work supported by the Center for Brains, Minds, and Machines (CBMM) funded by NSF STC award CCF-1231216. This work was also supported by AFRL contract No. FA8750-15-C-0010 and by ERC Consolidator Grant LEXICAL (648909).

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442–2452.

Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza, and Boris Katz. 2016. Universal dependencies for learner english. In *Proceedings of ACL*, pages 737–746.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. *arXiv preprint cs/9907013*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592.

Karën Fort and Benoît Sagot. 2010. Influence of pre-annotation on pos-tagged corpus development. In *Proceedings of the fourth linguistic annotation workshop*, pages 56–63.

Adrian Furnham and Hua Chu Boo. 2011. A literature review of the anchoring effect. *The Journal of Socio-Economics*, 40(1):35–42.

Kim Gerdes. 2013. Collaborative dependency annotation. *DepLing 2013*, 88.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of ACL*, volume 1, pages 1381–1391.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

André FT Martins, Miguel Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL*, pages 617–622.

Thomas Mussweiler and Fritz Strack. 2000. Numeric judgments under uncertainty: The role of knowledge in anchoring. *Journal of Experimental Social Psychology*, 36(5):495–518.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Linguistically debatable or just plain wrong? In *Proceedings of ACL: Short Papers*, pages 507–511.

Barbara Plank, Héctor Martínez Alonso, Željko Agić, Danijela Merkler, and Anders Søgaard. 2015. Do dependency parsing metrics correlate with human judgments? In *Proceedings of CoNLL*.

Owen Rambow, Cassandre Creswell, Rachel Szekely, Harriet Taber, and Marilyn A Walker. 2002. A dependency treebank for english. In *Proceedings of LREC*.

Geoffrey Sampson and Anna Babarczy. 2008. Definitional and human constraints on structural annotation of english. *Natural Language Engineering*, 14(04):471–494.

Geoffrey Sampson. 1995. English for the computer: Susanne corpus and analytic scheme.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of ACL*, pages 663–672.

Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In *Proceedings of LREC*, pages 2897–2904.

Arne Skjærholt. 2013. Influence of preprocessing on dependency syntax annotation: speed and agreement. *LAW VII & ID*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pages 173–180.

Amos Tversky and Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL*, pages 323–333.

Timothy D Wilson, Christopher E Houston, Kathryn M Etling, and Nancy Brekke. 1996. A new look at anchoring effects: basic anchoring and its antecedents. *Journal of Experimental Psychology: General*, 125(4):387.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of ACL*, pages 180–189.

Tense Manages to Predict Implicative Behavior in Verbs

Ellie Pavlick
University of Pennsylvania
epavlick@seas.upenn.edu

Chris Callison-Burch
University of Pennsylvania
ccb@cis.upenn.edu

Abstract

Implicative verbs (e.g. *manage*) entail their complement clauses, while non-implicative verbs (e.g. *want*) do not. For example, while *managing to solve the problem* entails *solving the problem*, no such inference follows from *wanting to solve the problem*. Differentiating between implicative and non-implicative verbs is therefore an essential component of natural language understanding, relevant to applications such as textual entailment and summarization. We present a simple method for predicting implicativeness which exploits known constraints on the tense of implicative verbs and their complements. We show that this yields an effective, data-driven way of capturing this nuanced property in verbs.

1 Overview

Understanding language requires the ability to perform basic inference– to make conclusions about what is likely true or false based on what is said. For example, given the sentence *She fixed the bug*, we should almost certainly infer that *the bug is fixed*. However, rather than stating plainly that *She fixed the bug*, one might instead say:

(1a) She **managed to fix** the bug before midnight.

(1b) She **happened to fix** the bug while refactoring.

In either case, the hearer should still infer that *the bug is fixed*. But it is not as easy as always inferring that embedded clauses are true. By changing only one word, these sentence no longer give a clear indication as to whether or not the bug has been fixed:

(2a) She **wanted to fix** the bug before midnight.

(2b) She **planned to fix** the bug while refactoring.

Implicative verbs, like those in (1), give rise to entailments, while non-implicative verbs, like those in (2), do not. It is therefore vital to natural language understanding to differentiate between clauses that are embedded under implicatives, which we can often infer to be either true or false, and those which are embedded under non-implicatives, for which such inferences cannot be made. In this paper, we exploit a known linguistic property of implicative verbs– that their complement clause is constrained to be in the same tense as the main clause– in order to predict the tendency of verbs to behave implicatively. We show that our method almost perfectly separates non-implicatives from implicatives in a small hand-labeled dataset, and that it provides strong signal for predicting entailments in sentences involving implicative verbs.

2 Implicative Verbs

Some English verbs can take infinitival complements, meaning they can appear in constructions of the form VB_1^* *to* VB_2, where VB_1^* is the "main" verb (which can be conjugated[1]) and VB_2 is the "complement" verb (which is in infinitive form). Examples (1a)-(2b) illustrate verbs taking infinitive complements.

Implicative verbs are a special subclass[2] of such verbs which give rise to entailments involving their

[1]Here, * indicates that VB_1 can match any verb form, e.g. VB, VBD, VBP, etc. VB_2 can only match the base form VB.

[2]We note that *factive verbs* represent another special class of verbs which can take infinitival complements. Unlike implica-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2225–2229,
Austin, Texas, November 1-5, 2016.

	Is the main verb negated?	Is the complement entailed?	Example
Implicative	−	Yes	I managed to solve the problem. ⇒ I solved the problem.
Implicative	+	No	I did not manage to solve the problem. ⇒ I did not solve the problem.
Implicative	−	No	I failed to solve the problem. ⇒ I did not solve the problem.
Implicative	+	Yes	I did not fail to solve the problem. ⇒ I solved the problem.
Non-Impl.	−	Unknown	I wanted to solve the problem. ⇏ I solved the problem.
Non-Impl.	+	Unknown	I did not want to solve the problem. ⇏ I did not solve the problem.

Table 1: Implicative verbs give rise to entailments involving their complement clauses. Non-implicatives entail neither the truth nor the falsity of their complements, and thus the truth/falsity of the complement is unaffected by negation of the main clause.

complement clauses. Individual implicatives can differ in the entailments they generate: e.g. while *manage* entails the truth of its complement, *fail* entails the falsity of its complement (*failed to solve the problem ⇒ didn't solve the problem*). Despite these differences, however, implicatives represent a coherent class of verbs in that they permit some inference to be made about their complements, and this inference is sensitive to the context (positive/negated) of the main clause. This contrasts with *non-implicative verbs*, like *want*, which do not permit any inference regarding their complements, and for which the truth of the complement is unaffected by negation in the main clause (Table 1).

The method described in this paper aims to separate implicatives from non-implicatives (*manage* vs. *want*), rather than to differentiate between types implicatives (*manage* vs. *fail*). Making this implicative/non-implicative distinction is a necessary first step toward handling inferences involving embedded clauses, and one that, to date, has only been performed using manually-constructed word lists (MacCartney, 2009; Recasens et al., 2013).

2.1 Tense Constraints on Complement Clauses

Karttunen (1971) observed that, in sentences involving implicatives, the tense of the main verb must necessarily match the tense of the complement clause. For example, (3), in which the main clause and the complement are both in the past tense, is acceptable but (4), in which the complement is in the future, is clearly not. For non-implicatives, however, no such constraint exists: (6) is perfectly felicitous.

(3) I managed to solve the problem last night.

(4) #I managed to solve the problem tomorrow.

(5) I planned to solve the problem last night.

(6) I planned to solve the problem tomorrow.

We exploit this property to predict *implicativeness*– whether the truth of a verb's complement can be inferred– by observing the verb's usage in practice.

3 Method

We hypothesize that, given a large corpus, we should be able to distinguish implicative verbs from non-implicative verbs by observing how often the main verb tense agrees/disagrees with the tense of the complement clause. Unfortunately, verbs in infinitival complement clauses are not conjugated, and so are not necessarily marked for tense. We therefore use the Stanford Temporal Tagger (TT) (Chang and Manning, 2012) in order to identify time-referring expressions (e.g. *tomorrow* or *last night*) and resolve them to either past, present, or future tense.

We find all sentences containing VB_1^* *to* VB_2 constructions in the Annotated Gigaword corpus (Napoles et al., 2012). We run the the TT over all of the sentences in order to identify time-referring expressions. We only consider sentences in which a time-referring expression appears and is in a direct dependency relationship with the complement verb (VB_2). We provide the TT with the document publication dates,[3] which are used to resolve each time mention to a specific calendar date and time. We then map these time expressions coarsely to either past, present, or future tense by comparing the

tives, factives presuppose, rather than entail, their complements. E.g. both *I was/was not glad to solve the problem* entail *I solved the problem*. We do not address factives here, as factives rarely take infinitival complements: more often, they take "that" complements (e.g. *know that, realize that*). Factives that do take infinitival complements are mostly phrasal (e.g. *be glad to*).

[3] Provided as metadata in the Annotated Gigaword.

resolved time to the document creation time. Because of the fact that days were often not resolved correctly, or at all, we eventually throw away sentences in which the complement clause is labeled as present tense, as these are rarely true references to the present, and rather the result of incorrect time resolution, or implicit future references (e.g. *I am going to solve the problem today* implies the future as in *later today*, but this is not captured by the TT). We also assign the main clause to past, present, or future tense by using the fine-grained POS tag and a set of heuristics (for example, to check for modals).[4]

We assign a *tense agreement* score to each verb v as follows. Let S be the set of all VB_1^* *to* VB_2 constructions in which $VB_1^* = v$. Then tense agreement is simply $\frac{1}{|S|} \times |\{s \in S \mid \text{complement tense = main tense}\}|$, i.e. the fraction of constructions in S in which the tenses of the main and complement clauses agree. We expect implicative verbs to occur mostly in agreeing constructions, and thus have high tense agreement, while non-implicatives may occur in both agreeing and non-agreeing constructions, and thus should have lower tense agreement. Note that while in theory, implicatives should never appear in non-agreeing constructions, the time annotation process is very imprecise, and thus we do not expect perfect results.

4 Evaluation

Recreating list from Karttunen (1971). Karttunen (1971) provides a short illustrative list of 7 known implicatives[5] and 8 non-implicatives (shown in Table 2). As a first evaluation, we test whether tense agreement can accurately separate the verbs in this list, such that the implicatives are assigned higher agreement scores than the non-implicatives. Table 2 shows that this is indeed the case. Tense agreement almost perfectly divides the list, with implicative verbs appearing above non-implicative verbs in all cases. The one exception is *decide* (reportedly non-implicative), which appears above *dare* (reportedly implicative). This error, however, seems understandable: while *decide* is not strictly implicative in the way *manage* is, it is often used as an implicative. E.g. the sentence *I decided to leave* would likely be taken to mean *I left*.

venture to	1.00	try to	0.42
forget to	0.80	agree to	0.34
manage to	0.79	promise to	0.22
bother to	0.61	want to	0.14
happen to	0.59	intend to	0.12
get to	0.52	plan to	0.10
decide to	0.45	hope to	0.03
dare to	0.44		

Table 2: Tense agreement scores for known implicatives (bold) and non-implicatives listed in Karttunen (1971). Ranking by tense agreement almost perfectly divides the two classes.

Predicting Entailment. Our interest is not in distinguishing implicatives from non-implicatives for its own sake, but rather to predict, based on the main verb, whether the truth of the complement can be inferred. We therefore conduct a second evaluation to assess how well tense agreement predicts this entailment property. We design our evaluation following the recognizing textual entailment (RTE) task (Dagan et al., 2006), in which two sentences are given, a premise p and a hypothesis h, and the goal is to determine whether p reasonably entails h. To construct our p/h pairs, we take all the verbs extracted in Section 3 which appear in at least 50 tense-labeled sentences. For each of these verbs, we choose 3 random sentences in which the verb appears as VB_1^* in a VB_1^* *to* VB_2 construction.[6] From each sentence, we extract the complement clause by deleting VB_1^* *to* from the sentence, and conjugating VB_2 to match the tense of VB_1^*. We then use the original sentence as p and the extracted complement as h: e.g. a p/h pair might look like *I get to interact with fellow professors/I interact with fellow professors*. We ask 5 independent annotators on Amazon Mechanical Turk to read each p and then determine whether h is true, false, or unclear given p.[7] We take the majority answer as the true label. We expect that implicative verbs should lead to judgements which are decidedly true or false while non-implicatives should lead

[4] Full set of heuristics in supplementary material.

[5] The original list had 8 implicatives, but we omit *remember* since, in our data, it occurred almost exclusively with recurring time expressions, which we were not able to map to a specific date/time and thus tense, e.g. *consumers must remember to make payments every 14 days*.

[6] These sentences can come from anywhere in the Gigaword corpus, they are not required to contain time expressions.

[7] Full annotation guidelines in supplementary material.

(**0.14**) UFJ **wants to merge** with Mitsubishi, a combination that'd surpass Citigroup as the world's biggest bank. $\not\Rightarrow$ The merger of Japanese Banks creates the world's biggest bank.
(**0.55**) After graduating, Gallager **chose to accept** a full scholarship to play football for Temple University. $\Rightarrow$ Gallager attended Temple University.
(**0.68**) Wilkins **was allowed to leave** in 1987 to join French outfit Paris Saint-Germain. $\Rightarrow$ Wilkins departed Milan in 1987.

Table 3: Examples from the RTE3 dataset (Giampiccolo et al., 2007) which require recognizing implicative behavior, even in verbs that are not implicative by definition. The tendency of certain verbs (e.g. *be allowed*) to behave as *de facto* implicatives is captured surprisingly well by the tense agreement score (shown in parentheses).

to mostly judgements of unclear.

Figure 1 shows that these expectations hold. When a verb with low tense agreement appeared as the main verb of a sentence, the truth of the complement could only be inferred 30% of the time. When a verb with high tense agreement appeared as the main verb, the truth of the complement could be inferred 91% of the time. This difference is significant at $p < 0.01$. That is, tense agreement provides a strong signal for identifying non-implicative verbs, and thus can help systems avoid false-positive entailment judgements, e.g. incorrectly inferring that *wanting to merge* $\Rightarrow$ *merging* (Table 3).

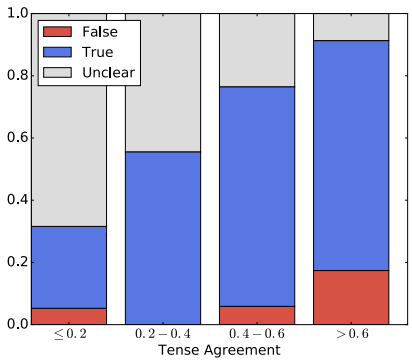


Figure 1: Whether or not complement is entailed for main verbs with varying levels of tense agreement. Verbs with high tense agreement yield more definitive judgments (true/false). Each bar represents aggregated judgements over approx. 20 verbs.

Interestingly, tense agreement accurately models verbs that are not implicative by definition, but which nonetheless tend to behave implicatively in practice. For example, our method finds high tense agreement for *choose to* and *be allowed to*, which are often used to communicate, albeit indirectly, that their complements did in fact happen. To convince ourselves that treating such verbs as implicatives makes sense in practice, we manually look through the RTE3 dataset (Giampiccolo et al., 2007) for examples containing high-scoring verbs according to our method. Table 3 shows some example inferences that hinge precisely on recognizing these types of *de facto* implicatives.

5 Discussion and Related Work

Language understanding tasks such as RTE (Clark et al., 2007; MacCartney, 2009) and bias detection (Recasens et al., 2013) have been shown to require knowledge of implicative verbs, but such knowledge has previously come from manually-built word lists rather than from data. Nairn et al. (2006) and Martin et al. (2009) describe automatic systems to handle implicatives, but require hand-crafted rules for each unique verb that is handled. The tense agreement method we present offers a starting point for acquiring such rules from data, and is well-suited for incorporating into statistical systems. The clear next step is to explore similar data-driven means for learning the specific behaviors of individual implicative verbs, which has been well-studied from a theoretical perspective (Karttunen, 1971; Nairn et al., 2006; Amaral et al., 2012; Karttunen, 2012). Another interesting extension concerns the role of tense in word representations. While currently, tense is rarely built directly into distributional representations of words (Mikolov et al., 2013; Pennington et al., 2014), our results suggest it may offer important insights into the semantics of individual words. We leave this question as a direction for future work.

6 Conclusion

Differentiating between implicative and non-implicative verbs is important for discriminating inferences that can and cannot be made in natural language. We have presented a data-driven method

that captures the implicative tendencies of verbs by exploiting the tense relationship between the verb and its complement clauses. This method effectively separates known implicatives from known non-implicatives, and, more importantly, provides good predictive signal in an entailment recognition task.

Acknowledgments

We would like to thank Florian Schwartz for valuable discussions. This research was supported by a Facebook Fellowship, and by gifts from the Alfred P. Sloan Foundation, Google, and Facebook. This material is based in part on research sponsored by the NSF grant under IIS-1249516 and DARPA under number FA8750-13-2-0017 (the DEFT program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

References

Patricia Amaral, Valeria de Paiva, Cleo Condoravdi, and Annie Zaenen. 2012. Where's the meeting that was cancelled? existential implications of transitive verbs. In *Proceedings of the 3rd Workshop on Cognitive Aspects of the Lexicon*, pages 183–194, Mumbai, India, December. The COLING 2012 Organizing Committee.

Angel X Chang and Christopher D Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740.

Peter Clark, Phil Harrison, John Thompson, William Murray, Jerry Hobbs, and Christiane Fellbaum. 2007. On the role of lexical and world knowledge in rte3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 54–59, Prague, June. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognizing textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognising textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Lauri Karttunen. 1971. Implicative verbs. *Language*, pages 340–358.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 124–131, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Fabienne Martin, Dennis Spohr, and Achim Stein. 2009. Disambiguation of polysemous verbs for rule-based inferencing. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 222–234, Tilburg, The Netherlands, January. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. *Inference in Computational Semantics (ICoS-5)*, pages 20–21.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada, June. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659, Sofia, Bulgaria, August. Association for Computational Linguistics.

Who did What: A Large-Scale Person-Centered Cloze Dataset

Takeshi Onishi Hai Wang Mohit Bansal Kevin Gimpel David McAllester
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{tonishi,haiwang,mbansal,kgimpel,mcallester}@ttic.edu

Abstract

We have constructed a new "Who-did-What" dataset of over 200,000 fill-in-the-gap (cloze) multiple choice reading comprehension problems constructed from the LDC English Gigaword newswire corpus. The WDW dataset has a variety of novel features. First, in contrast with the CNN and Daily Mail datasets (Hermann et al., 2015) we avoid using article summaries for question formation. Instead, each problem is formed from two independent articles — an article given as the passage to be read and a separate article on the same events used to form the question. Second, we avoid anonymization — each choice is a person named entity. Third, the problems have been filtered to remove a fraction that are easily solved by simple baselines, while remaining 84% solvable by humans. We report performance benchmarks of standard systems and propose the WDW dataset as a challenge task for the community.[1]

1 Introduction

Researchers distinguish the problem of general knowledge question answering from that of reading comprehension (Hermann et al., 2015; Hill et al., 2016). Reading comprehension is more difficult than knowledge-based or IR-based question answering in two ways. First, reading comprehension systems must infer answers from a given unstructured passage rather than structured knowledge sources such as Freebase (Bollacker et al., 2008) or the Google Knowledge Graph (Singhal, 2012). Second, machine comprehension systems cannot exploit the large level of redundancy present on the web to find statements that provide a strong syntactic match to the question (Yang et al., 2015). In contrast, a machine comprehension system must use the single phrasing in the given passage, which may be a poor syntactic match to the question.

In this paper, we describe the construction of a new reading comprehension dataset that we refer to as "Who-did-What". Two typical examples are shown in Table 1.[2] The process of forming a problem starts with the selection of a question article from the English Gigaword corpus. The question is formed by deleting a person named entity from the first sentence of the question article. An information retrieval system is then used to select a passage with high overlap with the first sentence of the question article, and an answer choice list is generated from the person named entities in the passage.

Our dataset differs from the CNN and Daily Mail comprehension tasks (Hermann et al., 2015) in that it forms questions from two distinct articles rather than summary points. This allows problems to be derived from document collections that do not contain manually-written summaries. This also reduces the syntactic similarity between the question and the relevant sentences in the passage, increasing the need for deeper semantic analysis.

To make the dataset more challenging we selectively remove problems so as to suppress four simple

[1] Available at tticnlp.github.io/who_did_what

[2] The passages here only show certain salient portions of the passage. In the actual dataset, the entire article is given. The correct answers are (3) and (2).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2230–2235,
Austin, Texas, November 1-5, 2016.

Passage: Britain's decision on Thursday to drop extradition proceedings against Gen. Augusto Pinochet and allow him to return to Chile is understandably frustrating ... Jack Straw, the home secretary, said the 84-year-old former dictator's ability to understand the charges against him and to direct his defense had been seriously impaired by a series of strokes. ... Chile's president-elect, Ricardo Lagos, has wisely pledged to let justice run its course. But the outgoing government of President Eduardo Frei is pushing a constitutional reform that would allow Pinochet to step down from the Senate and retain parliamentary immunity from prosecution. ...

Question: Sources close to the presidential palace said that Fujimori declined at the last moment to leave the country and instead he will send a high level delegation to the ceremony, at which Chilean President Eduardo Frei will pass the mandate to XXX.

Choices: (1) Augusto Pinochet (2) Jack Straw (3) Ricardo Lagos

Passage: Tottenham won 2-0 at Hapoel Tel Aviv in UEFA Cup action on Thursday night in a defensive display which impressed Spurs skipper Robbie Keane. ... Keane scored the first goal at the Bloomfield Stadium with Dimitar Berbatov, who insisted earlier on Thursday he was happy at the London club, heading a second. The 26-year-old Berbatov admitted the reports linking him with a move had affected his performances ... Spurs manager Juande Ramos has won the UEFA Cup in the last two seasons ...

Question: Tottenham manager Juande Ramos has hinted he will allow XXX to leave if the Bulgaria striker makes it clear he is unhappy.

Choices: (1) Robbie Keane (2) Dimitar Berbatov

Table 1: Sample reading comprehension problems from our dataset.

baselines — selecting the most mentioned person, the first mentioned person, and two language model baselines. This is also intended to produce problems requiring deeper semantic analysis.

The resulting dataset yields a larger gap between human and machine performance than existing ones. Humans can answer questions in our dataset with an 84% success rate compared to the estimates of 75% for CNN (Chen et al., 2016) and 82% for the CBT named entities task (Hill et al., 2016). In spite of this higher level of human performance, various existing readers perform significantly worse on our dataset than they do on the CNN dataset. For example, the Attentive Reader (Hermann et al., 2015) achieves 63% on CNN but only 55% on Who-did-What and the Attention Sum Reader (Kadlec et al., 2016) achieves 70% on CNN but only 59% on Who-did-What.

In summary, we believe that our Who-did-What dataset is more challenging, and requires deeper semantic analysis, than existing datasets.

2 Related Work

Our Who-did-What dataset is related to several recently developed datasets for machine comprehension. The MCTest dataset (Richardson et al., 2013) consists of 660 fictional stories with 4 multiple choice questions each. This dataset is too small to train systems for the general problem of reading comprehension.

The bAbI synthetic question answering dataset (Weston et al., 2016) contains passages describing a series of actions in a simulation followed by a question. For this synthetic data a logical algorithm can be written to solve the problems exactly (and, in fact, is used to generate ground truth answers).

The Children's Book Test (CBT) dataset, created by Hill et al. (2016), contains 113,719 cloze-style named entity problems. Each problem consists of 20 consecutive sentences from a children's story, a 21st sentence in which a word has been deleted, and a list of ten choices for the deleted word. The CBT dataset tests story completion rather than reading comprehension. The next event in a story is often not determined — surprises arise. This may explain why human performance is lower for CBT than for our dataset — 82% for CBT vs. 84% for Who-did-What. The 16% error rate for humans on Who-did-What seems to be largely due to noise in problem formation introduced by errors in named entity recognition and parsing. Reducing this noise in future versions of the dataset should significantly improve human performance. Another difference compared to CBT is that Who-did-What has shorter choice lists on average. Random guessing achieves only 10% on CBT but 32% on Who-did-What. The reduction

in the number of choices seems likely to be responsible for the higher performance of an LSTM system on Who-did-What – contextual LSTMs (the attentive reader of Hermann et al., 2015) improve from 44% on CBT (as reported by Hill et al., 2016) to 55% on Who-did-What.

Above we referenced the comprehension datasets created from CNN and Daily Mail articles by Hermann et al. (2015). The CNN and Daily Mail datasets together consist of 1.4 million questions constructed from approximately 300,000 articles. Of existing datasets, these are the most similar to Who-did-What in that they consist of cloze-style question answering problems derived from news articles. As discussed in Section 1, our Who-did-What dataset differs from these datasets in not being derived from article summaries, in using baseline suppression, and in yielding a larger gap between machine and human performance. The Who-did-What dataset also differs in that the person named entities are not anonymized, permitting the use of external resources to improve performance while remaining difficult for language models due to suppression.

3 Dataset Construction

We now describe the construction of our Who-did-What dataset in more detail. To generate a problem we first generate the question by selecting a random article — the "question article" — from the Gigaword corpus and taking the first sentence of that article — the "question sentence" — as the source of the cloze question. The hope is that the first sentence of an article contains prominent people and events which are likely to be discussed in other independent articles. To convert the question sentence to a cloze question, we first extract named entities using the Stanford NER system (Finkel et al., 2005) and parse the sentence using the Stanford PCFG parser (Klein and Manning, 2003).

The person named entities are candidates for deletion to create a cloze problem. For each person named entity we then identify a noun phrase in the automatic parse that is headed by that person. For example, if the question sentence is "President Obama met yesterday with Apple Founder Steve Jobs" we identify the two person noun phrases "President Obama" and "Apple Founder Steve Jobs". When a person named entity is selected for deletion, the entire noun phrase is deleted. For example, when deleting the second named entity, we get "President Obama met yesterday with XXX" rather than "President Obama met yesterday with Apple founder XXX". This increases the difficulty of the problems because systems cannot rely on descriptors and other local contextual cues. About 700,000 question sentences are generated from Gigaword articles (8% of the total number of articles).

Once a cloze question has been formed we select an appropriate article as a passage. The article should be independent of the question article but should discuss the people and events mentioned in the question sentence. To find a passage we search the Gigaword dataset using the Apache Lucene information retrieval system (McCandless et al., 2010), using the question sentence as the query. The named entity to be deleted is included in the query and required to be included in the returned article. We also restrict the search to articles published within two weeks of the date of the question article. Articles containing sentences too similar to the question in word overlap and phrase matching near the blanked phrase are removed. We select the best matching article satisfying our constraints. If no such article can be found, we abort the process and move on to a new question.

Given a question and a passage we next form the list of choices. We collect all person named entities in the passage except unblanked person named entities in the question. Choices that are subsets of longer choices are eliminated. For example the choice "Obama" would be eliminated if the list also contains "Barack Obama". We also discard ambiguous cases where a part of a blanked NE appears in multiple candidate answers, e.g., if a passage has "Bill Clinton" and "Hillary Clinton" and the blanked phrase is "Clinton". We found this simple coreference rule to work well in practice since news articles usually employ full names for initial mentions of persons. If the resulting choice list contains fewer than two or more than five choices, the process is aborted and we move on to a new question.[3]

After forming an initial set of problems we then

[3]The maximum of five helps to avoid sports articles containing structured lists of results.

remove "duplicated" problems. Duplication arises because Gigaword contains many copies of the same article or articles where one is clearly an edited version of another. Our duplication-removal process ensures that no two problems have very similar questions. Here, similarity is defined as the ratio of the size of the bag of words intersection to the size of the smaller bag.

In order to focus our dataset on the most interesting problems, we remove some problems to suppress the performance of the following simple baselines:

- First person in passage: Select the person that appears first in the passage.
- Most frequent person: Select the most frequent person in the passage.
- n-gram: Select the most likely answer to fill the blank under a 5-gram language model trained on Gigaword minus articles which are too similar to one of the questions in word overlap and phrase matching.
- Unigram: Select the most frequent last name using the unigram counts from the 5-gram model.

To minimize the number of questions removed we solve an optimization problem defined by limiting the performance of each baseline to a specified target value while removing as few problems as possible, i.e.,

$$\max_{\alpha(C)} \sum_{C \in \{0,1\}^{|b|}} \alpha(C)|T(C)| \quad (1)$$

subject to

$$\forall i \quad \sum_{C:C_i=1} \frac{\alpha(C)|T(C)|}{N} \leq k$$

$$N = \sum_{C \in \{0,1\}^{|b|}} \alpha(C)|T(C)| \quad (2)$$

where $T(C)$ is the subset of the questions solved by the subset C of the suppressed baselines, $\alpha(C)$ is a keeping rate for question set $T(C)$, $C_i = 1$ indicates the i-th baseline is in the subset, $|b|$ is the number of baselines, N is a total number of questions, and k is the upper bound for the baselines after suppression. We choose k to yield random performance for the baselines. The performance of the baselines before and after suppression is shown in Table 2. The suppression removed 49.9% of the questions.

Baseline	Accuracy Before	Accuracy After
First person in passage	0.60	0.32
Most frequent person	0.61	0.33
n-gram	0.53	0.33
Unigram	0.43	0.32
Random*	0.32	0.32

Table 2: Performance of suppressed baselines. *Random performance is computed as a deterministic function of the number of times each choice set size appears. Many questions have only two choices and there are about three choices on average.

	relaxed train	train	valid	test
# questions	185,978	127,786	10,000	10,000
avg. # choices	3.5	3.5	3.4	3.4
avg. # tokens	378	365	325	326
vocab. size	347,406		308,602	

Table 3: Dataset statistics.

Table 3 shows statistics of our dataset after suppression. We split the final dataset into train, validation, and test by taking the validation and test to be a random split of the most recent 20,000 problems as measured by question article date. In this way there is very little overlap in semantic subject matter between the training set and either validation or test. We also provide a larger "relaxed" training set formed by applying less baseline suppression (a larger value of k in the optimization). The relaxed training set then has a slightly different distribution from the train, validation, and test sets which are all fully suppressed.

4 Performance Benchmarks

We report the performance of several systems to characterize our dataset:

- Word overlap: Select the choice c inserted to the question q which is the most similar to any sentence s in the passage, i.e., $\mathrm{CosSim}(\mathrm{bag}(c + q), \mathrm{bag}(s))$.
- Sliding window and Distance baselines (and their combination) from Richardson et al. (2013).
- Semantic features: NLP feature based system from Wang et al. (2015).

- Attentive Reader: LSTM with attention mechanism (Hermann et al., 2015).
- Stanford Reader: An attentive reader modified with a bilinear term (Chen et al., 2016).
- Attention Sum (AS) Reader: GRU with a point-attention mechanism (Kadlec et al., 2016).
- Gated-Attention (GA) Reader: Attention Sum Reader with gated layers (Dhingra et al., 2016).

Table 4 shows the performance of each system on the test data. For the Attention and Stanford Readers, we anonymized the Who-did-What data by replacing named entities with entity IDs as in the CNN and Daily Mail datasets.

We see consistent reductions in accuracy when moving from CNN to our dataset. The Attentive and Stanford Reader drop by up to 10% and the AS and GA readers drop by up to 17%. The ranking of the systems also changes. In contrast to the Attentive/Stanford readers, the AS/GA readers explicitly leverage the frequency of the answer in the passage, a heuristic which appears beneficial for the CNN and Daily Mail tasks. Our suppression of the most-frequent-person baseline appears to more strongly affect the performance of these latter systems.

System	WDW	CNN
Word overlap	0.47	–
Sliding window	0.48	–
Distance	0.46	–
Sliding window + Distance	0.51	–
Semantic features	0.52	–
Attentive Reader	0.53	0.63^{I}
Attentive Reader (relaxed train)	0.55	
Stanford Reader	0.64	0.73^{II}
Stanford Reader (relaxed train)	0.65	
AS Reader	0.57	0.70^{III}
AS Reader (relaxed train)	0.59	
GA Reader	0.57	0.74^{IV}
GA Reader (relaxed train)	0.60	
Human Performance	$84/100$	$0.75+^{II}$

Table 4: System performance on test set. Human performance was computed by two annotators on a sample of 100 questions. Result marked I is from (Hermann et al., 2015), results marked II are from (Chen et al., 2016), result marked III is from (Kadlec et al., 2016), and result marked IV is from (Dhingra et al., 2016).

5 Conclusion

We presented a large-scale person-centered cloze dataset whose scalability and flexibility is suitable for neural methods. This dataset is different in a variety of ways from existing large-scale cloze datasets and provides a significant extension to the training and test data for machine comprehension.

Acknowledgments

We thank NVIDIA Corporation for donating GPUs used in this research.

References

[Bollacker et al.2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[Chen et al.2016] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367.

[Dhingra et al.2016] Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *CoRR*, abs/1606.01549.

[Finkel et al.2005] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370.

[Hermann et al.2015] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

[Hill et al.2016] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of International Conference on Learning Representations*.

[Kadlec et al.2016] Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text

understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918.

[Klein and Manning2003] Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 423–430.

[McCandless et al.2010] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition*. Manning Publications Co.

[Richardson et al.2013] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

[Singhal2012] Amit Singhal. 2012. Introducing the knowledge graph: things, not strings. *Official Google blog*.

[Wang et al.2015] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 700–706.

[Weston et al.2016] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of International Conference on Learning Representations*.

[Yang et al.2015] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WIKIQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Building compositional semantics and higher-order inference system for a wide-coverage Japanese CCG parser

Koji Mineshima[1]
mineshima.koji@ocha.ac.jp

Ribeka Tanaka[1]
tanaka.ribeka@is.ocha.ac.jp

Pascual Martínez-Gómez[2]
pascual.mg@aist.go.jp

Yusuke Miyao[3]
yusuke@nii.ac.jp

Daisuke Bekki[1]
bekki@is.ocha.ac.jp

[1]Ochanomizu University
Tokyo, Japan

[2]AIST
Tokyo, Japan

[3]National Institute of Informatics
The Graduate University for Advanced Studies
Tokyo, Japan

Abstract

This paper presents a system that compositionally maps outputs of a wide-coverage Japanese CCG parser onto semantic representations and performs automated inference in higher-order logic. The system is evaluated on a textual entailment dataset. It is shown that the system solves inference problems that focus on a variety of complex linguistic phenomena, including those that are difficult to represent in the standard first-order logic.

1 Introduction

Logic-based semantic representations have played an important role in the study of semantic parsing and inference. For English, several methods have been proposed to map outputs of parsers based on syntactic theories like CCG (Steedman, 2000) onto logical formulas (Bos, 2015). Output formulas have been used in various tasks, including Question Answering (Lewis and Steedman, 2013) and Recognizing Textual Entailment (RTE) (Bos and Markert, 2005; Beltagy et al., 2013; Bjerva et al., 2014).

Syntactic and semantic parsing for Japanese, by contrast, has been dominated by chunk-based dependency parsing and semantic role labelling (Kudo and Matsumoto, 2002; Kawahara and Kurohashi, 2011; Hayashibe et al., 2011). Recently, the method of inducing wide-coverage CCG resources for English (Hockenmaier and Steedman, 2007) has been applied to Japanese and a robust CCG parser based on it has been developed (Uematsu et al., 2015). However, building a method to map CCG trees in Japanese onto logical formulas is not a trivial task, mainly due to the differences in syntactic structures between English and Japanese (Section 3).

There are two primary contributions of this paper. First, based on an in-depth analysis of the syntax-semantics interface in Japanese, we present the first system that compositionally derives semantic representations for a wide-coverage Japanese CCG parser. Output representations are formulas in higher-order logic (HOL) combined with Neo-Davidsonian Event Semantics (Parsons, 1990). Second, we demonstrate the capacity of HOL for textual entailment. We evaluate the system on a Japanese textual entailment dataset (Kawazoe et al., 2015), a dataset constructed in a similar way to the FraCaS dataset for English (Cooper et al., 1994; MacCartney and Manning, 2007). Although it is usually thought that HOL is unfeasible for practical applications, the results show that the entire system is able to perform efficient logical inference on complex linguistic phenomena such as generalized quantifiers and intensional modifiers — phenomena that pose challenges to the standard first-order-logic-based approaches.

2 Background and system overview

This section provides a brief overview of the entire system as applied to RTE, a task of determining whether a given text (T) entails, contradicts, or is just consistent with, a given hypothesis (H). In logic-based approaches, the meanings of T and H are represented by logical formulas; whether the entailment relation holds is typically determined by checking whether $T \to H$ is a theorem in a logical system with the help of a knowledge base.

Currently, first-order logic (FOL) is the most pop-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2236–2242,
Austin, Texas, November 1-5, 2016.

ular logical system used for RTE (Bos and Markert, 2005; Lewis and Steedman, 2013; Bjerva et al., 2014). One advantage of systems based on FOL is that practical general-purpose theorem provers and model-builders are available. However, a drawback is that there are linguistic phenomena that cannot be represented in the standard FOL; a typical example is a generalized quantifier such as *most* (Barwise and Cooper, 1981). Accordingly, it has been standard in formal semantics of natural language to use HOL as a representation language (Montague, 1974). Although HOL does not have general-purpose theorem provers, there is room for developing an automated reasoning system specialized for natural language inference. In general, a higher-order representation makes the logical structure of a sentence more explicit than a first-order encoding does and hence can simplify the process of proof search (Miller and Nadathur, 1986). Recently, based on the evaluation on the FraCaS dataset (Cooper et al., 1994), Mineshima et al. (2015) showed that a higher-order inference system outperformed the Boxer/Nutcracker's first-order system (Bos, 2008) in both speed and accuracy. Likewise, Abzianidze (2015) developed a higher-order prover based on natural logic tableau system and showed that it achieved high accuracy comparable to state-of-the-art results on the SICK dataset (Marelli et al., 2014).

There are three main steps in our pipeline. The focus of this paper is on the last two components.

1. Syntactic parsing Input sentences are mapped onto CCG trees. We use a Japanese CCG parser Jigg (Noji and Miyao, 2016)[1], a statistical parser based on Japanese CCGbank (Uematsu et al., 2015).

2. Semantic parsing CCG derivation trees are compositionally mapped onto semantic representations in HOL. The compositional mapping is implemented via simply typed λ-calculus in the standard way (Bos, 2008; Martínez-Gómez et al., 2016).

3. Logical inference Theorem proving in HOL is performed to check for entailment and contradiction. Axioms and proof-search procedures are largely language-independent, so we use the higher-order inference system of Mineshima et al. (2015)[2] and adapt it for our purpose.

[1] https://github.com/mynlp/jigg

[2] https://github.com/mynlp/ccg2lambda

Syntactic category	Semantic representation
NP	$\lambda NF.\exists x(N(\mathsf{Base},x) \wedge F(x))$
$S\backslash NP_{ga}$	$\lambda QK.Q(\lambda I.I, \lambda x.\exists v(K(\mathsf{Base},v) \wedge (\mathsf{Nom}(v) = x)))$
$S\backslash NP_{ga}\backslash NP_o$	$\lambda Q_2Q_1K.Q_1(\lambda I.I, \lambda x_1.Q_2(\lambda I.I, \lambda x_2.\exists v(K(\mathsf{Base},v) \wedge (\mathsf{Nom}(v) = x_1) \wedge (\mathsf{Acc}(v) = x_2))))$
S/S	$\lambda SK.S(\lambda Jv.K(\lambda v'.(J(v') \wedge \mathsf{Base}(v')), v))$
NP/NP	$\lambda QNF.Q(\lambda Gx.N(\lambda y.(\mathsf{Base}(y) \wedge G(y)), x), F)$

Table 1: Examples of semantic templates. Base is the position in which the base form of a word appears.

3 Compositional Semantics and HOL

3.1 CCG and semantic lexicon

Combinatory Categorial Grammar (CCG) (Steedman, 2000) is a lexicalized grammar formalism suitable for implementing a compositional mapping from syntax to semantics. A syntactic category of CCG is either a basic category such as *S* and *NP* or a functional category of the form X/Y or $X\backslash Y$. The meaning of a sentence is computed from a small number of combinatory rules and the meanings of constituent words. In addition to standard combinatory rules, the Japanese CCG parser uses a small number of unary type-shifting rules (e.g., the relativization rule that changes the category $S\backslash NP$ to NP/NP), to which suitable meaning composition rules are given.

We follow the standard method of building a semantic lexicon in CCG-based logical semantics (Bos, 2008). There are two kinds of lexical entries: (1) *semantic templates* that are schematic entries assigned to syntactic categories, possibly with syntactic features and (2) *lexical entries* directly assigned to a limited number of logical and functional expressions. Lexical entries can be sensitive to a POS tag, a surface form, and other information contained in the parser output. Table 1 shows semantic templates for main syntactic categories. More details will be provided in Section 3.2 and 3.3.

We use a language of standard higher-order logic (simple type theory) (Carpenter, 1997) as a representation language. Expressions in HOL are assigned semantic types. We use three basic types: E (Entity), Ev (Event), and Prop (Proposition). Thus, the semantic types of expressions in our system are defined by the rule

$$\mathsf{T} ::= \mathsf{E} \mid \mathsf{Ev} \mid \mathsf{Prop} \mid \mathsf{T}_1 \Rightarrow \mathsf{T}_2$$

where $\mathsf{T}_1 \Rightarrow \mathsf{T}_2$ is a function type.

First-order language can be taken as a fragment of this system; apart from logical connectives and

$NP^\bullet = ((\mathsf{E} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{E} \Rightarrow \mathsf{Prop}) \Rightarrow (\mathsf{E} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{Prop}$
$S^\bullet = ((\mathsf{Ev} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{Ev} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{Prop}$
$(C1/C2)^\bullet = (C1\backslash C2)^\bullet = C2^\bullet \Rightarrow C1^\bullet$

Figure 1: The mapping from syntactic categories to semantic types. $\Rightarrow$ is right-associative.

quantifiers, all primitive expressions in first-order logic are confined to constant symbols of type E and predicates of type $\mathsf{E} \Rightarrow \mathsf{Prop}$, $\mathsf{E} \Rightarrow \mathsf{E} \Rightarrow \mathsf{Prop}$, and so on. Thus, adopting higher-order language does not lead to the loss of the expressive power of first-order language.

The Japanese CCG parser simplifies the standard CCG and uses two basic categories, *S* and *NP*. Accordingly, a mapping $(\cdot)^\bullet$ from syntactic categories to semantic types can be defined as in Figure 1. Keeping the correspondence between syntactic categories and semantic types in the semantic lexicon guarantees that a well-formed formula is compositionally derived from the meaning assignment to each leaf of a CCG derivation tree.

3.2 Semantic composition for VPs

To model a semantics for VPs in Japanese, we adopt Neo-Davidsonian Event Semantics (Parsons, 1990; Jurafsky and Martin, 2009), which is widely used in the NLP field. For instance, the sentence (1) is analyzed as having the logical form in (2):

(1) ジョン が　ゆっくり 歩い た。
John NOM slowly walk PAST
'John walked slowly'

(2) $\exists v(\mathsf{walk}(v) \wedge (\mathsf{Nom}(v) = \mathsf{john}) \wedge \mathsf{slow}(v) \wedge \mathsf{Past}(v))$

In this approach, verbs are analyzed as 1-place predicates over events; arguments and adjuncts of VPs are also analyzed as event predicates. This semantic uniformity is suitable to handling Japanese syntactic structures in which the arguments of a VP is often implicit and thus the argument-adjunct distinction is less transparent than languages like English (Pietroski, 2005). As is seen in (2), we adopt the unique-role requirement for case markers (Carlson, 1984); for instance, the nominative case marker does not denote the relation $\mathsf{Nom}(v, x)$, as in the event semantics in Boxer (Bos, 2008), but the function $\mathsf{Nom}(v) = x$. This treatment allows us to make use of logical properties of equality and hence is more suited to theorem-proving in our setting.

To derive a semantic representation in event semantics compositionally, we adopt the compositional semantics of VPs in Champollion (2015) and analyze VPs themselves as introducing existential quantification over events. To derive the correct meaning for VP modifiers, the semantic type of a verb is raised so that the verb takes a modifier as argument but not vice versa. Figures 2 and 3 give example derivations.

VP modifiers such as *slowly* license an inference from *John walked slowly* to *John walked*, an inference correctly captured by the formula in (2). In English and Japanese, however, there are intensional VP modifiers that do not license this inference pattern. Thus, the sentence *John almost walked* does not entail *John walked* (Dowty, 1979). While it is not easy to provide a desirable analysis in first-order language (Hobbs, 1985), HOL gives a perspicuous representation:

(3) $\exists v(\mathsf{almost}(\mathsf{walk}, v) \wedge (\mathsf{Nom}(v) = \mathsf{john}) \wedge \mathsf{Past}(v))$

Here, almost is a higher-order predicate having the semantic type $(\mathsf{Ev} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{Ev} \Rightarrow \mathsf{Prop}$. The meaning assignment to VP modifiers of category S/S in Table 1 is for extensional modifiers; an intensional modifier is assigned the representation $\lambda SK.S(\lambda Jv.K(\mathsf{Base}(J), v))$ in the lexical entry, which results in a representation as in (3).

3.3 Semantic composition for NPs

The quantificational structure of an NP plays a crucial role in capturing basic entailment patterns such as monotonicity inference. In the case of English, quantificational structures are specified by the type of determiners (e.g. *a, the, every, some, no*); together with the category distinction between *N* and *NP*, which is supported in English CCGbank (Hockenmaier and Steedman, 2007), one can provide a correct representation for NPs.

By contrast, Japanese is a classifier language, where NPs freely occur without determiners in argument position (Chierchia, 1998). For example, the subject in (4) appears in argument position without accompanying any determiner.

(4) 小さな 犬　が　吠え た。
small dog NOM bark PAST
'A small dog barked'

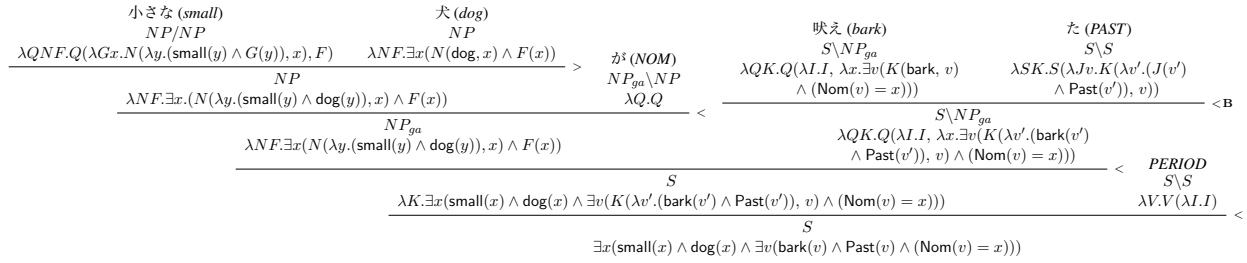


Figure 2: A CCG derivation tree for the sentence "A small dog barked".

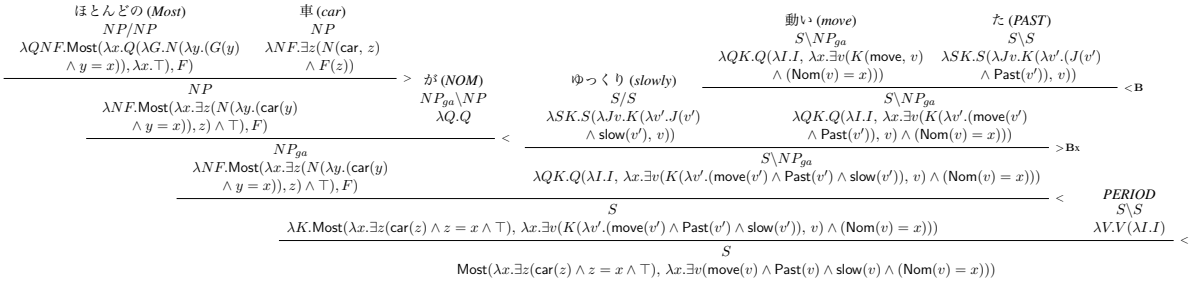


Figure 3: A CCG derivation tree for the sentence "Most cars moved slowly". $\top$ denotes the tautology.

Bekki (2010) provides a comprehensive CCG grammar for Japanese that adopts the *N-NP* distinction and analyzes Japanese bare NPs as accompanying the null determiner. The Japanese CCGbank, by contrast, simplifies Bekki's (2010) grammar and avoids the use of the null determiner; it does not use the category *N* and takes all NPs in Japanese to have the syntactic category *NP*. This discrepancy in NP-structure between English and Japanese poses a challenge to the standard approach to building compositional semantics.

To provide a compositional semantics adapted for the Japanese CCG, we take NPs themselves as introducing quantification over individuals, along the same lines as the semantics for VPs. The semantic type of NPs needs to be raised so that they take NP-modifiers as argument (cf. the template for *NP* in Table 1). Figure 2 shows a derivation for the sentence in (4), where the adjective *small* modifies the NP *dog* to form a bare NP *small dog*. It should be noted that the predicate $\mathsf{small}(x)$ is correctly inserted inside the scope of the existential quantification introduced by the NP *dog*. The so-called privative adjectives (e.g. *fake* and *former*) are analyzed in the same way as intensional VP modifiers.

Following the analysis in Mineshima et al. (2015), we analyze non-first-order generalized quantifier *most* as having the higher-order logical form $\mathsf{Most}(F, G)$, where Most has the type of generalized quantifier $(\mathsf{E} \Rightarrow \mathsf{Prop}) \Rightarrow (\mathsf{E} \Rightarrow \mathsf{Prop}) \Rightarrow \mathsf{Prop}$. Figure 3 shows an example derivation for a sentence containing a generalized quantifier *most*. Our system also handles floating quantifiers in Japanese.

4 Experiments

We evaluate our system[3] on Japanese Semantics test suite (JSeM)[4] (Kawazoe et al., 2015), a Japanese dataset for textual entailment designed in a similar way to the FraCaS dataset for English. These datasets focus on the types of logical inferences that do not require world knowledge. JSeM has Japanese translations of FraCaS problems and an extended set of problems focusing on Japanese syntax and semantics. Each problem has one or more premises, followed by a hypothesis. There are three types of answer: *yes* (entailment), *no* (contradiction), and *unknown* (neutral). Each problem is annotated with the types of inference (logical entailment, presupposition, etc.) and of linguistic phenomena.

We evaluate the system on 523 problems in the dataset. We focus on problems tagged with one of the five phenomena: generalized quantifier, plu-

[3]The system will be available at `https://github.com/mynlp/ccg2lambda`.

[4]`http://researchmap.jp/community-inf/JSeM/?lang=english`

Section	#Problem	Gold	System	SLC
Quantifier	337	**92.3**	78.0	88.4
Plural	41	**68.3**	56.1	51.2
Adjective	65	**67.7**	63.1	44.6
Verb	36	**77.8**	75.0	55.5
Attitude	44	**88.6**	86.4	75.0
Total	523	**86.0**	75.0	76.7

Table 2: Accuracy on each section of JSeM.

	Acc.	Prec.	Rec.	Time
Gold parses	**86.0**	**94.9**	**81.3**	3.30s
w/o HOL axioms	69.8	93.3	56.5	2.47s
System parses	75.0	92.7	65.4	3.58s
SLC	76.7	77.5	79.3	n/a
Most common class (*yes*)	56.8	56.8	85.6	n/a

Table 3: Accuracy, precision, recall, and average proof time.

ral, adjective, verb, and attitude. We use problems whose inference type is logical entailment, excluding anaphora and presupposition. We use Kuromoji[5] for morphological analysis. To focus on the evaluation of semantic parsing and inference, we use gold syntactic parses, which show an upper bound on the performance of the semantic component. Gold syntactic parses are manually selected from n-best outputs of the CCG parser. For the higher-order inference system, we use the axioms presented in Mineshima et al. (2015) adapted with the necessary modification for our event semantics.

Given premises $P_1, \ldots, P_n$ and a hypothesis H, the system outputs *yes* ($P_1 \wedge \cdots \wedge P_n \rightarrow H$ is proved), *no* ($P_1 \wedge \cdots \wedge P_n \rightarrow \neg H$ is proved), or *unknown* (neither is proved in a fixed proof-search space).[6] We set a 30 seconds timeout for each inference run; the system outputs *unknown* after it. The current semantic lexicon has 36 templates and 113 lexical entries.

Table 2 and 3 show the results. The system with gold syntactic parses achieved 86% accuracy on the total 523 problems, with high precision and reasonable speed. There was no timeout.[7] The accuracy dropped to 70% when ablating HOL axioms (Table 3). SLC refers to the performance of a supervised learning classifier[8] based on 5-fold cross-validation for each section. Although direct comparison is not possible, our system with gold parses outperforms it for all sections.

[5] http://www.atilika.org/

[6] Note that natural-logic-based systems (MacCartney and Manning, 2008) do not handle multi-premised problems.

[7] Our pipeline was run single-threaded on Ubuntu Linux 64 bits with a CPU at 2.67GHz.

[8] We used NTCIR RITE baseline tools (http://www.cl.ecei.tohoku.ac.jp/rite2/doku.php).

Section	#Problem	Gold	System	M15
Quantifier	335	92.5	78.2	78.4
Plural	38	65.8	52.6	66.7
Adjective	21	57.1	47.6	68.2
Verb	9	66.7	66.7	62.5
Attitude	14	78.6	78.6	76.9
Total	417	87.3	74.1	73.3

Table 4: The results on a subset of JSeM that is a translation of FraCaS. M15 refers to the accuracy of Mineshima et al. (2015) on the corresponding sections of FraCaS.

Out of the 523 problems, 417 are Japanese translations of the FraCaS problems. Table 4 shows a comparison between the performance of our system on this subset of the JSeM problems and the performance of the RTE system for English in Mineshima et al. (2015) on the corresponding problems in the FraCaS dataset. Mineshima et al. (2015) used system parses of the English C&C parser (Clark and Curran, 2007). The total accuracy of our system is comparable to that of Mineshima et al. (2015).

Most errors we found are due to syntactic parse errors caused by the CCG parser, where no correct syntactic parses were found in n-best responses. Comparison between gold parses and system parses shows that correct syntactic disambiguation improves performance.

5 Conclusion

To our knowledge, this study provides the first semantic parsing system based on CCG that compositionally maps real texts in Japanese onto logical forms. We have also demonstrated the capacity of HOL for textual entailment. The evaluation on JSeM showed that our system performs efficient logical inference on various semantic phenomena, including those that challenge the standard FOL. The attractiveness of a logic-based system is that it is highly modular and can be extended with other components such as a robust knowledge base (Lewis and Steedman, 2013; Beltagy et al., 2013; Bjerva et al., 2014). Such an extension will be a focus of future work.

Acknowledgments We are grateful to the three anonymous reviewers for their helpful comments and suggestions. This research has been supported by the JST CREST program.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Daisuke Bekki. 2010. *A Formal Theory of Japanese Grammar: The Conjugation System, Syntactic Structures, and Semantic Composition*. Kuroshio. (In Japanese).

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *2nd Joint Conference on Lexical and Computational Semantics: Proceeding of the Main Conference and the Shared Task*, pages 11–21.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286.

Johan Bos. 2015. Open-domain semantic parsing with Boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics*, pages 301–304.

Greg Carlson. 1984. Thematic roles and their role in semantic interpretation. *Linguistics*, 22(3):259–280.

Bob Carpenter. 1997. *Type-Logical Semantics*. MIT press.

Lucas Champollion. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, 38(1):31–66.

Gennaro Chierchia. 1998. Reference to kinds across language. *Natural Language Semantics*, 6(4):339–405.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS — a framework for computational semantics. *Deliverable*, D16.

David Dowty. 1979. *Word Meaning and Montague Grammar*. Springer.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of IJCNLP 2011*, pages 201–209.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 60–69.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing*. Prentice-Hall, Inc.

Daisuke Kawahara and Sadao Kurohashi. 2011. Generative modeling of coordination by factoring parallelism and selectional preferences. In *Proceedings of IJCNLP 2011*, pages 456–464.

Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015. An inference problem set for evaluating semantic theories and semantic processing systems for Japanese. In *Proceedings of LENLS12*, pages 67–73.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 63–69.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 521–528.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC2014*, pages 216–223.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: a compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, pages 85–90.

Dale Miller and Gopalan Nadathur. 1986. Some uses of higher-order logic in computational linguistics. In *Proceedings of the 24th annual meeting on Association for Computational Linguistics*, pages 247–256.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061.

Richard Montague. 1974. *Formal Philosophy: Selected Papers*. Yale University Press New Haven.

Hiroshi Noji and Yusuke Miyao. 2016. Jigg: a framework for an easy natural language processing pipeline. In *Proceedings of ACL 2016 System Demonstrations*, pages 103–108.

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press.

Paul Pietroski. 2005. *Events and Semantic Architecture*. Oxford University Press.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Sumire Uematsu, Takuya Matsuzaki, Hiroki Hanaoka, Yusuke Miyao, and Hideki Mima. 2015. Integrating multiple dependency corpora for inducing wide-coverage Japanese CCG resources. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(1):1–24.

Learning to Generate Compositional Color Descriptions

Will Monroe,[1] **Noah D. Goodman,**[2] and **Christopher Potts**[3]
Departments of [1]Computer Science, [2]Psychology, and [3]Linguistics
Stanford University, Stanford, CA 94305
wmonroe4@cs.stanford.edu, {ngoodman, cgpotts}@stanford.edu

Abstract

The production of color language is essential for grounded language generation. Color descriptions have many challenging properties: they can be vague, compositionally complex, and denotationally rich. We present an effective approach to generating color descriptions using recurrent neural networks and a Fourier-transformed color representation. Our model outperforms previous work on a conditional language modeling task over a large corpus of naturalistic color descriptions. In addition, probing the model's output reveals that it can accurately produce not only basic color terms but also descriptors with non-convex denotations ("greenish"), bare modifiers ("bright", "dull"), and compositional phrases ("faded teal") not seen in training.

1 Introduction

Color descriptions represent a microcosm of grounded language semantics. Basic color terms like "red" and "blue" provide a rich set of semantic building blocks in a continuous meaning space; in addition, people employ compositional color descriptions to express meanings not covered by basic terms, such as "greenish blue" or "the color of the rust on my aunt's old Chevrolet" (Berlin and Kay, 1991). The production of color language is essential for referring expression generation (Krahmer and Van Deemter, 2012) and image captioning (Kulkarni et al., 2011; Mitchell et al., 2012), among other grounded language generation problems.

We consider color description generation as a grounded language modeling problem. We present an effective new model for this task that uses a long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997; Graves, 2013) and a Fourier-basis color representation inspired by feature representations in computer vision.

Color	Top-1	Sample
(83, 80, 28)	"green"	"very green"
(232, 43, 37)	"blue"	"royal indigo"
(63, 44, 60)	"olive"	"pale army green"
(39, 83, 52)	"orange"	"macaroni"

Table 1: A selection of color descriptions sampled from our model that were not seen in training. Color triples are in HSL. *Top-1* shows the model's highest-probability prediction.

We compare our model with LUX (McMahan and Stone, 2015), a Bayesian generative model of color semantics. Our model improves on their approach in several respects, which we demonstrate by examining the meanings it assigns to various unusual descriptions: (1) it can generate compositional color descriptions not observed in training (Fig. 3); (2) it learns correct denotations for underspecified modifiers, which name a variety of colors ("dark", "dull"; Fig. 2); and (3) it can model non-convex denotations, such as that of "greenish", which includes both greenish yellows and blues (Fig. 4). As a result, our model also produces significant improvements on several grounded language modeling metrics.

2 Model formulation

Formally, a model of color description generation is a probability distribution $S(d \mid c)$ over sequences of

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2243–2248,
Austin, Texas, November 1-5, 2016.

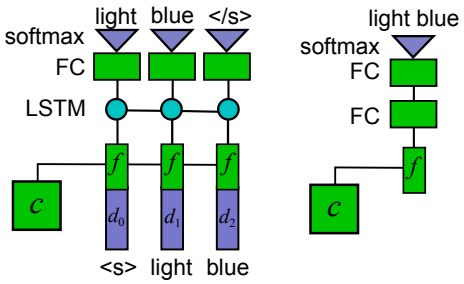


Figure 1: Left: sequence model architecture; right: atomic-description baseline. FC denotes fully connected layers.

tokens d conditioned on a color c, where c is represented as a 3-dimensional real vector in HSV space.[1]

Architecture Our main model is a recurrent neural network sequence decoder (Fig. 1, left panel). An input color $c = (h, s, v)$ is mapped to a representation f (see Color features, below). At each time step, the model takes in a concatenation of f and an embedding for the previous output token d_i, starting with the start token $d_0 =$ <s>. This concatenated vector is passed through an LSTM layer, using the formulation of Graves (2013). The output of the LSTM at each step is passed through a fully-connected layer, and a softmax nonlinearity is applied to produce a probability distribution for the following token.[2] The probability of a sequence is the product of probabilities of the output tokens up to and including the end token </s>.

We also implemented a simple feed-forward neural network, to demonstrate the value gained by modeling descriptions as sequences. This architecture (*atomic*; Fig. 1, right panel) consists of two fully-connected hidden layers, with a ReLU nonlinearity after the first and a softmax output over all full color descriptions seen in training. This model therefore treats the descriptions as atomic symbols rather than sequences.

Color features We compare three representations:

- *Raw*: The original 3-dimensional color vectors, in HSV space.
- *Buckets*: A discretized representation, dividing HSV space into rectangular regions at three resolutions ($90\times10\times10$, $45\times5\times5$, $1\times1\times1$) and assigning a separate embedding to each region.
- *Fourier*: Transformation of HSV vectors into a Fourier basis representation. Specifically, the representation f of a color (h, s, v) is given by

$$\hat{f}_{jk\ell} = \exp\left[-2\pi i \left(jh^* + ks^* + \ell v^*\right)\right]$$
$$f = \begin{bmatrix}\mathrm{Re}\{\hat{f}\} & \mathrm{Im}\{\hat{f}\}\end{bmatrix} \qquad j,k,\ell = 0..2$$

  where $(h^*, s^*, v^*) = (h/360, s/200, v/200)$.

The Fourier representation is inspired by the use of Fourier feature descriptors in computer vision applications (Zhang and Lu, 2002). It is a nonlinear transformation that maps the 3-dimensional HSV space to a 54-dimensional vector space. This representation has the property that most regions of color space denoted by some description are extreme along a single direction in Fourier space, thus largely avoiding the need for the model to learn non-monotonic functions of the color representation.

Training We train using Adagrad (Duchi et al., 2011) with initial learning rate $\eta = 0.1$, hidden layer size and cell size 20, and dropout (Hinton et al., 2012) with a rate of 0.2 on the output of the LSTM and each fully-connected layer. We identified these hyperparameters with random search, evaluating on a held-out subset of the training data.

We use random normally-distributed initialization for embeddings ($\sigma = 0.01$) and LSTM weights ($\sigma = 0.1$), except for forget gates, which are initialized to a constant value of 5. Dense weights use normalized uniform initialization (Glorot and Bengio, 2010).

3 Experiments

We demonstrate the effectiveness of our model using the same data and statistical modeling metrics as McMahan and Stone (2015).

Data The dataset used to train and evaluate our model consists of pairs of colors and descriptions collected in an open online survey (Munroe, 2010). Participants were shown a square of color and asked to write a free-form description of the color in a text box. McMahan and Stone filtered the responses to normalize spelling differences and exclude spam responses and descriptions that occurred

[1] HSV: hue-saturation-value. The visualizations and tables in this paper instead use HSL (hue-saturation-lightness), which yields somewhat more intuitive diagrams and differs from HSV by a trivial reparameterization.

[2] Our implementation uses Lasagne (Dieleman et al., 2015), a neural network library based on Theano (Al-Rfou et al., 2016).

Model	Feats.	Perp.	AIC	Acc.
atomic	raw	28.31	1.08×10^6	28.75%
atomic	buckets	16.01	1.31×10^6	38.59%
atomic	Fourier	15.05	8.86×10^5	38.97%
RNN	raw	13.27	8.40×10^5	40.11%
RNN	buckets	13.03	1.26×10^6	39.94%
RNN	Fourier	**12.35**	$\mathbf{8.33\times10^5}$	**40.40%**
HM	buckets	14.41	4.82×10^6	39.40%
LUX	raw	13.61	4.13×10^6	39.55%
RNN	Fourier	**12.58**	$\mathbf{4.03\times10^6}$	**40.22%**

Table 2: Experimental results. Top: development set; bottom: test set. AIC is not comparable between the two splits. HM and LUX are from McMahan and Stone (2015). We reimplemented HM and re-ran LUX from publicly available code, confirming all results to the reported precision except perplexity of LUX, for which we obtained a figure of 13.72.

very rarely. The resulting dataset contains 2,176,417 pairs divided into training (1,523,108), development (108,545), and test (544,764) sets.

Metrics We quantify model effectiveness with the following evaluation metrics:

- *Perplexity*: The geometric mean of the reciprocal probability assigned by the model to the descriptions in the dataset, conditioned on the respective colors. This expresses the same objective as log conditional likelihood. We follow McMahan and Stone (2015) in reporting perplexity per-description, not per-token as in the language modeling literature.
- *AIC*: The Akaike information criterion (Akaike, 1974) is given by AIC $= 2\ell + 2k$, where ℓ is log likelihood and k is the total number of real-valued parameters of the model (e.g., weights and biases, or bucket probabilities). This quantifies a tradeoff between accurate modeling and model complexity.
- *Accuracy*: The percentage of most-likely descriptions predicted by the model that exactly match the description in the dataset (recall@1).

Results The top section of Table 2 shows development set results comparing modeling effectiveness for atomic and sequence model architectures and different features. The Fourier feature transformation generally improves on raw HSV vectors and discretized embeddings. The value of modeling descriptions as sequences can also be observed in these results; the LSTM models consistently outperform their atomic counterparts.

Additional development set experiments (not shown in Table 2) confirmed smaller design choices for the recurrent architecture. We evaluated a model with two LSTM layers, but we found that the model with only one layer yielded better perplexity. We also compared the LSTM with GRU and vanilla recurrent cells; we saw no significant difference between LSTM and GRU, while using a vanilla recurrent unit resulted in unstable training. Also note that the color representation f is input to the model at every time step in decoding. In our experiments, this yielded a small but significant improvement in perplexity versus using the color representation as the initial state.

Test set results appear in the bottom section. Our best model outperforms both the histogram baseline (HM) and the improved LUX model of McMahan and Stone (2015), obtaining state-of-the-art results on this task. Improvements are highly significant on all metrics ($p < 0.001$, approximate permutation test, $R = 10{,}000$ samples; Padó, 2006).

4 Analysis

Given the general success of LSTM-based models at generation tasks, it is perhaps not surprising that they yield good raw performance when applied to color description. The color domain, however, has the advantage of admitting faithful visualization of descriptions' semantics: colors exist in a 3-dimensional space, so a two-dimensional visualization can show an acceptably complete picture of an entire distribution over the space. We exploit this to highlight three specific improvements our model realizes over previous ones.

We construct visualizations by querying the model for the probability $S(d \mid c)$ of the same description for each color in a uniform grid, summing the probabilities over the hue dimension (left cross-section) and the saturation dimension (right cross-section), normalizing them to sum to 1, and plotting the log of the resulting values as a grayscale image.

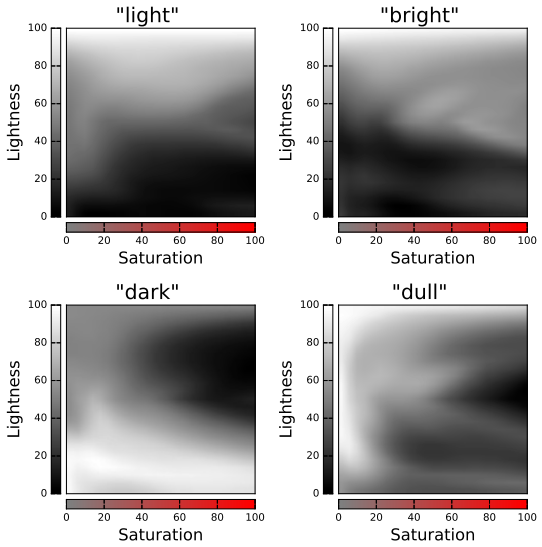


Figure 2: Conditional likelihood of bare modifiers according to our generation model as a function of color. White represents regions of high likelihood. We omit the hue dimension, as these modifiers do not express hue constraints.

Formally, each visualization is a pair of functions (L, R), where

$$L(s,\ell) = \log\left[\frac{\int dh\ S(d \mid c = (h,s,\ell))}{\int dc'\ S(d \mid c')}\right]$$

$$R(h,\ell) = \log\left[\frac{\int ds\ S(d \mid c = (h,s,\ell))}{\int dc'\ S(d \mid c')}\right]$$

The maximum value of each function is plotted as white, the minimum value is black, and intermediate values linearly interpolated.

Learning modifiers Our model learns accurate meanings of adjectival modifiers apart from the full descriptions that contain them. We examine this in Fig. 2, by plotting the probabilities assigned to the bare modifiers "light", "bright", "dark", and "dull". "Light" and "dark" unsurprisingly denote high and low lightness, respectively. Less obviously, they also exclude high-saturation colors. "Bright", on the other hand, features both high-lightness colors and saturated colors—"bright yellow" can refer to the prototypical yellow, whereas "light yellow" cannot. Finally, "dull" denotes unsaturated colors in a variety of lightnesses.

Compositionality Our model generalizes to compositional descriptions not found in the training set. Fig. 3 visualizes the probability assigned to the

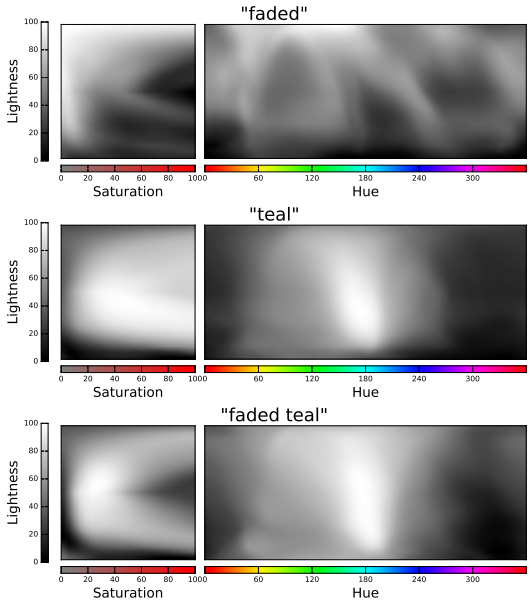


Figure 3: Conditional likelihood of "faded", "teal", and "faded teal". The two meaning components can be seen in the two cross-sections: "faded" denotes a low saturation value, and "teal" denotes hues near the center of the spectrum.

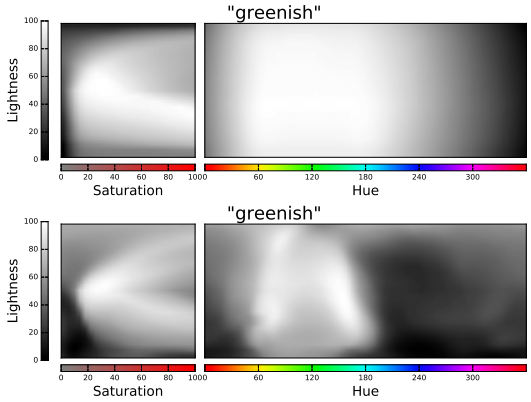


Figure 4: Conditional likelihood of "greenish" as a function of color. The distribution is bimodal, including greenish yellows and blues but not true greens. Top: LUX; bottom: our model.

novel utterance "faded teal", along with "faded" and "teal" individually. The meaning of "faded teal" is intersective: "faded" colors are lower in saturation, excluding the colors of the rainbow (the V on the right side of the left panel); and "teal" denotes colors with a hue near 180° (center of the right panel).

Non-convex denotations The Fourier feature transformation and the nonlinearities in the model allow it to capture a rich set of denotations. In particular, our model addresses the shortcoming identified by McMahan and Stone (2015) that their model cannot capture non-convex denotations. The description

Color	Top-1	Sample
(36, 86, 63)	"orange"	"ugly"
(177, 85, 26)	"teal"	"robin's"
(29, 45, 71)	"tan"	"reddish green"
(196, 27, 71)	"grey"	"baby royal"

Table 3: Error analysis: some color descriptions sampled from our model that are incorrect or incomplete.

"greenish" (Fig. 4) has such a denotation: "greenish" specifies a region of color space surrounding, but not including, true greens.

Error analysis Table 3 shows some examples of errors found in samples taken from the model. The main type of error the system makes is ungrammatical descriptions, particularly fragments lacking a basic color term (e.g., "robin's"). Rarer are grammatical but meaningless compositions ("reddish green") and false descriptions. When queried for its single most likely prediction, $\arg\max_d S(d \mid c)$, the result is nearly always an acceptable, "safe" description—manual inspection of 200 such top-1 predictions did not identify any errors.

5 Conclusion and future work

We presented a model for generating compositional color descriptions that is capable of producing novel descriptions not seen in training and significantly outperforms prior work at conditional language modeling.[3] One natural extension is the use of character-level sequence modeling to capture complex morphology (e.g., "-ish" in "greenish"). Kawakami et al. (2016) build character-level models for predicting colors given descriptions in addition to describing colors. Their model uses a *Lab*-space color representation and uses the color to initialize the LSTM instead of feeding it in at each time step; they also focus on visualizing point predictions of their description-to-color model, whereas we examine the full distributions implied by our color-to-description model.

Another extension we plan to investigate is modeling of context, to capture how people describe colors differently to contrast them with other colors via pragmatic reasoning (DeVault and Stone, 2007; Golland et al., 2010; Monroe and Potts, 2015).

[3]We release our code at `https://github.com/stanfordnlp/color-describer`.

Acknowledgments

We thank Jiwei Li, Jian Zhang, Anusha Balakrishnan, and Daniel Ritchie for valuable advice and discussions. This research was supported in part by the Stanford Data Science Initiative, NSF BCS 1456077, and NSF IIS 1159679.

References

Hirotugu Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, et al. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Brent Berlin and Paul Kay. 1991. *Basic color terms: Their universality and evolution*. University of California Press.

David DeVault and Matthew Stone. 2007. Managing ambiguities across utterances in dialogue. In Ron Artstein and Laure Vieu, editors, *Proceedings of DECALOG 2007: Workshop on the Semantics and Pragmatics of Dialogue*.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, et al. 2015. Lasagne: First release.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *EMNLP*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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

Kazuya Kawakami, Chris Dyer, Bryan Routledge, and Noah A. Smith. 2016. Character sequence models for colorful words. In *EMNLP*.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, et al. 2011. Baby talk: Understanding and generating image descriptions. In *CVPR*.

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, et al. 2012. Midge: Generating image descriptions from computer vision detections. In *EACL*.

Will Monroe and Christopher Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of the 20th Amsterdam Colloquium*.

Randall Munroe. 2010. Color survey results. Online at http://blog.xkcd.com/2010/05/03/color-surveyresults.

Sebastian Padó, 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation*. `http://www.nlpado.de/~sebastian/software/sigf.shtml`.

Dengsheng Zhang and Guojun Lu. 2002. Shape-based image retrieval using generic Fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848.

A Decomposable Attention Model for Natural Language Inference

Ankur P. Parikh
Google
New York, NY

Oscar Täckström
Google
New York, NY

Dipanjan Das
Google
New York, NY

Jakob Uszkoreit
Google
Mountain View, CA

{aparikh,oscart,dipanjand,uszkoreit}@google.com

Abstract

We propose a simple neural architecture for natural language inference. Our approach uses attention to decompose the problem into subproblems that can be solved separately, thus making it trivially parallelizable. On the Stanford Natural Language Inference (SNLI) dataset, we obtain state-of-the-art results with almost an order of magnitude fewer parameters than previous work and without relying on any word-order information. Adding intra-sentence attention that takes a minimum amount of order into account yields further improvements.

1 Introduction

Natural language inference (NLI) refers to the problem of determining entailment and contradiction relationships between a premise and a hypothesis. NLI is a central problem in language understanding (Katz, 1972; Bos and Markert, 2005; van Benthem, 2008; MacCartney and Manning, 2009) and recently the large SNLI corpus of 570K sentence pairs was created for this task (Bowman et al., 2015). We present a new model for NLI and leverage this corpus for comparison with prior work.

A large body of work based on neural networks for text similarity tasks including NLI has been published in recent years (Hu et al., 2014; Rocktäschel et al., 2016; Wang and Jiang, 2016; Yin et al., 2016, *inter alia*). The dominating trend in these models is to build complex, deep text representation models, for example, with convolutional networks (LeCun et al., 1990, CNNs henceforth) or long short-term memory networks (Hochreiter and Schmidhuber, 1997, LSTMs henceforth) with the goal of deeper sentence comprehension. While these approaches have yielded impressive results, they are often computationally very expensive, and result in models having millions of parameters (excluding embeddings).

Here, we take a different approach, arguing that for natural language inference it can often suffice to simply align bits of local text substructure and then aggregate this information. For example, consider the following sentences:

- *Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.*
- *Bob is awake.*
- *It is sunny outside.*

The first sentence is complex in structure and it is challenging to construct a compact representation that expresses its entire meaning. However, it is fairly easy to conclude that the second sentence follows from the first one, by simply aligning *Bob* with *Bob* and *cannot sleep* with *awake* and recognizing that these are synonyms. Similarly, one can conclude that *It is sunny outside* contradicts the first sentence, by aligning *thunder and lightning* with *sunny* and recognizing that these are most likely incompatible.

We leverage this intuition to build a simpler and more lightweight approach to NLI within a neural framework; with considerably fewer parameters, our model outperforms more complex existing neural architectures. In contrast to existing approaches, our approach only relies on alignment and is fully computationally decomposable with respect to the input text. An overview of our approach is given in Figure 1. Given two sentences, where each word is repre-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2249–2255,
Austin, Texas, November 1-5, 2016.

sented by an embedding vector, we first create a soft alignment matrix using neural attention (Bahdanau et al., 2015). We then use the (soft) alignment to decompose the task into subproblems that are solved separately. Finally, the results of these subproblems are merged to produce the final classification. In addition, we optionally apply intra-sentence attention (Cheng et al., 2016) to endow the model with a richer encoding of substructures prior to the alignment step.

Asymptotically our approach does the same total work as a vanilla LSTM encoder, while being trivially parallelizable across sentence length, which can allow for considerable speedups in low-latency settings. Empirical results on the SNLI corpus show that our approach achieves state-of-the-art results, while using almost an order of magnitude fewer parameters compared to complex LSTM-based approaches.

2 Related Work

Our method is motivated by the central role played by alignment in machine translation (Koehn, 2009) and previous approaches to sentence similarity modeling (Haghighi et al., 2005; Das and Smith, 2009; Chang et al., 2010; Fader et al., 2013), natural language inference (Marsi and Krahmer, 2005; MacCartney et al., 2006; Hickl and Bensley, 2007; MacCartney et al., 2008), and semantic parsing (Andreas et al., 2013). The neural counterpart to alignment, *attention* (Bahdanau et al., 2015), which is a key part of our approach, was originally proposed and has been predominantly used in conjunction with LSTMs (Rocktäschel et al., 2016; Wang and Jiang, 2016) and to a lesser extent with CNNs (Yin et al., 2016). In contrast, our use of attention is purely based on word embeddings and our method essentially consists of feed-forward networks that operate largely independently of word order.

3 Approach

Let $\mathbf{a} = (a_1, \ldots, a_{\ell_a})$ and $\mathbf{b} = (b_1, \ldots, b_{\ell_b})$ be the two input sentences of length ℓ_a and ℓ_b, respectively. We assume that each a_i, $b_j \in \mathbb{R}^d$ is a word embedding vector of dimension d and that each sentence is prepended with a "NULL" token. Our training data comes in the form of labeled pairs $\{\mathbf{a}^{(n)}, \mathbf{b}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, where $\mathbf{y}^{(n)} = (y_1^{(n)}, \ldots, y_C^{(n)})$ is an indicator vector encoding the label and C is the number of output classes. At test time, we receive a pair of sentences $(\mathbf{a}, \mathbf{b})$ and our goal is to predict the correct label $\mathbf{y}$.

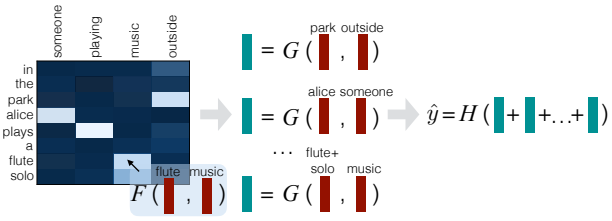


Figure 1: Pictoral overview of the approach, showing the *Attend* (left), *Compare* (center) and *Aggregate* (right) steps.

Input representation. Let $\bar{\mathbf{a}} = (\bar{a}_1, \ldots, \bar{a}_{\ell_a})$ and $\bar{\mathbf{b}} = (\bar{b}_1, \ldots, \bar{b}_{\ell_b})$ denote the input representation of each fragment that is fed to subsequent steps of the algorithm. The vanilla version of our model simply defines $\bar{\mathbf{a}} := \mathbf{a}$ and $\bar{\mathbf{b}} := \mathbf{b}$. With this input representation, our model does not make use of word order. However, we discuss an extension using intra-sentence attention in Section 3.4 that uses a minimal amount of sequence information.

The core model consists of the following three components (see Figure 1), which are trained jointly:

Attend. First, soft-align the elements of $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ using a variant of neural attention (Bahdanau et al., 2015) and decompose the problem into the comparison of aligned subphrases.

Compare. Second, separately compare each aligned subphrase to produce a set of vectors $\{\mathbf{v}_{1,i}\}_{i=1}^{\ell_a}$ for $\mathbf{a}$ and $\{\mathbf{v}_{2,j}\}_{j=1}^{\ell_b}$ for $\mathbf{b}$. Each $\mathbf{v}_{1,i}$ is a nonlinear combination of a_i and its (softly) aligned subphrase in $\mathbf{b}$ (and analogously for $\mathbf{v}_{2,j}$).

Aggregate. Finally, aggregate the sets $\{\mathbf{v}_{1,i}\}_{i=1}^{\ell_a}$ and $\{\mathbf{v}_{2,j}\}_{j=1}^{\ell_b}$ from the previous step and use the result to predict the label $\hat{y}$.

3.1 Attend

We first obtain unnormalized attention weights e_{ij}, computed by a function F', which decomposes as:

$$e_{ij} := F'(\bar{a}_i, \bar{b}_j) := F(\bar{a}_i)^T F(\bar{b}_j)\,. \qquad (1)$$

This decomposition avoids the quadratic complexity that would be associated with separately applying F' $\ell_a \times \ell_b$ times. Instead, only $\ell_a + \ell_b$ applications of F are needed. We take F to be a feed-forward neural network with ReLU activations (Glorot et al., 2011).

These attention weights are normalized as follows:

$$\beta_i := \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} \bar{b}_j \,,$$
$$\alpha_j := \sum_{i=1}^{\ell_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a} \exp(e_{kj})} \bar{a}_i \,. \quad (2)$$

Here β_i is the subphrase in $\bar{\mathbf{b}}$ that is (softly) aligned to $\bar{a}_i$ and vice versa for α_j.

3.2 Compare

Next, we separately compare the aligned phrases $\{(\bar{a}_i, \beta_i)\}_{i=1}^{\ell_a}$ and $\{(\bar{b}_j, \alpha_j)\}_{j=1}^{\ell_b}$ using a function G, which in this work is again a feed-forward network:

$$\mathbf{v}_{1,i} := G([\bar{a}_i, \beta_i]) \quad \forall i \in [1, \ldots, \ell_a] \,,$$
$$\mathbf{v}_{2,j} := G([\bar{b}_j, \alpha_j]) \quad \forall j \in [1, \ldots, \ell_b] \,. \quad (3)$$

where the brackets $[\cdot, \cdot]$ denote concatenation. Note that since there are only a linear number of terms in this case, we do not need to apply a decomposition as was done in the previous step. Thus G can jointly take into account both $\bar{a}_i$, and β_i.

3.3 Aggregate

We now have two sets of comparison vectors $\{\mathbf{v}_{1,i}\}_{i=1}^{\ell_a}$ and $\{\mathbf{v}_{2,j}\}_{j=1}^{\ell_b}$. We first aggregate over each set by summation:

$$\mathbf{v}_1 = \sum_{i=1}^{\ell_a} \mathbf{v}_{1,i} \quad , \quad \mathbf{v}_2 = \sum_{j=1}^{\ell_b} \mathbf{v}_{2,j} \,. \quad (4)$$

and feed the result through a final classifier H, that is a feed forward network followed by a linear layer:

$$\hat{\mathbf{y}} = H([\mathbf{v}_1, \mathbf{v}_2]) \,, \quad (5)$$

where $\hat{\mathbf{y}} \in \mathbb{R}^C$ represents the predicted (unnormalized) scores for each class and consequently the predicted class is given by $\hat{y} = \text{argmax}_i \hat{\mathbf{y}}_i$.

For training, we use multi-class cross-entropy loss with dropout regularization (Srivastava et al., 2014):

$$L(\theta_F, \theta_G, \theta_H) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_c^{(n)} \log \frac{\exp(\hat{y}_c)}{\sum_{c'=1}^{C} \exp(\hat{y}_{c'})} \,.$$

Here $\theta_F, \theta_G, \theta_H$ denote the learnable parameters of the functions F, G and H, respectively.

3.4 Intra-Sentence Attention (Optional)

In the above model, the input representations are simple word embeddings. However, we can augment this input representation with *intra-sentence attention* to encode compositional relationships between words within each sentence, as proposed by Cheng et al. (2016). Similar to Eqs. 1 and 2, we define

$$f_{ij} := F_{\text{intra}}(a_i)^T F_{\text{intra}}(a_j) \,, \quad (6)$$

where F_{intra} is a feed-forward network. We then create the self-aligned phrases

$$a'_i := \sum_{j=1}^{\ell_a} \frac{\exp(f_{ij} + d_{i-j})}{\sum_{k=1}^{\ell_a} \exp(f_{ik} + d_{i-k})} a_j \,. \quad (7)$$

The distance-sensitive bias terms $d_{i-j} \in \mathbb{R}$ provides the model with a minimal amount of sequence information, while remaining parallelizable. These terms are bucketed such that all distances greater than 10 words share the same bias. The input representation for subsequent steps is then defined as $\bar{a}_i := [a_i, a'_i]$ and analogously $\bar{b}_i := [b_i, b'_i]$.

4 Computational Complexity

We now discuss the asymptotic complexity of our approach and how it offers a higher degree of parallelism than LSTM-based approaches. Recall that d denotes embedding dimension and ℓ means sentence length. For simplicity we assume that all hidden dimensions are d and that the complexity of matrix($d \times d$)-vector($d \times 1$) multiplication is $O(d^2)$.

A key assumption of our analysis is that $\ell < d$, which we believe is reasonable and is true of the SNLI dataset (Bowman et al., 2015) where $\ell < 80$, whereas recent LSTM-based approaches have used $d \geq 300$. This assumption allows us to bound the complexity of computing the ℓ^2 attention weights.

Complexity of LSTMs. The complexity of an LSTM cell is $O(d^2)$, resulting in a complexity of $O(\ell d^2)$ to encode the sentence. Adding attention as in Rocktäschel et al. (2016) increases this complexity to $O(\ell d^2 + \ell^2 d)$.

Complexity of our Approach. Application of a feed-forward network requires $O(d^2)$ steps. Thus, the **Compare** and **Aggregate** steps have complexity $O(\ell d^2)$ and $O(d^2)$ respectively. For the **Attend** step,

Method	Train Acc	Test Acc	#Parameters
Lexicalized Classifier (Bowman et al., 2015)	99.7	78.2	–
300D LSTM RNN encoders (Bowman et al., 2016)	83.9	80.6	3.0M
1024D pretrained GRU encoders (Vendrov et al., 2015)	98.8	81.4	15.0M
300D tree-based CNN encoders (Mou et al., 2015)	83.3	82.1	3.5M
300D SPINN-PI encoders (Bowman et al., 2016)	89.2	83.2	3.7M
100D LSTM with attention (Rocktäschel et al., 2016)	85.3	83.5	252K
300D mLSTM (Wang and Jiang, 2016)	92.0	86.1	1.9M
450D LSTMN with deep attention fusion (Cheng et al., 2016)	88.5	86.3	3.4M
Our approach (vanilla)	89.5	86.3	382K
Our approach with intra-sentence attention	90.5	**86.8**	582K

Table 1: Train/test accuracies on the SNLI dataset and number of parameters (excluding embeddings) for each approach.

Method	**N**	**E**	**C**
Bowman et al. (2016)	80.6	88.2	85.5
Wang and Jiang (2016)	81.6	91.6	87.4
Our approach (vanilla)	83.6	91.3	85.8
Our approach w/ intra att.	83.7	92.1	86.7

Table 2: Breakdown of accuracy with respect to classes on SNLI development set. **N**=neutral, **E**=entailment, **C**=contradiction.

F is evaluated $O(\ell)$ times, giving a complexity of $O(\ell d^2)$. Each attention weight e_{ij} requires one dot product, resulting in a complexity of $O(\ell^2 d)$.

Thus the total complexity of the model is $O(\ell d^2 + \ell^2 d)$, which is equal to that of an LSTM with attention. However, note that with the assumption that $\ell < d$, this becomes $O(\ell d^2)$ which is the same complexity as a regular LSTM. Moreover, unlike the LSTM, our approach has the advantage of being parallelizable over ℓ, which can be useful at test time.

5 Experiments

We evaluate our approach on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). Given a sentences pair $(\mathbf{a}, \mathbf{b})$, the task is to predict whether $\mathbf{b}$ is *entailed* by $\mathbf{a}$, $\mathbf{b}$ *contradicts* $\mathbf{a}$, or whether their relationship is *neutral*.

5.1 Implementation Details

The method was implemented in TensorFlow (Abadi et al., 2015).

Data preprocessing: Following Bowman et al. (2015), we remove examples labeled "–" (no gold label) from the dataset, which leaves 549,367 pairs for training, 9,842 for development, and 9,824 for testing. We use the tokenized sentences from the non-binary parse provided in the dataset and prepend each sentence with a "NULL" token. During training, each sentence was padded up to the maximum length of the batch for efficient training (the padding was explicitly masked out so as not to affect the objective/gradients). For efficient batching in TensorFlow, we semi-sorted the training data to first contain examples where both sentences had length less than 20, followed by those with length less than 50, and then the rest. This ensured that most training batches contained examples of similar length.

Embeddings: We use 300 dimensional GloVe embeddings (Pennington et al., 2014) to represent words. Each embedding vector was normalized to have ℓ_2 norm of 1 and projected down to 200 dimensions, a number determined via hyperparameter tuning. Out-of-vocabulary (OOV) words are hashed to one of 100 random embeddings each initialized to mean 0 and standard deviation 1. All embeddings remain fixed during training, but the projection matrix is trained. All other parameter weights (hidden layers etc.) were initialized from random Gaussians with mean 0 and standard deviation 0.01.

Each hyperparameter setting was run on a single machine with 10 asynchronous gradient-update threads, using Adagrad (Duchi et al., 2011) for optimization with the default initial accumulator value of 0.1. Dropout regularization (Srivastava et al., 2014) was used for all ReLU layers, but not for the final linear layer. We additionally tuned the following hyperparameters and present their chosen values in

ID	Sentence 1	Sentence 2	DA (vanilla)	DA (intra att.)	SPINN-PI	mLSTM	Gold
A	Two kids are standing in the ocean hugging each other.	Two kids enjoy their day at the beach.	**N**	**N**	E	E	**N**
B	A dancer in costumer performs on stage while a man watches.	the man is captivated	**N**	**N**	E	E	**N**
C	They are sitting on the edge of a fountain	The fountain is splashing the persons seated.	**N**	**N**	C	C	**N**
D	Two dogs play with tennis ball in field.	Dogs are watching a tennis match.	N	**C**	**C**	**C**	**C**
E	Two kids begin to make a snowman on a sunny winter day.	Two penguins making a snowman.	N	**C**	**C**	**C**	**C**
F	The horses pull the carriage, holding people and a dog, through the rain.	Horses ride in a carriage pulled by a dog.	E	E	**C**	**C**	**C**
G	A woman closes her eyes as she plays her cello.	The woman has her eyes open.	E	E	E	E	**C**
H	Two women having drinks and smoking cigarettes at the bar.	Three women are at a bar.	E	E	E	E	**C**
I	A band playing with fans watching.	A band watches the fans play	E	E	E	E	**C**

Table 3: Example wins and losses compared to other approaches. DA (Decomposable Attention) refers to our approach while SPINN-PI and mLSTM are previously developed methods (see Table 1).

parentheses: network size (2-layers, each with 200 neurons), batch size (4), [1] dropout ratio (0.2) and learning rate (0.05–vanilla, 0.025–intra-attention). All settings were run for 50 million steps (each step indicates one batch) but model parameters were saved frequently as training progressed and we chose the model that did best on the development set.

5.2 Results

Results in terms of 3-class accuracy are shown in Table 1. Our vanilla approach achieves state-of-the-art results with almost an order of magnitude fewer parameters than the LSTMN of Cheng et al. (2016). Adding intra-sentence attention gives a considerable improvement of 0.5 percentage points over the existing state of the art. Table 2 gives a breakdown of accuracy on the development set showing that most of our gains stem from *neutral*, while most losses come from *contradiction* pairs.

Table 3 shows some wins and losses. Examples A-C are cases where both variants of our approach are correct while both SPINN-PI (Bowman et al., 2016) and the mLSTM (Wang and Jiang, 2016) are incorrect. In the first two cases, both sentences contain phrases that are either identical or highly lexically related (e.g. "Two kids" and "ocean / beach") and our approach correctly favors neutral in these cases. In Example C, it is possible that relying on word-order may confuse SPINN-PI and the mLSTM due to how "fountain" is the object of a preposition in the first sentence but the subject of the second.

The second set of examples (D-F) are cases where our vanilla approach is incorrect but mLSTM and SPINN-PI are correct. Example F requires sequential information and neither variant of our approach can predict the correct class. Examples D-E are interesting however, since they don't require word order information, yet intra-attention seems to help. We suspect this may be because the word embeddings are not fine-grained enough for the algorithm to conclude that "play/watch" is a contradiction, but intra-attention, by adding an extra layer of composition/nonlinearity to incorporate context, compensates for this.

Finally, Examples G-I are cases that all methods get wrong. The first is actually representative of many examples in this category where there is one critical word that separates the two sentences (close vs open in this case) and goes unnoticed by the algorithms. Examples H requires inference about numbers and Example I needs sequence information.

6 Conclusion

We presented a simple attention-based approach to natural language inference that is trivially parallelizable. The approach outperforms considerably more complex neural methods aiming for text understanding. Our results suggest that, at least for this task, pairwise comparisons are relatively more important than global sentence-level representations.

Acknowledgements

We thank Slav Petrov, Tom Kwiatkowski, Yoon Kim, Erick Fonseca, Mark Neumann for useful discussion and Sam Bowman and Shuohang Wang for providing us their model outputs for error analysis.

[1] 16 or 32 also work well and are a bit more stable.

References

[Abadi et al.2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow. org*.

[Andreas et al.2013] Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of ACL*.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

[Bos and Markert2005] Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of EMNLP*.

[Bowman et al.2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

[Bowman et al.2016] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of ACL*.

[Chang et al.2010] Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of HLT-NAACL*.

[Cheng et al.2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of EMNLP*.

[Das and Smith2009] Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL-IJCNLP*.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Fader et al.2013] Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of ACL*.

[Glorot et al.2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of AISTATS*.

[Haghighi et al.2005] Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of HLT-NAACL*.

[Hickl and Bensley2007] Andrew Hickl and Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hu et al.2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in NIPS*.

[Katz1972] Jerrold J. Katz. 1972. *Semantic theory*. Harper & Row.

[Koehn2009] Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

[LeCun et al.1990] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in NIPS*.

[MacCartney and Manning2009] Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the IWCS*.

[MacCartney et al.2006] Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*.

[MacCartney et al.2008] Bill MacCartney, Michel Galley, and Christopher D Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

[Marsi and Krahmer2005] Erwin Marsi and Emiel Krahmer. 2005. Classification of semantic relations by humans and machines. In *Proceedings of the ACL workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

[Mou et al.2015] Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of ACL (short papers)*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

[Rocktäschel et al.2016] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[van Benthem2008] Johan van Benthem. 2008. *A brief history of natural logic*. College Publications.

[Vendrov et al.2015] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. In *Proceedings of ICLR*.

[Wang and Jiang2016] Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of NAACL*.

[Yin et al.2016] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. In *Transactions of the Association of Computational Linguistics*.

Deep Reinforcement Learning for Mention-Ranking Coreference Models

Kevin Clark
Computer Science Department
Stanford University
kevclark@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

Abstract

Coreference resolution systems are typically trained with heuristic loss functions that require careful tuning. In this paper we instead apply reinforcement learning to directly optimize a neural mention-ranking model for coreference evaluation metrics. We experiment with two approaches: the REINFORCE policy gradient algorithm and a reward-rescaled max-margin objective. We find the latter to be more effective, resulting in a significant improvement over the current state-of-the-art on the English and Chinese portions of the CoNLL 2012 Shared Task.

1 Introduction

Coreference resolution systems typically operate by making sequences of local decisions (e.g., adding a coreference link between two mentions). However, most measures of coreference resolution performance do not decompose over local decisions, which means the utility of a particular decision is not known until all other decisions have been made.

Due to this difficulty, coreference systems are usually trained with loss functions that heuristically define the goodness of a particular coreference decision. These losses contain hyperparameters that are carefully selected to ensure the model performs well according to coreference evaluation metrics. This complicates training, especially across different languages and datasets where systems may work best with different settings of the hyperparameters.

To address this, we explore using two variants of reinforcement learning to directly optimize a coreference system for coreference evaluation metrics. In particular, we modify the max-margin coreference objective proposed by Wiseman et al. (2015) by incorporating the reward associated with each coreference decision into the loss's slack rescaling. We also test the REINFORCE policy gradient algorithm (Williams, 1992).

Our model is a neural mention-ranking model. Mention-ranking models score pairs of mentions for their likelihood of coreference rather than comparing partial coreference clusters. Hence they operate in a simple setting where coreference decisions are made independently. Although they are less expressive than entity-centric approaches to coreference (e.g., Haghighi and Klein, 2010), mention-ranking models are fast, scalable, and simple to train, causing them to be the dominant approach to coreference in recent years (Durrett and Klein, 2013; Wiseman et al., 2015). Having independent actions is particularly useful when applying reinforcement learning because it means a particular action's effect on the final reward can be computed efficiently.

We evaluate the models on the English and Chinese portions of the CoNLL 2012 Shared Task. The REINFORCE algorithm is competitive with a heuristic loss function while the reward-rescaled objective significantly outperforms both[1]. We attribute this to reward rescaling being well suited for a ranking task due to its max-margin loss as well as benefiting from directly optimizing for coreference metrics. Error analysis shows that using the reward-rescaling loss results in a similar number of mistakes as the heuristic loss, but the mistakes tend to be less severe.

[1]Code and trained models are available at https://github.com/clarkkev/deep-coref.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2256–2262,
Austin, Texas, November 1-5, 2016.

2 Neural Mention-Ranking Model

We use the neural mention-ranking model described in Clark and Manning (2016), which we briefly go over in this section. Given a mention m and candidate antecedent c, the mention-ranking model produces a score for the pair $s(c, m)$ indicating their compatibility for coreference with a feedforward neural network. The candidate antecedent may be any mention that occurs before m in the document or NA, indicating that m has no antecedent.

Input Layer. For each mention, the model extracts various words (e.g., the mention's head word) and groups of words (e.g., all words in the mention's sentence) that are fed into the neural network. Each word is represented by a vector $\boldsymbol{w}_i \in \mathbb{R}^{d_w}$. Each group of words is represented by the average of the vectors of each word in the group. In addition to the embeddings, a small number of additional features are used, including distance, string matching, and speaker identification features. See Clark and Manning (2016) for the full set of features and an ablation study.

These features are concatenated to produce an I-dimensional vector $\boldsymbol{h}_0$, the input to the neural network. If $c = \text{NA}$, features defined over pairs of mentions are not included. For this case, we train a separate network with an identical architecture to the pair network except for the input layer to produce anaphoricity scores.

Hidden Layers. The input gets passed through three hidden layers of rectified linear (ReLU) units (Nair and Hinton, 2010). Each unit in a hidden layer is fully connected to the previous layer:

$$\boldsymbol{h}_i(c, m) = \max(0, \boldsymbol{W}_i \boldsymbol{h}_{i-1}(c, m) + \boldsymbol{b}_i)$$

where $\boldsymbol{W}_1$ is a $M_1 \times I$ weight matrix, $\boldsymbol{W}_2$ is a $M_2 \times M_1$ matrix, and $\boldsymbol{W}_3$ is a $M_3 \times M_2$ matrix.

Scoring Layer. The final layer is a fully connected layer of size 1:

$$s(c, m) = \boldsymbol{W}_4 \boldsymbol{h}_3(c, m) + b_4$$

where $\boldsymbol{W}_4$ is a $1 \times M_3$ weight matrix. At test time, the mention-ranking model links each mention with its highest scoring candidate antecedent.

3 Learning Algorithms

Mention-ranking models are typically trained with heuristic loss functions that are tuned via hyperparameters. These hyperparameters are usually given as costs for different error types, which are used to bias the coreference system towards making more or fewer coreference links. In this section we first describe a heuristic loss function incorporating this idea from Wiseman et al. (2015). We then propose new training procedures based on reinforcement learning that instead directly optimize for coreference evaluation metrics.

3.1 Heuristic Max-Margin Objective

The heuristic loss from Wiseman et al. is governed by the following error types, which were first proposed by Durrett et al. (2013).

Suppose the training set consists of N mentions $m_1, m_2, ..., m_N$. Let $\mathcal{C}(m_i)$ denote the set of candidate antecedents of a mention m_i (i.e., mentions preceding m_i and NA) and $\mathcal{T}(m_i)$ denote the set of true antecedents of m_i (i.e., mentions preceding m_i that are coreferent with it or $\{\text{NA}\}$ if m_i has no antecedent). Then we define the following costs for linking m_i to a candidate antecedent $c \in \mathcal{C}(m_i)$:

$$\Delta_h(c, m_i) = \begin{cases} \alpha_{\text{FN}} & \text{if } c = \text{NA} \ \wedge \mathcal{T}(m_i) \neq \{\text{NA}\} \\ \alpha_{\text{FA}} & \text{if } c \neq \text{NA} \ \wedge \mathcal{T}(m_i) = \{\text{NA}\} \\ \alpha_{\text{WL}} & \text{if } c \neq \text{NA} \ \wedge a \notin \mathcal{T}(m_i) \\ 0 & \text{if } a \in \mathcal{T}(m_i) \end{cases}$$

for "false new," "false anaphor," "wrong link", and correct coreference decisions.

The heuristic loss is a slack-rescaled max-margin objective parameterized by these error costs. Let $\hat{t}_i$ be the highest scoring true antecedent of m_i:

$$\hat{t}_i = \underset{c \in \mathcal{C}(m_i) \wedge \Delta_h(c, m_i) = 0}{\text{argmax}} s(c, m_i)$$

Then the heuristic loss is given as

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \max_{c \in \mathcal{C}(m_i)} \Delta_h(c, m_i)(1 + s(c, m_i) - s(\hat{t}_i, m_i))$$

Finding Effective Error Penalties. We fix $\alpha_{\text{WL}} = 1.0$ and search for α_{FA} and α_{FN} out of $\{0.1, 0.2, ..., 1.5\}$ with a variant of grid search. Each new trial uses the unexplored set of hyperparame-

ters that has the closest Manhattan distance to the best setting found so far on the dev set. The search is halted when all immediate neighbors (within 0.1 distance) of the best setting have been explored. We found $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$ to be best for English and $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.5, 1.0)$ to be best for Chinese on the CoNLL 2012 data.

3.2 Reinforcement Learning

Finding the best hyperparameter settings for the heuristic loss requires training many variants of the model, and at best results in an objective that is correlated with coreference evaluation metrics. To address this, we pose mention ranking in the reinforcement learning framework (Sutton and Barto, 1998) and propose methods that directly optimize the model for coreference metrics.

We can view the mention-ranking model as an *agent* taking a series of *actions* $a_{1:T} = a_1, a_2, ..., a_T$, where T is the number of mentions in the current document. Each action a_i links the ith mention in the document m_i to a candidate antecedent. Formally, we denote the set of actions available for the ith mention as $\mathcal{A}_i = \{(c, m_i) : c \in \mathcal{C}(m_i)\}$, where an action (c, m) adds a coreference link between mentions m and c. The mention-ranking model assigns each action the score $s(c, m)$ and takes the highest-scoring action at each step.

Once the agent has executed a sequence of actions, it observes a *reward* $R(a_{1:T})$, which can be any function. We use the B^3 coreference metric for this reward (Bagga and Baldwin, 1998). Although our system evaluation also includes the MUC (Vilain et al., 1995) and CEAF_{ϕ_4} (Luo, 2005) metrics, we do not incorporate them into the loss because MUC has the flaw of treating all errors equally and CEAF_{ϕ_4} is slow to compute.

Reward Rescaling. Crucially, the actions taken by a mention-ranking model are independent. This means it is possible to change any action a_i to a different one $a'_i \in \mathcal{A}_i$ and see what reward the model would have gotten by taking that action instead: $R(a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_T)$. We use this idea to improve the slack-rescaling parameter Δ in the max-margin loss $\mathcal{L}(\theta)$. Instead of setting its value based on the error type, we compute exactly how much each action hurts the final reward:

$$\Delta_r(c, m_i) = -R(a_1, ..., (c, m_i), ..., a_T) + \max_{a'_i \in \mathcal{A}_i} R(a_1, ..., a'_i, ..., a_T)$$

where $a_{1:T}$ is the highest scoring sequence of actions according to the model's current parameters. Otherwise the model is trained in the same way as with the heuristic loss.

The REINFORCE Algorithm. We also explore using the REINFORCE policy gradient algorithm (Williams, 1992). We can define a probability distribution over actions using the mention-ranking model's scoring function as follows:

$$p_\theta(a) \propto e^{s(c,m)}$$

for any action $a = (c, m)$. The REINFORCE algorithm seeks to maximize the expected reward

$$J(\theta) = \mathbb{E}_{[a_{1:T} \sim p_\theta]} R(a_{1:T})$$

It does this through gradient ascent. Computing the full gradient is prohibitive because of the expectation over all possible action sequences, which is exponential in the length of the sequence. Instead, it gets an unbiased estimate of the gradient by sampling a sequence of actions $a_{1:T}$ according to p_θ and computing the gradient only over the sample.

We take advantage of the independence of actions by using the following gradient estimate, which has lower variance than the standard REINFORCE gradient estimate.

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{T} \sum_{a'_i \in \mathcal{A}_i} [\nabla_\theta p_\theta(a'_i)](R(a_1, ..., a'_i, ..., a_T) - b_i)$$

where b_i is a baseline used to reduce the variance, which we set to $\mathbb{E}_{a'_i \in \mathcal{A}_i \sim p_\theta} R(a_1, ..., a'_i, ..., a_T)$.

4 Experiments and Results

We run experiments on the English and Chinese portions of the CoNLL 2012 Shared Task data (Pradhan et al., 2012) and evaluate with the MUC, B^3, and CEAF_{ϕ_4} metrics. Our experiments were run using predicted mentions from Stanford's rule-based coreference system (Raghunathan et al., 2010).

We follow the training methodology from Clark and Manning (2016): hidden layers of sizes M_1 = 1000, M_2 = M_3 = 500, the RMSprop optimizer

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Avg. F_1
CoNLL 2012 English Test Data										
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
Clark & Manning (2016)	79.91	69.30	74.23	71.01	56.53	62.95	63.84	54.33	58.70	65.29
Heuristic Loss	79.63	70.25	74.65	69.21	57.87	63.03	63.62	53.97	58.40	65.36
REINFORCE	80.08	69.61	74.48	70.70	56.96	63.09	63.59	54.46	58.67	65.41
Reward Rescaling	79.19	70.44	74.56	69.93	57.99	63.40	63.46	55.52	59.23	65.73
CoNLL 2012 Chinese Test Data										
Björkelund & Kuhn (2014)	69.39	62.57	65.80	61.64	53.87	57.49	59.33	54.65	56.89	60.06
Clark & Manning (2016)	74.45	64.73	69.25	68.71	55.54	61.43	63.14	57.48	60.18	63.62
Heuristic Loss	72.20	66.51	69.24	64.71	58.16	61.26	61.98	58.41	60.14	63.54
REINFORCE	74.05	65.38	69.44	67.52	56.43	61.48	62.38	57.77	59.98	63.64
Reward Rescaling	73.64	65.62	69.40	67.48	56.94	61.76	62.46	58.60	60.47	63.88

Table 1: Comparison of the methods together with other state-of-the-art approaches on the test sets.

(Hinton and Tieleman, 2012), dropout (Hinton et al., 2012) with a rate of 0.5, and pretraining with the all pairs classification and top pairs classification tasks. However, we improve on the previous system by using using better mention detection, more effective hyperparameters, and more epochs of training.

4.1 Results

We compare the heuristic loss, REINFORCE, and reward rescaling approaches on both datasets. We find that REINFORCE does slightly better than the heuristic loss, but reward rescaling performs significantly better than both across both languages.

We attribute the modest improvement of REINFORCE to it being poorly suited for a ranking task. During training it optimizes the model's performance in expectation, but at test-time it takes the most probable sequence of actions. This mismatch occurs even at the level of an individual decision: the model only links the current mention to a single antecedent, but is trained to assign high probability to all correct antecedents. We believe the benefit of REINFORCE being guided by coreference evaluation metrics is offset by this disadvantage, which does not occur in the max-margin approaches. The reward-rescaled max-margin loss combines the best of both worlds, resulting in superior performance.

4.2 The Benefits of Reinforcement Learning

In this section we examine the reward-based cost function Δ_r and perform error analysis to determine how reward rescaling improves the mention-ranking model's accuracy.

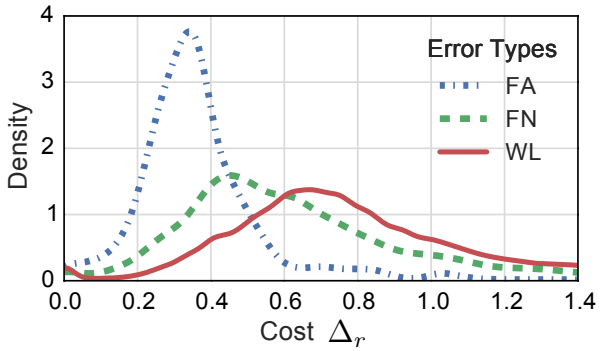


Figure 1: Density plot of the costs Δ_r associated with different error types on the English CoNLL 2012 test set.

Comparison with Heuristic Costs. We compare the reward-based cost function Δ_r with the error types used in the heuristic loss. For English, the average value of Δ_r is 0.79 for FN errors and 0.38 for FA errors when the costs are scaled so the average value of a WL error is 1.0. These are very close to the hyperparameter values $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$ found by grid search. However, there is a high variance in costs for each error type, suggesting that using a fixed penalty for each type as in the heuristic loss is insufficient (see Figure 1).

Avoiding Costly Mistakes. Embedding the costs of actions into the loss function causes the reward-rescaling model to prioritize getting the more important coreference decisions (i.e., the ones with the biggest impact on the final score) correct. As a

Class of Mentions	Average Cost $\overline{\Delta}_r$			# Heuristic Loss Errors			# Reward Rescaling Errors		
	FN	FA	WL	FN	FA	WL	FN	FA	WL
Proper nouns	0.90	0.38	1.02	403	597	221	334	660	233
Pronouns in phone conversations	0.86	0.39	1.21	82	85	81	90	78	67

Table 3: Examples of classes of mention on which the reward-rescaling loss significantly improves upon the heuristic loss due to its reward-based cost function. Reported numbers are from the English CoNLL 2012 test set.

Model	FN	FA	WL
Heuristic Loss	1719	1956	1258
Reward Rescaling	1725	1994	1247

Table 2: Number of "false new," "false anaphoric," and "wrong link" errors produced by the models on the English CoNLL 2012 test set.

result, it makes fewer costly mistakes at test time. Costly mistakes often involve large clusters of mentions: incorrectly combining two coreference clusters of size ten is much worse than incorrectly combining two clusters of size one. However, the cost of an action also depends on other factors like the number of errors already present in the clusters and the utilities of the other available actions.

Table 2 shows the breakdown of errors made by the heuristic and reward-rescaling models on the test set. The reward-rescaling model makes slightly more errors, meaning its improvement in performance must come from its errors being less severe.

Example Improvements. Table 3 shows two classes of mentions where the reward-rescaling loss particularly improves over the heuristic loss.

Proper nouns have a higher average cost for "false new" errors (0.90) than other mentions types (0.77). This is perhaps because proper nouns are important for connecting clusters of mentions far apart in a document, so incorrectly linking a proper noun to NA could result in a large decrease in recall. Because it more heavily weights these high-cost errors during training, the reward-rescaling model makes fewer "false new" errors for proper nouns than the heuristic loss. Although there is an increase in other kinds of errors as a result, most of these are low-cost "false anaphoric" errors.

The pronouns in the "telephone conversation" genre often group into extremely large coreference clusters, which means a "wrong link" error can have a very large negative effect on the score. This is reflected in its high average cost of 1.21. After prioritizing these examples during training, the reward-rescaling model creates significantly fewer wrong links than the heuristic loss, which is trained using a fixed cost of 1.0 for all wrong links.

5 Related Work

Mention-ranking models have been widely used for coreference resolution (Denis and Baldridge, 2007; Rahman and Ng, 2009; Durrett and Klein, 2013). These models are typically trained with heuristic loss functions that assign costs to different error types, as in the heuristic loss we describe in Section 3.1 (Fernandes et al., 2012; Durrett et al., 2013; Björkelund and Kuhn, 2014; Wiseman et al., 2015; Martschat and Strube, 2015; Wiseman et al., 2016).

To the best of our knowledge reinforcement learning has not been applied to coreference resolution before. However, imitation learning algorithms such as SEARN (Daumé III et al., 2009) have been used to train coreference resolvers (Daumé III, 2006; Ma et al., 2014; Clark and Manning, 2015). These algorithms also directly optimize for coreference evaluation metrics, but they require an expert policy to learn from instead of relying on rewards alone.

6 Conclusion

We propose using reinforcement learning to directly optimize mention-ranking models for coreference evaluation metrics, obviating the need for hyperparameters that must be carefully selected for each particular language, dataset, and evaluation metric. Our reward-rescaling approach also increases the model's accuracy, resulting in significant gains over the current state-of-the-art.

Acknowledgments

We thank Kelvin Guu, William Hamilton, Will Monroe, and the anonymous reviewers for their thoughtful comments and suggestions. This work was supported by NSF Award IIS-1514268.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Association of Computational Linguistics (ACL)*.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution with entity-level distributed representations. In *Association for Computational Linguistics (ACL)*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Hal Daumé III. 2006. *Practical structured learning techniques for natural language processing*. Ph.D. thesis, University of Southern California, Los Angeles, CA.

Pascal Denis and Jason Baldridge. 2007. A ranking approach to pronoun resolution. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1588–1593.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1971–1982.

Greg Durrett, David Leo Wright Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Association for Computational Linguistics (ACL)*, pages 114–124.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 41–48.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 385–393.

Geoffrey Hinton and Tijmen Tieleman. 2012. Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning, 4*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32.

Chao Ma, Janardhan Rao Doppa, J Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich, and Prasad Tadepalli. 2014. Prune-and-score: Learning for greedy coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics (TACL)*, 3:405–418.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 1–40.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–977.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT Press.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Sam Wiseman, Alexander M Rush, Stuart M Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 92–100.

Sam Wiseman, Alexander M Rush, Stuart M Shieber, and Jason Weston. 2016. Learning global features

for coreference resolution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*.

A Stacking Gated Neural Architecture for Implicit Discourse Relation Classification

Lianhui Qin[1,2], **Zhisong Zhang**[1,2], **Hai Zhao**[1,2,*]
[1]Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{qinlianhui, zzs2011}@sjtu.edu.cn,zhaohai@cs.sjtu.edu.cn

Abstract

Discourse parsing is considered as one of the most challenging natural language processing (NLP) tasks. Implicit discourse relation classification is the bottleneck for discourse parsing. Without the guide of explicit discourse connectives, the relation of sentence pairs are very hard to be inferred. This paper proposes a stacking neural network model to solve the classification problem in which a convolutional neural network (CNN) is utilized for sentence modeling and a collaborative gated neural network (CGNN) is proposed for feature transformation. Our evaluation and comparisons show that the proposed model outperforms previous state-of-the-art systems.

1 Introduction

As a fundamental task in natural language processing (NLP), discourse parsing entails the discovery of the latent relational structure in multi-sentence level analysis. It is also central to many practical tasks such as question answering (Liakata et al., 2013; Jansen et al., 2014), machine translation (Meyer and Popescu-Belis, 2012; Meyer and Webber, 2013) and automatic summarization (Murray et al., 2006; Yoshida et al., 2014). Discourse parsing is also the shared task of CoNLL 2015 and 2016 (Xue et al., 2015; Xue et al., 2016), and many previous works previous on this task (Qin et al., 2016b; Li et al., 2016; Chen et al., 2015; Wang and Lan, 2016). In a discourse parser, implicit relation recognition has been the bottleneck due to lack of explicit connectives (like *"because"* or *"and"*) that can be strong indicators for the senses between adjacent clauses (Qin et al., 2016b; Pitler et al., 2009; Lin et al., 2014). This work therefore focuses on implicit relation recognition that infers the senses of the discourse relations within adjacent sentence pairs.

Most previous works on PDTB implicit relation recognition only focus on one-versus-others binary classification problems of the top level four classes (Pitler et al., 2009; Zhou et al., 2010; Park and Cardie, 2012; Biran and McKeown, 2013; Rutherford and Xue, 2014; Braud and Denis, 2015). Traditional classification methods directly rely on feature engineering, based on bag-of-words, production rules, and some linguistically-informed features (Zhou et al., 2010; Rutherford and Xue, 2014). However, discourse relations root in semantics, which may be hard to recover from surface level feature, thus these methods did not report satisfactory performance. Recently, neural network (NN) models have shown competitive or even better results than traditional linear models with handcrafted sparse features (Wang et al., 2016b; Zhang et al., 2016a; Jia and Zhao, 2014). They have been proved to be effective for many tasks (Qin et al., 2016a; Wang et al., 2016a; Zhang et al., 2016b; Wang et al., 2015; Wang et al., 2014; Cai and

*Corresponding author. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114, No. 61672343 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2263–2270,
Austin, Texas, November 1-5, 2016.

Zhao, 2016), also including discourse parsing. Ji and Eisenstein (2015) adopt recursive neural network and incorporate with entity-augmented distributed semantics. Zhang et al. (2015) explore a shallow convolutional neural network and achieve competitive performance. Although simple neural network has been shown effective, the result has not been quite satisfactory which suggests that there is still space for improving.

The concerned task could be straightforwardly formalized as a sentence-pair classification problem, which needs inferring senses solely based on the two arguments without cues of connectives. Two problems should be carefully handled in this task: how to model sentences and how to capture the interactions between the two arguments. The former could be addressed by Convolutional Neural Network (CNN) which has been proved effective for sentence modeling (Kalchbrenner et al., 2014; Kim, 2014), while the latter is the key problem, which might need deep semantic analysis for the interaction of two arguments. To solve the latter problem, we propose collaborative gated neural network (CGNN) which is partially inspired by Highway Network whose gate mechanism achieves success (Srivastava et al., 2015). Our method will be evaluated on the benchmark dataset against state-of-the-art methods.

The rest of the paper is organized as follows: Section 2 briefly describes our model, introducing the stacking architecture of CNN and CGNN, Section 3 shows the experiments and analysis, and Section 4 concludes this paper.

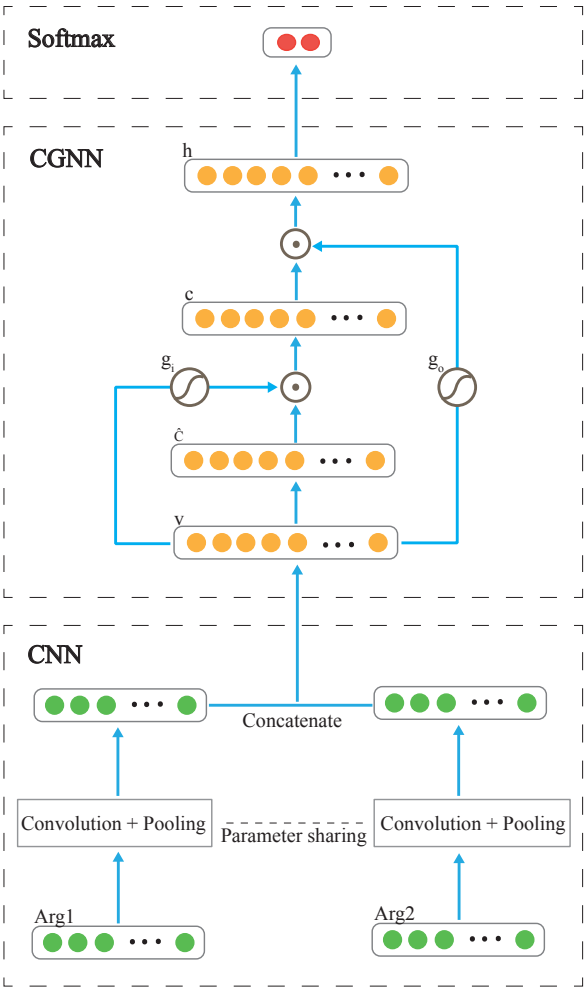


Figure 1: Model Architecture

2 Method

The architecture of the model, as shown in Figure 1, is straightforward. It can be divided into three parts: 1) CNN for modeling arguments; 2) CGNN unit for feature transformation; 3) a conventional softmax layer for the final classification. CNN is used to obtain the vector representations for the sentences, CGNN further captures and transforms the features for the final classification.

2.1 Convolutional Neural Network

As CNN has been broadly adopted for modeling sentences, we will explain it in brevity. For two arguments, typical sentence modeling process will be applied: sentence embedding (including embeddings for words and part-of-speech (POS) tags) through projection layer, convolution operations (with multiple groups of filters) through the convolution layer, obtaining the sentence representation through one-max-pooling. The two arguments will get their sentence vectors independently without any interfering, and the convolution operation will be the same by sharing parameters. The final argument-pair representation will be the vector $\mathbf{v}$ which is concatenated from two sentence vectors and this vector will be used as the input of the CGNN unit.

	COMP.	CONT.	EXP.	TEMP.	AVG
CNN Only	39.07	54.73	65.94	30.19	47.48
CNN+MLP	37.81	56.30	69.44	32.29	48.96
CNN+LSTM	39.15	53.44	68.85	29.79	47.81
CNN+Highway	37.72	56.35	68.94	30.56	48.39
CNN+CGNN	**41.55**	**57.32**	**71.50**	**35.43**	**51.45**

Table 1: F_1 scores (%) with different models.

2.2 Collaborative Gated Neural Network

For implicit sense classification, the key is how to effectively capture the interactions between the two arguments. The interactions could be word pairs, phrase pairs or even the latent meaning of the two full arguments. Pitler et al. (2009) has shown that word pair features are helpful. To model these interactions, we have to make a full use of the sentence vectors obtained from CNN. However, common neural hidden layers might be insufficient to deal with the challenge. We need to seek more powerful neural models, i.e., gated neural network.

In recent years, gated mechanism has gained popularity in neural models. Although it is first introduced in the cells of recurrent neural networks, like Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014), traditional feed-forward neural models such as the Highway Network could also benefit from it (Srivastava et al., 2015). The existing studies show that the gated mechanism in highway network serves not only a means for easier training, but also a tool to route information in a trained network.

Motivated by the idea of highway network, we propose a collaborative gated neural network (CGNN) for this task. The architecture of CGNN is illustrated in Figure 1, and it contains a sequence of transformations. First, the inner-cell $\hat{\mathbf{c}}$ is obtained through linear transformation and non-linear activation on the input $\mathbf{v}$, and this process is exactly the operation of an ordinary neural layer.

$$\hat{\mathbf{c}} = \tanh(\mathbf{W}^c \cdot \mathbf{v} + \mathbf{b}^c)$$

Meanwhile, the two gates $\mathbf{g}_i$ and $\mathbf{g}_o$ are calculated independently because they are only influenced by the original input through different parameters:

$$\mathbf{g}_i = \sigma(\mathbf{W}^i \cdot \mathbf{v} + \mathbf{b}^i)$$
$$\mathbf{g}_o = \sigma(\mathbf{W}^o \cdot \mathbf{v} + \mathbf{b}^o)$$

where the σ denotes sigmoid function which guarantees the values in the gates are in [0,1]. Two gated operations are applied sequentially, where a gated operation indicates the element-wise multiplication of an inner-cell and a gate. Between the two gated operations, a non-linear activation operation is applied. The procedure could be formulated as follows:

$$\mathbf{c} = \hat{\mathbf{c}} \odot \mathbf{g}_i$$
$$\mathbf{h} = \tanh(\mathbf{c}) \odot \mathbf{g}_o$$

where $\odot$ denotes element-wise multiplication, $\mathbf{c}$ is the second inner-cell and $\mathbf{h}$ is the output of CGNN unit.

Although the two gates are generated independently, they will work collaboratively because they control the information flow of the inner-cells sequentially which resembles logical AND operation in a probabilistic version. In fact, the transformations after $\hat{\mathbf{c}}$ will concern only element-wise operations which might give finer controls for each dimension, and the information can only flow on the dimensions where both gates are "open". This procedure will help select the most crucial features.

The gates in this model are mainly used for routing information from sentence-pairs vectors. When there is only one gate in our network, the model works similar to the highway network (Srivastava et al., 2015).

2.3 Output and Training

After the transformation of the CGNN unit, the transformed vector $\mathbf{h}$ will be sent to a conventional softmax for classification.

The training object *J* will be the cross-entropy error *E* with *L2* regularization:

$$E(\hat{y}, y) = -\sum_{j}^{l} y_j \times \log(Pr(\hat{y}_j))$$

$$J(\theta) = \frac{1}{m}\sum_{k}^{m} E(\hat{y}^{(k)}, y^{(k)}) + \frac{\lambda}{2}\|\theta\|^2$$

where y_j is the gold label and $\hat{y}_j$ is the predicted one. We adopt the diagonal variant of AdaGrad (Duchi et al., 2011) for the optimization process.

3 Experiments

3.1 Setting

As for the benchmark dataset, Penn Discourse Treebank (PDTB) (Prasad et al., 2008) corpus[1] is used for evaluation. In the PDTB, each discourse relation is annotated between two argument spans.

To be consistent with the setups of prior works, we formulate the implicit relation classification task as four one-versus-other binary classification problems only using the four top level classes: COMPARISON (COMP.), CONTINGENCY (CONT.), EXPANSION (EXP.) and TEMPORAL (TEMP.). While different works include different relations of varying specificities, all of them include these four core relations (Pitler et al., 2009). Following dataset splitting convention of the previous works, we use sections 2-20 for training, sections 21-22 for testing and sections 0-1 for development set. The proposed model is possible to be extended for multi-class classification of discourse parsing, but for the comparisons with most of previous works, we will follow them and focus on the binary classification problems.

For other hyper-parameters of the model and training process, we fix the lengths of both the input arguments to be 80, and apply truncating or zero-padding when necessary. The dimensions for word embeddings and POS embeddings are respectively 300 and 50, and the embedding layer adopts a dropout of 0.2. The word embeddings are initialized with pre-trained word vectors using *word2vec* [2] (Mikolov et al., 2013) and other parameters are randomly initialized including POS embeddings. We set the starting learning rate to 0.001. For CNN model, we utilize three groups of filters with window widths of (2, 2, 2) and their filter numbers are all set to 1024. The hyper-parameters are the same for all models and we do not tune them individually.

3.2 Model Analysis

For transformation of sentence vectors, a simple Multilayer Perceptron (MLP) layer could be a straightforward choice, while more complex neural modules, such as LSTM and highway network, could also be considered. Our model utilizes a CGNN unit with refined gated mechanism for the transformation. Will the proposed CGNN really bring about further performance improvement? We now answer this question empirically.

As shown in Table 1, CNN model usually performs well on its own. Utilizing an MLP layer or a Highway layer could improve the accuracies on CONTINGENCY, EXPANSION, TEMPORARY except for COMPARISON. Though the primary motivation of Highway is to ease gradient-based training of highly deep networks through utilizing gated units, it works merely as an ordinary MLP in the proposed model, which explains the reason that it performs like MLP. Despite one of four classes, COMPARISON, not receiving performance improvement, introducing a non-linear transformation layer lets the classification benefit as a whole. "CNN+LSTM" denotes the method of using LSTM to read the convolution sequence (without pooling operation), and it even does not perform better than MLP.

The CGNN achieves the best performance on all classes including COMPARISON. It gains 3.97% imrovement on average F1 score using CNN only model. We assume that CGNN is well-suited to work with CNN, adaptively transforming and combining local features detected by the individual filters.

3.3 Results

We show the main results in Tables 2 and 3. The metrics include precision (P), recall (R), accuracy (Acc) and F1 score. Since not all of these metrics are reported in previous work, the comparisons are correspondingly in Table 2 and 3. Some previous work merges *Entrel* with *Expansion*, which is also explored in our study and noted as EXP.+.

[1] http://www.seas.upenn.edu/~pdtb/

[2] http://www.code.google.com/p/word2vec

	COMP.		CONT.		EXP.+		TEMP.		AVG.	
	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F1	Acc
Pitler et al. (2009)	21.96	56.59	47.13	67.30	76.42	63.62	16.76	63.49	40.57	62.75
Zhou et al. (2010)	31.79	58.22	47.16	48.96	70.11	54.54	20.30	55.48	40.32	54.30
P&C (2012)	31.32	74.66	49.82	72.09	79.22	69.14	26.57	79.32	46.73	73.80
M&B (2013)	25.40	63.36	46.94	68.09	75.87	62.84	20.23	68.35	42.11	65.66
J& (2015)	35.93	70.27	52.78	76.95	80.02	69.80	27.63	87.11	49.09	76.03
B&D(2015)	36.36	-	55.76	-	61.76	-	29.30	-	45.80	-
Chen et al. (2016)	40.17	-	54.76	-	80.62	-	31.32	-	51.72	-
Current	**41.55**	71.22	**57.32**	73.80	**80.96**	68.44	**35.43**	84.32	**53.82**	74.45

Table 2: Comparisons of F_1 scores (%) (symbol + means EXP. with *Entrel*).

		P	R	F_1
COMP.	R&Xue (2014)	27.34	72.41	39.70
	Zhang et al.(2015)	22.00	67.76	33.22
	Current	29.48	70.39	**41.55**
CONT.	R&Xue (2014)	44.52	69.96	54.42
	Zhang et al.(2015)	39.80	75.29	52.04
	Current	50.69	65.95	**57.32**
EXP.	R&Xue (2014)	59.59	85.50	70.23
	Zhang et al.(2015)	56.29	91.11	69.59
	Current	60.81	86.76	**71.50**
TEMP.	R&Xue (2014)	18.52	63.64	28.69
	Zhang et al.(2015)	20.22	62.35	30.54
	Current	26.63	52.94	**35.43**
AVG.	R&Xue (2014)	37.49	72.88	48.26
	Zhang et al.(2015)	34.58	74.13	46.35
	Current	41.90	69.01	**51.45**

Table 3: Comparisons of F_1 scores (%) (EXP. without *Entrel*).

We compare with best-performed or competitive models including both traditional linear methods and recent neural methods. For traditional methods: Pitler et al. (2009) use several linguistically informed features, including polarity tags, Levin verb classes, length of verb phrases, modality, context, and lexical features; Zhou et al. (2010) improve the performance through predicting connective words as features; Park and Cardie (2012) propose a locally-optimal feature set and further identify factors for feature extraction that can have a major impact performance, including stemming and lexicon look-up; Biran and McKeown (2013) collect word pairs from arguments of explicit examples to help the learning; Rutherford and Xue (2014) employ Brown cluster pair and coreference patterns for performance enhancement. Several neural methods have also been included for comparison: Zhang et al. (2015) propose a simplified neural network which has only three different pooling operations (max, min, average); Ji and Eisenstein (2015) compute distributed semantics representation by composition up the syntactic parse tree through recursive neural network; Braud and Denis (2015) consider shallow lexical features and word embeddings. Chen et al. (2016) replace the original words by word embeddings to overcome the data sparsity problem and they also utilize gated relevance network to capture the semantic interaction between word pairs. The gated network is different from ours but also works well.

Our model achieves F-measure improvements of 1.85% on COMPARISON, 1.56% on CONTINGENCY, 1.27% on EXPANSION, 0.94% on EXPANSION+, 4.89% on TEMPORAL, against the state-of-the-art of each class. We improve by 4.73% on average F1 score when not including ENTREL in EXPANSION as reported in Table 2 and 3.19% on average F1 score otherwise as reported in Table 3. The results show that our model achieves the best performance and especially makes the most remarkable progress on TEMPORAL.

4 Conclusion

In this paper, we propose a stacking gated neural architecture for implicit discourse relation classification. Our model includes convolution and collaborative gated neural network. The analysis and experiments show that CNN performs well on its own and combining CGNN provides further gains. Our evaluation on PTDB shows that the proposed model outperforms previous state-of-the-art systems.

References

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 69–73, Sofia, Bulgaria, August.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2201–2211, Lisbon, Portugal, September.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–420, Berlin, Germany, August.

Changge Chen, Peilu Wang, and Hai Zhao. 2015. Shallow discourse parsing using constituent parsing tree. In *Proceedings of the CoNLL-15 shared task*, pages 37–41, Beijing, China, July.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1726–1735, Berlin, Germany, August.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 977–986, Baltimore, Maryland, June.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*, 3:329–344.

Zhongye Jia and Hai Zhao. 2014. A Joint Graph Model for Pinyin-to-Chinese Conversion with Typo Correction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1512–1523, Baltimore, Maryland, June.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 655–665, Baltimore, Maryland, June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October.

Zhongyi Li, Hai Zhao, Chenxi Pang, Lili Wang, and Huan Wang. 2016. A constituent syntactic parse tree based discourse parser. In *Proceedings of the CoNLL-16 shared task*, pages 60–64, Berlin, Germany, August.

Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2013. A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 747–757, Seattle, Washington, USA, October.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Thomas Meyer and Andrei Popescu-Belis. 2012. Using sense-labeled discourse connectives for statistical machine translation. In *Proceedings of the Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, pages 129–138, Avignon, France, April.

Thomas Meyer and Bonnie Webber. 2013. Implicitation of discourse connectives in (machine) translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 19–26, Sofia, Bulgaria, August.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119, South Lake Tahoe, Nevada, US, December.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pages 367–374, New York City, USA, June.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 108–112, Seoul, South Korea, July.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of he Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 683–691, Suntec, Singapore, August.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth conference on International Language Resources and Evaluation (LREC-2008)*, pages 2961–2968, Marrakech, Morocco, May.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *the 26th International Conference on Computational Linguistics (COLING)*, Osaka, Japan, December.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. Shallow discourse parsing using convolutional neural network. In *Proceedings of the CoNLL-16 shared task*, pages 70–77, Berlin, Germany, August.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 645–654, Gothenburg, Sweden, April.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Jianxiang Wang and Man Lan. 2016. Two End-to-end Shallow Discourse Parsers for English and Chinese in CoNLL-2016 Shared Task. In *Proceedings of the CoNLL-16 shared task*, pages 33–40, Berlin, Germany, August.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 189–195, Doha, Qatar, October.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Word embedding for recurrent neural network based TTS synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883, Brisbane, Australia.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2016a. Learning distributed word representations for bidirectional LSTM recurrent neural network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 527–533, San Diego, California, June.

Rui Wang, Masao Utiyama, Isao Goto, Eiichiro Sumita, Hai Zhao, and Bao-Liang Lu. 2016b. Converting continuous-space language models into n-gram language models with efficient bilingual pruning for statistical machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 15(3):11.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The CoNLL-2015 Shared Task on Shallow Discourse Parsing. In *Proceedings of the CoNLL-15 shared task*, pages 1–16, Beijing, China, July.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. CoNLL 2016 Shared Task on Multilingual Shallow Discourse Parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19, Berlin, Germany, August.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839, Doha, Qatar, October.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2230–2235, Lisbon, Portugal, September.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Hai Zhao, Graham Neubig, and Satoshi Nakamura. 2016a. Learning local word reorderings for hierarchical phrase-based statistical machine translation. *Machine Translation*, pages 1–18.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016b. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1382–1392, Berlin, Germany, August.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse con-

nectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1507–1514, Beijing, China, August.

Insertion Position Selection Model for Flexible Non-Terminals in Dependency Tree-to-Tree Machine Translation

Toshiaki Nakazawa
Japan Science and Technology Agency
5-3, Yonbancho, Chiyoda-ku, Tokyo, 102-8666, Japan
nakazawa@pa.jst.jp

John Richardson and **Sadao Kurohashi**
Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
john@nlp.ist.i.kyoto-u.ac.jp kuro@i.kyoto-u.ac.jp

Abstract

Dependency tree-to-tree translation models are powerful because they can naturally handle long range reorderings which is important for distant language pairs. The translation process is easy if it can be accomplished only by replacing non-terminals in translation rules with other rules. However it is sometimes necessary to adjoin translation rules. Flexible non-terminals have been proposed as a promising solution for this problem. A flexible non-terminal provides several insertion position candidates for the rules to be adjoined, but it increases the computational cost of decoding. In this paper we propose a neural network based insertion position selection model to reduce the computational cost by selecting the appropriate insertion positions. The experimental results show the proposed model can select the appropriate insertion position with a high accuracy. It reduces the decoding time and improves the translation quality owing to reduced search space.

1 Introduction

Tree-to-tree machine translation models currently receive limited attention. However, we believe that using target-side syntax is important to achieve high-quality translations between distant language pairs which require long range reorderings. Especially, using dependency trees on both source and target sides is promising for this purpose (Menezes and Quirk, 2007; Nakazawa and Kurohashi, 2010; Richardson et al., 2014). Tree-based translation models naturally realize word reorderings using the non-terminals or anchors for the attachment in the translation rules: therefore they do not need a reordering model which string-based models require. For example, suppose we have a translation rule with the word alignment shown in Figure 1, it is easy to translate a new input sentence which has "図書館 (*library*)" instead of "公園 (*park*)" because we can accomplish it by simply substituting "library" for the target word "park" without considering the reordering. In this case, the source word "公園" and target word "park" work as the non-terminals.

On the other hand, it is problematic when we need to adjoin a subtree which is not presented in training sentences, which we call *floating* subtree in this paper. The floating subtrees are not necessarily adjuncts, but any words or phrases. For example, suppose the Japanese input sentence in Figure 1 has "突然 (*suddenly*)", but the training corpus provides only a translation rule without the word. In this case we cannot directly use the rule for the translation because it does not know where to insert the translation of the floating word in the output. As another example, there is no context information available for the children of the OOV word in the input sentence, so we need some special process to translate them.

Previous work deals with this problem by using glue rules (Chiang, 2005) or limiting the dependency structures to be well-formed (Shen et al., 2008). Richardson et al. (2016) introduces the concept of *flexible* non-terminals. It provides multiple possible insertion positions for the floating subtree rather than fixed insertion positions. A possible insertion position must satisfy the following conditions:

- it must be a child of the aligned word of the parent of the floating subtree

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2271–2277,
Austin, Texas, November 1-5, 2016.

- it must not violate the projectivity of the dependency tree

For example, possible insertion positions for the floating word "突然" are shown in gray arrows in Figure 1. Since "突然" is a child of "電話する", and the translation of "電話する" is "called", insertion positions must be a child of "called". Also, insertion positions do not violate the projectivity of the target tree. Flexible non-terminals are analogous to the *auxiliary tree* of the tree adjoining grammars (TAG) (Joshi, 1985), which is successfully adopted in machine translation (DeNeefe and Knight, 2009). The difference is that TAG is defined on the constituency trees rather than the dependency trees.

Flexible non-terminals are powerful to handle floating subtrees and it achieve better translation quality. However the computational cost of decoding becomes high even though they are compactly represented in the lattice form (Cromieres and Kurohashi, 2014). In our experiments, using flexible non-terminals causes the decoding to be 3 to 6 times slower than when they are not used. Flexible non-terminals increase the number of translation rules because the insertion positions are selected during the decoding. However, we think it is possible to restrict possible insertion positions or even select only one insertion position by looking at the tree structures on both sides.

In this paper, we propose a method to select the appropriate insertion position before decoding. This can not only reduce the decoding time but also improve the translation quality because of reduced search space.

2 Insertion Position Selection

We assume that correct insertion positions can be determined before decoding, using the word to be inserted (I) with the context on the source side and the context of the insertion positions on the target side. On the source side, we use the parent of I (P_s) and the distance of I from P_s (D_s). On the target side, we use the previous (S_p) and next (S_n) sibling of the insertion position, the parent of the insertion position (P_t) and the distance of the insertion position from P_t (D_t). The distances are calculated on the siblings rather than the words in the sentence, and it is a positive or negative value if the insertion position is to the left or to the right of the parent respectively.

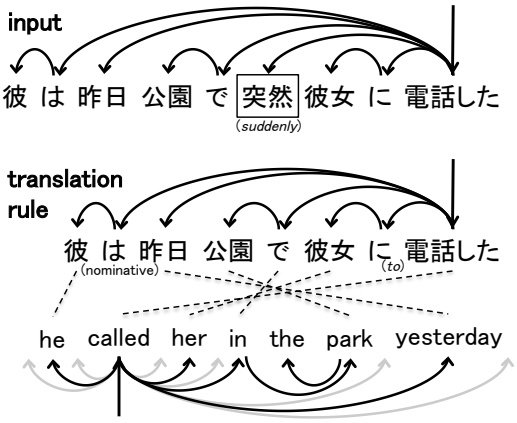


Figure 1: Example of an input sentence and a translation rule. We want to insert "突然 (*suddenly*)" which is not in the translation rule. The possible insertion positions in the target sentence are shown in gray arrows.

Taking the insertion position between "park" and "yesterday" in Figure 1 as an example, I = "突然", P_s = "電話した", D_s = +2, S_p = "park", S_n = "yesterday", P_t = "called" and D_t = -3. In cases where S_p or S_n is empty, we use special words "[[LEFT-START]]", "[[LEFT-END]]", "[[RIGHT-START]]" and "[[RIGHT-END]]". In the case of "yesterday" in Figure 1, S_p = "in" and S_n = "[[RIGHT-END]]". These clues are fed into the neural network model to solve the insertion position selection problem.

2.1 Neural Network Model

Figure 2 shows the neural network model for the insertion position selection. Given an insertion position candidate with an index k, the words (I, P_s, S_p^k, S_n^k, P_t) are first converted into vector representations through the same three embedding layers: surface form embedding (200 dimensions.), part-of-speech embedding (10 dimensions) and dependency type (or phrase category) embedding (10 dimensions), and they are concatenated to create the 220-dimension vectors. The word embedding is a randomly initialized transformation from an one-hot vector to a 200 or 10-dimensional vector, and it is learned during the whole network training.

Using these words and the distances, we create source and target context vectors c_s^k and c_t^k which represent the information of source and target sides, respectively. The distances (integer values) are directly inputted to the network. Then the context vec-

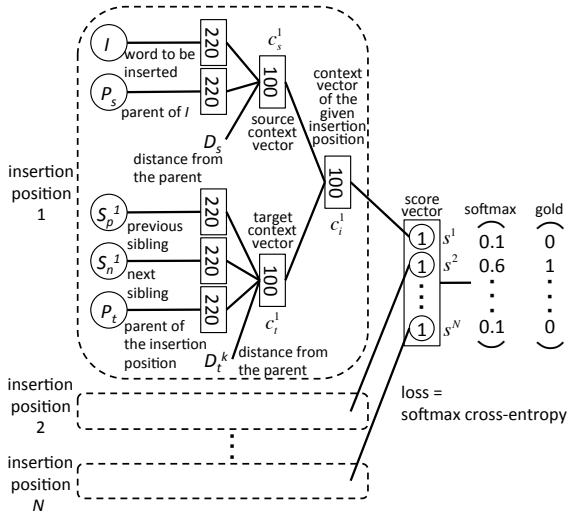


Figure 2: The neural network for the insertion position selection. The numbers inside the boxes show the dimensions of the vectors.

tor of the given insertion position c_i^k is created using c_s^k and c_t^k. Finally we get the score of the given insertion position s^k from c_i^k. These vectors are calculated as follows:

$$\begin{aligned} c_s^k &= \tanh(W_{c_s}[I; P_s; D_s]) \\ c_t^k &= \tanh(W_{c_t}[S_p; S_n; P_t; D_t^k]) \\ c_i^k &= \tanh(W_{c_i}[c_s^k; c_t^k]) \\ s^k &= W_s c_i^k \end{aligned}$$

where ";" means concatenation of the vectors. The size of c_s^k, c_t^k and c_i^k is 100 in our experiments.

The same network is applied to all the other insertion positions and get their scores. Finally the scores are normalized by the softmax function, and the loss is calculated by the softmax cross-entropy as the loss function. All the links between layers are fully-connected. We use dropout (50%) to avoid overfitting.

2.2 Training Data Generation

The data for training the neural network model can be automatically generated from the word-aligned parallel corpus with dependency parses in both sides by Algorithm 1. The ALIGNMENT function returns the aligned word in the target tree for the given source word[1], and the ISPARENTCHILD function returns TRUE if P_t is the parent of C_t.

[1]In case of the multiple word alignment, we only use the root word of them in both source and target sides.

Algorithm 1 Training Data Generation for NN

```
for all P_s ∈ words in source tree do
    P_t = ALIGNMENT(P_s)
    for all C_s ∈ CHILDREN(P_s) do
        C_t = ALIGNMENT(C_s)
        if ISPARENTCHILD(P_t, C_t) then
            GENERATEDATA(P_s, C_s, P_t, C_t)
        end if
    end for
end for
```

	Ja ↔ En	Ja ↔ Zh
Training	2,020,106	667,520
Development	1,789	2,115
Test	1,812	2,174

Table 1: The number of sentences in ASPEC used in our experiments.

The GENERATEDATA function generates one instance of training data to predict the position of C_t from P_s, C_s and P_t with their contexts by removing C_t in the target tree. The position where C_t exists is regarded as the correct insertion position, and others as incorrect insertion positions. Note that C_s corresponds to I in Figure 2.

2.3 Insertion Position Selection in Translation

Once the neural network model is trained, it can be applied to select the most appropriate insertion positions in the translation rules for the given floating subtree by looking at the score of each insertion position. Translation rules only contain part of the original parallel sentence in most of the cases. This means that the context used for selecting the insertion position is different from that in the training data for the neural network. For example, if the input sentence does not have "公園 で (*in the park*)" in Figure 1, the number of possible insertion positions is 6 and we do not use "in" as the context. However, this is not so problematic because similar or same context may appear in the different part of the corpus.

3 Experiments

We conducted two kinds of experiments: the insertion position selection and translation. We used ASPEC (Nakazawa et al., 2016) as the dataset and the numbers of the sentences of the corpus are shown in Table 1. The Japanese morphological analyzer (Kurohashi et al., 1994) and dependency parser (Kurohashi and Nagao, 1994) are used for Japanese

	Ja → En	En → Ja	Ja → Zh	Zh → Ja
Training	15.7M		5.7M	
Development	160K		58K	
Test	160K		58K	
ave. # IP	3.39	3.15	3.72	3.41
best epoch	89	71	61	79
mean loss	0.089	0.058	0.105	0.056
Accuracy (%)	97.08	97.72	96.51	97.99
Logit Accuracy (%)	55.00	89.03	68.04	83.16

Table 2: The statistics of the data and results of the insertion position selection experiments.

sentences. English sentences are first parsed by nlparser (Charniak and Johnson, 2005) and then converted into word dependency trees using Collins' head percolation table (Collins, 1999). We used Chinese word segmenter KKN (Shen et al., 2014) and dependency parser SKP (Shen et al., 2012) for Chinese sentences. The supervised word alignment Nile (Riesa et al., 2011) was used.

We used a state-of-the-art dependency tree-to-tree decoder (Richardson et al., 2014) with the default settings. The neural network is constructed and trained using the Chainer (Tokui et al., 2015).

3.1 Insertion Position Selection

The training, development and test data for the neural network is automatically generated by the procedure explained in Section 2.2. The size of the generated data from the ASPEC and the average number of insertion positions for each floating subtree are shown in Table 2. We trained the model for 100 epochs and used the best model on the development data for testing. The vocabulary size for the surface form was 50,000.

For comparison, we also tried the logistic regression to predict the correct insertion positions. Because our training data is huge, we used Multi-core LIBLINEAR[2] with L2-regularized logistic regression (primal) solver. The format of training instances are: one-hot (binary) vectors for surface form, POS and dependency type, and distances scaled to [0, 1]. We first find the best value for the C parameter, then train the model. The best insertion position is selected using the estimated probabilities for each insertion position.

The experimental results are also shown in Table 2. We evaluated the results by the mean loss of the model and the accuracy on the test data. The result shows that our model can select the correct insertion position with very high accuracy for every language pair while the classical logistic regression model cannot. This supports our claim stated in the beginning of Section 2.

X → Ja is easier and achieved slightly better accuracy than the reverse direction because Japanese is a head-final language and all children are generally put on the left of their parents. There are some cases judged as incorrect but acceptable insertion positions, and hence the true accuracies are higher than the ones reported above. We also investigated the top-2 accuracy and found that it is above 99.0% for Ja → X and 99.5% for X → Ja.

Table 3 shows the detailed result of Ja $rightarrow$ En experiment. The number of insertion-position is at least 2 (left/right of the parent) and it is easy to solve (more than 99% accuracy). 3 is a situation where the parent has one child, and it is still not so difficult (97-98% accuracy). About 70% of the test data have only 2 or 3 insertion-positions. Difficult cases are the sentences which have many adjuncts as in Figure 1, but we used the scientific paper corpora, where not so many adjuncts appear.

3.2 Translation

We conducted translation experiment using the ASPEC in 3 settings:

- No Flexible: not using the flexible non-terminals and using simple glue rules as in the baseline model of (Richardson et al., 2016) [3]
- Baseline: using the flexible non-terminals without the insertion position selection
- Proposed: using only the most appropriate insertion position for the flexible non-terminals

We also report the translation quality of conventional models for comparison: phrase-based SMT (PBSMT) and hierarchical phrase-based SMT (Hiero). We used the default settings of Moses except -distortion-limit=20 for PBSMT.

The translation quality is evaluated by the automatic evaluation measures BLEU (Papineni et al.,

[2]https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/multicore-liblinear/

[3]52.5% of all the translation rules require glue rule, but it is applied to 22.6% of the rules actually used in the translation.

# of ins. pos. and ratio	Top N accuracy (%)								
	1	2	3	4	5	6	7	8	9
2 (49.39%)	99.22								
3 (18.15%)	96.37	99.60							
4 (11.77%)	95.76	99.06	99.82						
5 (6.30%)	94.24	98.39	99.38	99.85					
6 (4.99%)	93.50	97.82	99.11	99.63	99.98				
7 (3.96%)	93.13	97.40	98.87	99.56	99.89	99.98			
8 (2.42%)	92.77	97.07	98.43	99.23	99.61	99.77	99.97		
9 (1.50%)	92.14	96.19	97.85	99.01	99.50	99.67	99.92	100.00	
10 (0.77%)	92.04	96.62	97.75	98.47	99.03	99.52	99.92	99.92	100.00

Table 3: Detailed Ja → En insertion position selection experimental result.

	Ja → En			En → Ja			Ja → Zh			Zh → Ja		
	BLEU	RIBES	Time	BLEU	RIBES	Time	BLEU	RIBES	Time	BLEU	RIBES	Time
PBSMT	18.45	64.51	-	27.48	68.37	-	27.96	78.90	-	34.65	77.25	-
Hiero	18.72	65.11	-	30.19	73.47	-	27.71	**80.91**	-	**35.43**	**81.04**	-
No Flexible	20.28	65.08	1.00	28.77	75.21	1.00	24.85	66.60	1.00	30.51	73.08	1.00
Baseline	21.61	69.82	6.28	**30.57**	76.13	3.30	28.79	78.11	5.16	34.32	77.82	5.28
Proposed	**22.07**†	**70.49**†	2.25	30.50	**76.69**†	1.27	**29.83**†	79.73†	2.21	34.71†	79.25†	1.89

Table 4: The results of the translation experiments. † means the Proposed method achieved significantly better score than the Baseline ($p < 0.01$).

2002) and RIBES (Isozaki et al., 2010) with the significance testing by bootstrap resampling (Koehn, 2004). RIBES is more sensitive to word order than BLEU, so we expect an improvement in RIBES. We also investigated relative decoding time compared to the No Flexible setting. Note that we used the word "decoding" for only exploring the search space, and it does not include constructing the search space (as the table lookup in Phrase-based SMT). Our whole translation process is:

1. translation rule extraction
2. insertion-position selection
3. decoding

At the time of the second step, we have all the translation rules applicable to the input sentence. The computation time for each step is $3 \gg 1 \gg 2$ so we only focus on the time for step 3 in the experiments (the computation time for step 2 is negligibly small).

The results are shown in Table 4. The Proposed method achieved significantly better automatic evaluation scores than the Baseline for all the language pairs except the BLEU score of En → Ja direction. Also, the decoding time is reduced by about 60% relative to that of the Baseline.

Our tree-based model is better than the conventional models except C → J, where the accuracy of Chinese parsing for the input sentences has a bad effect.

4 Conclusion

In this paper we have proposed a neural network based insertion position selection model to reduce the computational cost of the decoding for dependency tree-to-tree translation with flexible non-terminals. The model successfully finds the appropriate insertion position from the candidates and it leads to faster translation speed and better translation quality due to the reduced search space.

Currently, we use only words as the context but it seems promising to use subtrees as well. For example, using the information of the subtree "in the park" is more informative than using only "in" in Figure 1. This is especially important for Japanese as the target language because children of verbs are often case markers and they do not provide enough information when selecting the appropriate insertion position. It is possible to adopt existing models of creating vector representation of dependency subtrees such as the model using recursive neural networks (Liu et al., 2015) and convolutional neural networks (Mou et al., 2015).

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270. Association for Computational Linguistics.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Fabien Cromieres and Sadao Kurohashi. 2014. Translation rules with right-hand side lattices. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 577–588. Association for Computational Linguistics.

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 206–250. Cambridge University Press, Cambridge.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of The International Workshop on Sharable Natural Language*, pages 22–28.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290. Association for Computational Linguistics.

Arul Menezes and Chris Quirk, 2007. *Proceedings of the Second Workshop on Statistical Machine Translation*, chapter Using Dependency Order Templates to Improve Generality in Translation, pages 1–8. Association for Computational Linguistics.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2315–2325, Lisbon, Portugal, September. Association for Computational Linguistics.

Toshiaki Nakazawa and Sadao Kurohashi. 2010. Fully syntactic ebmt system of kyoto team in ntcir-8. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies (NTCIR-8)*, pages 403–410.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portorož, Slovenia, May.

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

John Richardson, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2014. Kyotoebmt: An example-based dependency-to-dependency translation framework. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84. Association for Computational Linguistics.

John Richardson, Fabien Cromierès, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Flexible non-terminals for dependency tree-to-tree reordering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 11–19. Association for Computational Linguistics.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical*

Methods in Natural Language Processing, pages 497–507. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Association for Computational Linguistics*.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2012. A reranking approach for dependency parsing with variable-sized subtree features. In *Proceedings of 26th Pacific Asia Conference on Language Information and Computing*, pages 308–317.

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese morphological analysis with character-level pos tagging (short paper). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*, Baltimore, USA.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Why Neural Translations are the Right Length

Xing Shi[1], **Kevin Knight**[1], **and Deniz Yuret**[2]
[1]Information Sciences Institute & Computer Science Department
University of Southern California
{xingshi,knight}@isi.edu
[2]Computer Engineering, Koç University
dyuret@ku.edu.tr

Abstract

We investigate how neural, encoder-decoder translation systems output target strings of appropriate lengths, finding that a collection of hidden units learns to explicitly implement this functionality.

1 Introduction

The neural encoder-decoder framework for machine translation (Neco and Forcada, 1997; Castaño and Casacuberta, 1997; Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015) provides new tools for addressing the field's difficult challenges. In this framework (Figure 1), we use a recurrent neural network (*encoder*) to convert a source sentence into a dense, fixed-length vector. We then use another recurrent network (*decoder*) to convert that vector into a target sentence. In this paper, we train long short-term memory (LSTM) neural units (Hochreiter and Schmidhuber, 1997) trained with back-propagation through time (Werbos, 1990).

A remarkable feature of this simple neural MT (NMT) model is that it produces translations of the right length. When we evaluate the system on previously unseen test data, using BLEU (Papineni et al., 2002), we consistently find the length ratio between MT outputs and human references translations to be very close to 1.0. Thus, no brevity penalty is incurred. This behavior seems to come for free, without special design.

By contrast, builders of standard statistical MT (SMT) systems must work hard to ensure correct length. The original mechanism comes from the IBM SMT group, whose famous Models 1-5 included a learned table $\epsilon(y|x)$, with x and y being the lengths of source and target sentences (Brown et al., 1993). But they did not deploy this table when decoding a foreign sentence f into an English sentence e; it did not participate in incremental scoring and pruning of candidate translations. As a result (Brown et al., 1995):

"However, for a given f, if the goal is to discover the most probable e, then the product P(e) P(f|e) is too small for long English strings as compared with short ones. As a result, short English strings are improperly favored over longer English strings. This tendency is counteracted in part by the following modification: Replace P(f|e) with $c^{length(e)} \cdot P(f|e)$ *for some empirically chosen constant c. This modification is treatment of the symptom rather than treatment of the disease itself, but it offers some temporary relief. The cure lies in better modeling."*

More temporary relief came from Minimum Error-Rate Training (MERT) (Och, 2003), which automatically sets c to maximize BLEU score. MERT also sets weights for the language model P(e), translation model P(f|e), and other features. The length feature combines so sensitively with other features that MERT frequently returns to it as it revises one weight at a time.

NMT's ability to correctly model length is remarkable for these reasons:

- SMT relies on maximum BLEU training to obtain a length ratio that is prized by BLEU, while NMT obtains the same result through generic maximum likelihood training.
- Standard SMT models explicitly "cross off"

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2278–2282,
Austin, Texas, November 1-5, 2016.

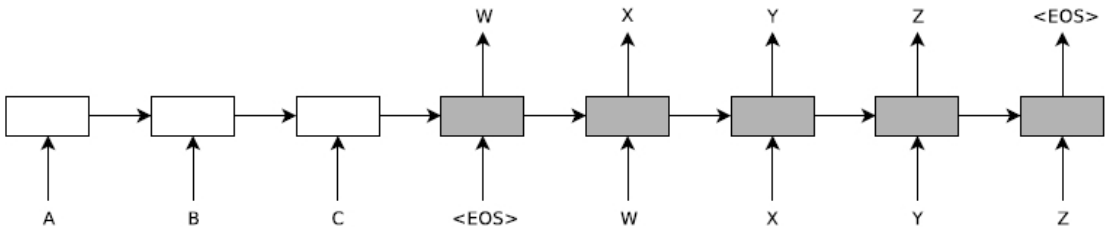


Figure 1: The encoder-decoder framework for neural machine translation (NMT) (Sutskever et al., 2014). Here, a source sentence C B A (fed in reverse as A B C) is translated into a target sentence W X Y Z. At each step, an evolving real-valued vector summarizes the state of the encoder (left half) and decoder (right half).

source words and phrases as they are translated, so it is clear when an SMT decoder has finished translating a sentence. NMT systems lack this explicit mechanism.

- SMT decoding involves heavy search, so if one MT output path delivers an infelicitous ending, another path can be used. NMT decoding explores far fewer hypotheses, using a tight beam without recombination.

In this paper, we investigate how length regulation works in NMT.

2 A Toy Problem for Neural MT

We start with a simple problem in which source strings are composed of symbols *a* and *b*. The goal of the translator is simply to copy those strings. Training cases look like this:

a a a b b a <EOS>	→ *a a a b b a* <EOS>
b b a <EOS>	→ *b b a* <EOS>
a b a b a b a a <EOS>	→ *a b a b a b a a* <EOS>
b b a b b a b b a <EOS>	→ *b b a b b a b b a* <EOS>

The encoder must summarize the content of any source string into a fixed-length vector, so that the decoder can then reconstruct it.[1] With 4 hidden LSTM units, our NMT system can learn to solve this problem after being trained on 2500 randomly chosen strings of lengths up to 9.[2] [3]

To understand how the learned system works, we encode different strings and record the resulting LSTM cell values. Because our LSTM has four hidden units, each string winds up at some point in four-dimensional space. We plot the first two dimensions (unit_1 and unit_2) in the left part of Figure 2, and we plot the other two dimensions (unit_3 and unit_4) in the right part. There is no dimension reduction in these plots. Here is what we learn:

- unit_1 records the approximate length of the string. Encoding a string of length 7 may generate a value of -6.99 for unit_1.
- unit_2 records the number of *b*'s minus the number of *a*'s, thus assigning a more positive value to *b*-heavy strings. It also includes a +1 bonus if the string ends with *a*.
- unit_3 records a prefix of the string. If its value is less than 1.0, the string starts with *b*. Otherwise, it records the number of leading *a*'s.
- unit_4 has a more diffuse function. If its value is positive, then the string consists of all *b*'s (with a possible final *a*). Otherwise, its value correlates with both negative length and the preponderance of *b*'s.

For our purposes, unit_1 is the interesting one. Figure 3 shows the progression of "*a b a b b b*" as it gets encoded (top figure), then decoded (bottom two figures). During encoding, the value of unit_1 decreases by approximately 1.0 each time a letter is read. During decoding, its value increases each time a letter is written. When it reaches zero, it signals the decoder to output <EOS>.

The behavior of unit_1 shows that the translator incorporates explicit length regulation. It also explains two interesting phenomena:

- When asked to transduce previously-unseen strings up to length 14, the system occasionally makes a mistake, mixing up an *a* or *b*. However, the output length is never wrong.[4]

[1] We follow Sutskever et al. (2014) in feeding the input string backwards to the encoder.

[2] Additional training details: 100 epochs, 100 minibatch size, 0.7 learning rate, 1.0 gradient clipping threshold.

[3] We use the toolkit: https://github.com/isi-nlp/Zoph_RNN

[4] Machine translation researchers have also noticed that

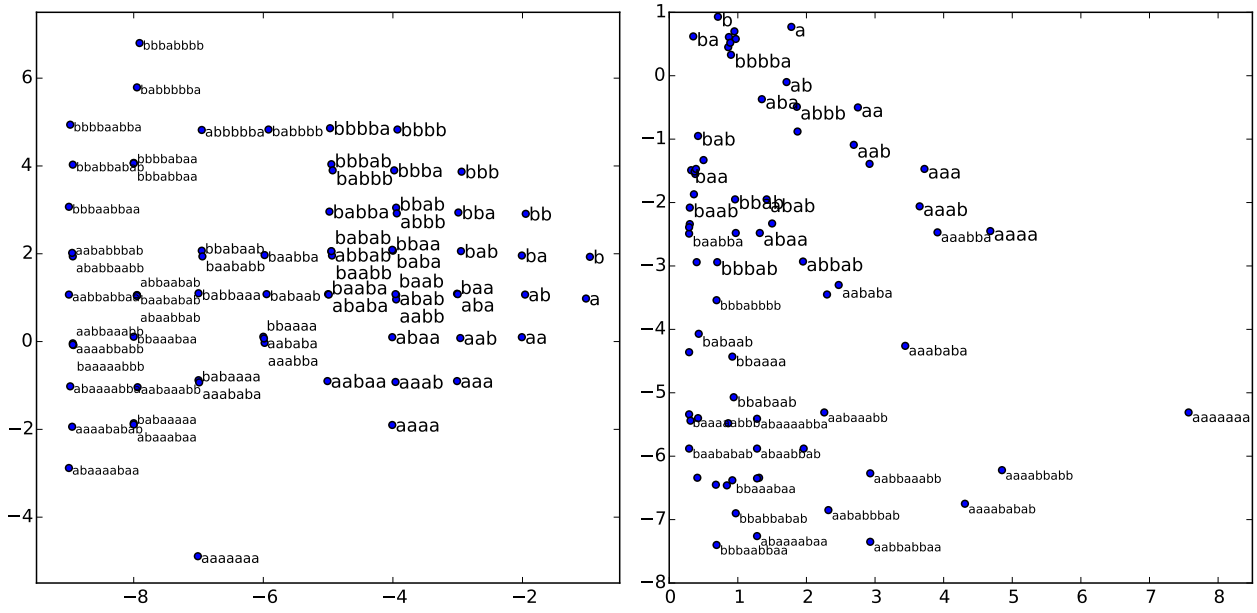


Figure 2: After learning, the recurrent network can convert any string of *a*'s and *b*'s into a 4-dimensional vector. The left plot shows the encoded strings in dimensions described by the cell states of LSTM unit_1 (x-axis) and unit_2 (y-axis). unit_1 learns to record the length of the string, while unit_2 records whether there are more *b*'s than *a*'s, with a +1 bonus for strings that end in *a*. The right plot shows the cell states of LSTM unit_3 (x-axis) and unit_4 (y-axis). unit_3 records how many *a*'s the string begins with, while unit_4 correlates with both length and the preponderance of *b*'s. Some text labels are omitted for clarity.

- When we ask the system to transduce very long strings, beyond what it has been trained on, its output length may be slightly off. For example, it transduces a string of 28 *b*'s into a string of 27 *b*'s. This is because unit_1 is not incremented and decremented by exactly 1.0.

3 Full-Scale Neural Machine Translation

Next we turn to full-scale NMT. We train on data from the WMT 2014 English-to-French task, consisting of 12,075,604 sentence pairs, with 303,873,236 tokens on the English side, and 348,196,030 on the French side. We use 1000 hidden LSTM units. We also use *two layers* of LSTM units between source and target.[5]

After the LSTM encoder-decoder is trained, we send test-set English strings through the encoder portion. Every time a word token is consumed, we record the LSTM cell values and the length of the string so far. Over 143,379 token observations, we investigate how the LSTM encoder tracks length.

Top 10 units by ...	1st layer	2nd layer
Individual R^2	0.868	0.947
Greedy addition	0.968	0.957
Beam search	0.969	0.958

Table 1: R^2 values showing how differently-chosen sets of 10 LSTM hidden units correlate with length in the NMT encoder.

With 1000 hidden units, it is difficult to build and inspect a heat map analogous to Figure 3. Instead, we seek to predict string length from the cell values, using a weighted, linear combination of the 1000 LSTM cell values. We use the least-squares method to find the best predictive weights, with resulting R^2 values of 0.990 (for the first layer, closer to source text) and 0.981 (second layer). So the entire network records length very accurately.

However, unlike in the toy problem, no single unit tracks length perfectly. The best unit in the second layer is unit_{109}, which correlates with R^2=0.894.

We therefore employ three mechanisms to locate

when the translation is completely wrong, the length is still correct (anonymous).

[5]Additional training details: 8 epochs, 128 minibatch size, 0.35 learning rate, 5.0 gradient clipping threshold.

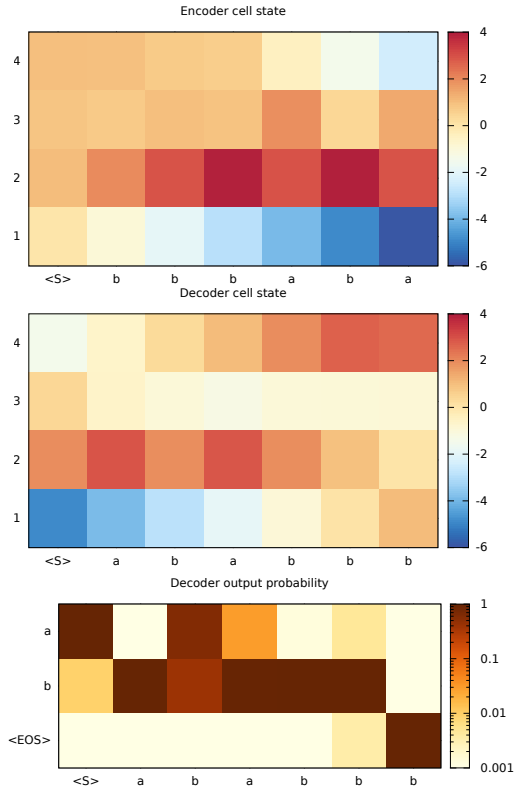


Figure 3: The progression of LSTM state as the recurrent network encodes the string "*a b a b b b*". Columns show the inputs over time and rows show the outputs. Red color indicates positive values, and blue color indicates negative. The value of unit$_1$ decreases during the encoding phase (top figure) and increases during the decoding phase (middle figure). The bottom figure shows the decoder's probability of ending the target string (<EOS>).

k	Best subset of LSTM's 1000 units	R^2
1	109	0.894
2	334, 109	0.936
3	334, 442, 109	0.942
4	334, 442, 109, 53	0.947
5	334, 442, 109, 53, 46	0.951
6	334, 442, 109, 53, 46, 928	0.953
7	334, 442, 109, 53, 46, 433, 663	0.955

Table 2: Sets of k units chosen by beam search to optimally track length in the NMT encoder. These units are from the LSTM's second layer.

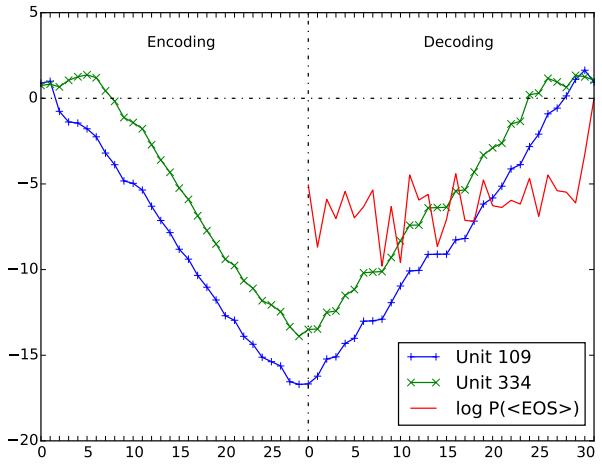


Figure 4: Action of translation unit$_{109}$ and unit$_{334}$ during the encoding and decoding of a sample sentence. Also shown is the softmax log-prob of output <EOS>.

a subset of units responsible for tracking length. We select the top k units according to: (1) individual R^2 scores, (2) greedy search, which repeatedly adds the unit which maximizes the set's R^2 value, and (3) beam search. Table 1 shows different subsets we obtain. These are quite predictive of length. Table 2 shows how R^2 increases as beam search augments the subset of units.

4 Mechanisms for Decoding

For the toy problem, Figure 3 (middle part) shows how the cell value of unit$_1$ moves back to zero as the target string is built up. It also shows (lower part) how the probability of target word <EOS> shoots up once the correct target length has been achieved.

MT decoding is trickier, because source and target strings are not necessarily the same length, and

target length depends on the words chosen. Figure 4 shows the action of unit_{109} and unit_{334} for a sample sentence. They behave similarly on this sentence, but not identically. These two units do not form a simple switch that controls length—rather, they are high-level features computed from lower/previous states that contribute quantitatively to the decision to end the sentence.

Figure 4 also shows the log P(<EOS>) curve, where we note that the probability of outputting <EOS> rises sharply (from 10^{-8} to 10^{-4} to 0.998), rather than gradually.

5 Conclusion

We determine how target length is regulated in NMT decoding. In future work, we hope to determine how other parts of the translator work, especially with reference to grammatical structure and transformations.

Acknowledgments

This work was supported by ARL/ARO (W911NF-10-1-0533), DARPA (HR0011-15-C-0115), and the Scientific and Technological Research Council of Turkey (TÜBİTAK) (grants 114E628 and 215E201).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

P. Brown, S. della Pietra, V. della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

P. F. Brown, J. Cocke, S. della Pietra, V. della Pietra, F. Jelinek, J. C. Lai, and R. L. Mercer. 1995. Method and system for natural language translation. US Patent 5,477,451.

M. A. Castaño and F. Casacuberta. 1997. A connectionist approach to machine translation. In *EUROSPEECH*.

S. Hochreiter and J. Schmidhuber. 1997. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, pages 473–479.

M. Luong, H. Pham, and C. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

R. Neco and M. Forcada. 1997. Asynchronous translations with recurrent neural nets. In *International Conf. on Neural Networks*, volume 4, pages 2535–2540.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*.

P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Supervised Attentions for Neural Machine Translation

Haitao Mi **Zhiguo Wang** **Abe Ittycheriah**
T.J. Watson Research Center
IBM
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{hmi, zhigwang, abei}@us.ibm.com

Abstract

In this paper, we improve the attention or alignment accuracy of neural machine translation by utilizing the alignments of training sentence pairs. We simply compute the distance between the machine attentions and the "true" alignments, and minimize this cost in the training procedure. Our experiments on large-scale Chinese-to-English task show that our model improves both translation and alignment qualities significantly over the large-vocabulary neural machine translation system, and even beats a state-of-the-art traditional syntax-based system.

1 Introduction

Neural machine translation (NMT) has gained popularity in recent two years (e.g. (Bahdanau et al., 2014; Jean et al., 2015; Luong et al., 2015; Mi et al., 2016b; Li et al., 2016), especially for the attention-based models of Bahdanau et al. (2014).

The attention model plays a crucial role in NMT, as it shows which source word(s) the model should focus on in order to predict the next target word. However, the attention or alignment quality of NMT is still very low (Mi et al., 2016a; Tu et al., 2016).

In this paper, we alleviate the above issue by utilizing the *alignments* (human annotated data or machine alignments) of the training set. Given the alignments of all the training sentence pairs, we add an alignment distance cost to the objective function. Thus, we not only maximize the log translation probabilities, but also minimize the alignment distance cost. Large-scale experiments over Chinese-to-English on various test sets show that our best method for a single system improves the translation quality significantly over the large vocabulary NMT system (Section 5) and beats the state-of-the-art syntax-based system.

2 Neural Machine Translation

As shown in Figure 1, attention-based NMT (Bahdanau et al., 2014) is an encoder-decoder network. the encoder employs a bi-directional recurrent neural network to encode the source sentence $\mathbf{x} = (x_1, ..., x_l)$, where l is the sentence length (including the end-of-sentence $\langle$eos$\rangle$), into a sequence of hidden states $\mathbf{h} = (h_1, ..., h_l)$, each h_i is a concatenation of a left-to-right $\overrightarrow{h_i}$ and a right-to-left $\overleftarrow{h_i}$.

Given $\mathbf{h}$, the decoder predicts the target translation by maximizing the conditional log-probability of the correct translation $\mathbf{y}^* = (y_1^*, ...y_m^*)$, where m is the sentence length (including the end-of-sentence). At each time t, the probability of each word y_t from a target vocabulary V_y is:

$$p(y_t|\mathbf{h}, y_{t-1}^*..y_1^*) = g(s_t, y_{t-1}^*), \tag{1}$$

where g is a two layer feed-forward neural network over the embedding of the previous word y_{t-1}^*, and the hidden state s_t. The s_t is computed as:

$$s_t = q(s_{t-1}, y_{t-1}^*, H_t) \tag{2}$$

$$H_t = \begin{bmatrix} \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \tag{3}$$

where q is a gated recurrent units, H_t is a weighted sum of $\mathbf{h}$; the weights, α, are computed with a two layer feed-forward neural network r:

$$\alpha_{t,i} = \frac{\exp\{r(s_{t-1}, h_i, y_{t-1}^*)\}}{\sum_{k=1}^{l} \exp\{r(s_{t-1}, h_k, y_{t-1}^*)\}} \tag{4}$$

We put all $\alpha_{t,i}$ ($t = 1...m$, $i = 1...l$) into a matrix $\mathcal{A}'$, we have a matrix (alignment) like (c) in Figure 2, where each row (for each target word) is a probability distribution over the source sentence $\mathbf{x}$.

The training objective is to maximize the conditional log-probability of the correct translation y^*

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2283–2288,
Austin, Texas, November 1-5, 2016.

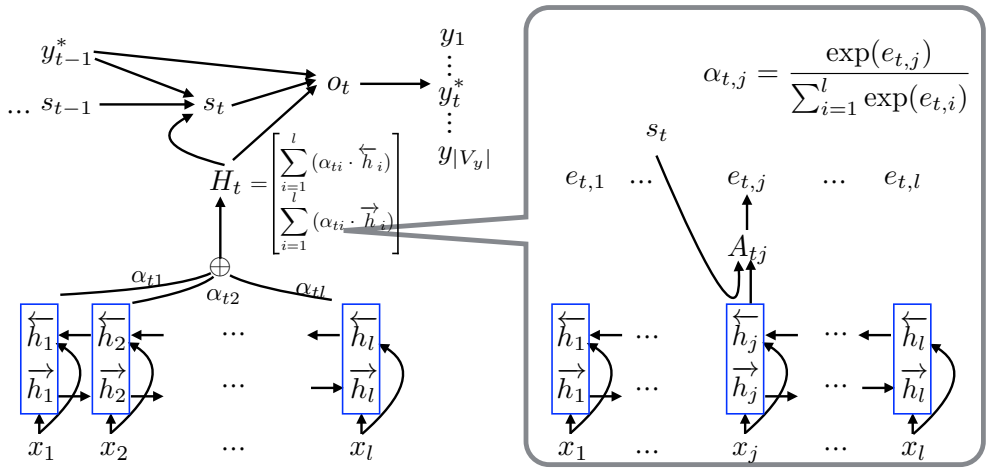


Figure 1: The architecture of attention-based NMT (Bahdanau et al., 2014). The source sentence $\mathbf{x} = (x_1, ..., x_l)$ with length l, x_l is an end-of-sentence token $\langle$eos$\rangle$ on the source side. The reference translation is $\mathbf{y}^* = (y^*_1, ..., y^*_m)$ with length m, similarly, y^*_m is the target side $\langle$eos$\rangle$. $\overleftarrow{h_i}$ and $\overrightarrow{h_i}$ are bi-directional encoder states. $\alpha_{t,j}$ is the attention probability at time t, position j. H_t is the weighted sum of encoding states. s_t is a hidden state. o_t is an output state. Another one layer neural network projects o_t to the target output vocabulary, and conducts softmax to predict the probability distribution over the output vocabulary. The attention model (the right box) is a two layer feedforward neural network, $A_{t,j}$ is an intermediate state, then another layer converts it into a real number $e_{t,j}$, the final attention probability at position j is $\alpha_{t,j}$.

given x with respect to the parameters θ

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N}\sum_{t=1}^{m} \log p(y^{*n}_t | \mathbf{x}^n, y^{*n}_{t-1}..y^{*n}_1), \tag{5}$$

where n is the n-th sentence pair $(\mathbf{x}^n, \mathbf{y}^{*n})$ in the training set, N is the total number of pairs.

3 Alignment Component

The attentions, $\alpha_{t,1}...\alpha_{t,l}$, in each step t play an important role in NMT. However, the accuracy is still far behind the traditional MaxEnt alignment model in terms of alignment F1 score (Mi et al., 2016b; Tu et al., 2016). Thus, in this section, we explicitly add an alignment distance to the objective function in Eq. 5. The "truth" alignments for each sentence pair can be from human annotated data, unsupervised or supervised alignments (e.g. GIZA++ (Och and Ney, 2000) or MaxEnt (Ittycheriah and Roukos, 2005)).

Given an alignment matrix $\mathcal{A}$ for a sentence pair $(\mathbf{x}, \mathbf{y})$ in Figure 2 (a), where we have an end-of-source-sentence token $\langle$eos$\rangle = x_l$, and we align all the unaligned target words (y^*_3 in this example) to $\langle$eos$\rangle$, also we force y^*_m (end-of-target-sentence) to be aligned to x_l with probability one. Then we conduct two transformations to get the probability distribution matrices ((b) and (c) in Figure 2).

3.1 Simple Transformation

The first transformation simply normalizes each row. Figure 2 (b) shows the result matrix $\mathcal{A}^*$. The last column in red dashed lines shows the alignments of the special end-of-sentence token $\langle$eos$\rangle$.

3.2 Smoothed Transformation

Given the original alignment matrix $\mathcal{A}$, we create a matrix $\mathcal{A}^*$ with all points initialized with zero. Then, for each alignment point $\mathcal{A}_{t,i} = 1$, we update $\mathcal{A}^*$ by adding a Gaussian distribution, $g(\mu, \sigma)$, with a window size w (t-w, ... t ... t+w). Take the $\mathcal{A}_{1,1} = 1$ for example, we have $\mathcal{A}^*_{1,1}$ += 1, $\mathcal{A}^*_{1,2}$ += 0.61, and $\mathcal{A}^*_{1,3}$ += 0.14 with w=2, $g(\mu, \sigma)$=$g(0, 1)$. Then we normalize each row and get (c). In our experiments, we use a shape distribution, where σ = 0.5.

3.3 Objectives

Alignment Objective: Given the "true" alignment $\mathcal{A}^*$, and the machine attentions $\mathcal{A}'$ produced by NMT model, we compute the Euclidean distance bewteen $\mathcal{A}^*$ and $\mathcal{A}'$.

$$d(\mathcal{A}', \mathcal{A}^*) = \sqrt{\sum_{t=1}^{m}\sum_{i=1}^{l}(\mathcal{A}'_{t,i} - {\mathcal{A}^*}_{t,i})^2}. \tag{6}$$

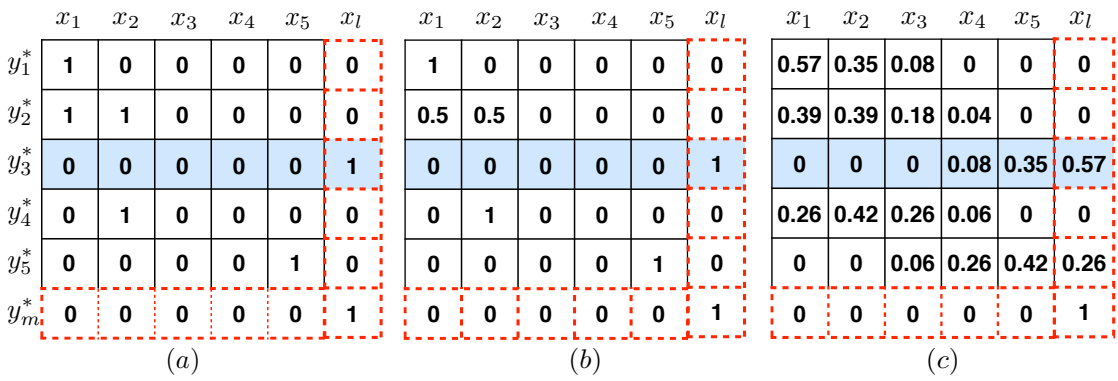


Figure 2: Alignment transformation. A special token, $\langle \text{eos} \rangle$, is introduced to the source sentence, we align all the unaligned target words (y_3^* in this case) to $\langle \text{eos} \rangle$. (a): the original alignment matrix $\mathcal{A}$ from GIZA++ or MaxEnt aligner. (b): simple normalization by rows (probability distribution over the source sentence $\mathbf{x}$). (c): smoothed transformation followed by normalization by rows, and typically, we always align end-of-source-sentence x_l to end-of-target-sentence y_m by probability one.

NMT Objective: We plug Eq. 6 to Eq. 5, we have

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \Big\{ \sum_{t=1}^{m} \log p(y_t^{*n} | \mathbf{x}^n, y_{t-1}^{*n} .. y_1^{*n}) - d(\mathcal{A}'^n, \mathcal{A}^{*n}) \Big\}. \tag{7}$$

There are two parts: translation and alignment, so we can optimize them jointly, or separately (e.g. we first optimize alignment only, then optimize translation). Thus, we divide the network in Figure 1 into alignment **A** and translation **T** parts:

- **A**: all networks before the hidden state s_t,
- **T**: the network $g(s_t, y_{t-1}^*)$.

If we only optimize **A**, we keep the parameters in **T** unchanged. We can also optimize them jointly **J**. In our experiments, we test different optimization strategies.

4 Related Work

In order to improve the attention or alignment accuracy, Cheng et al. (2016) adapted the agreement-based learning (Liang et al., 2006; Liang et al., 2008), and introduced a combined objective that takes into account both translation directions (source-to-target and target-to-source) and an agreement term between the two alignment directions. By contrast, our approach directly uses and optimizes NMT parameters using the "supervised" alignments.

5 Experiments

5.1 Data Preparation

We run our experiments on Chinese to English task. The training corpus consists of approximately 5 million sentences available within the DARPA BOLT Chinese-English task. The corpus includes a mix of newswire, broadcast news, and webblog. We do not include HK Law, HK Hansard and UN data. The Chinese text is segmented with a segmenter trained on CTB data using conditional random fields (CRF). Our development set is the concatenation of several tuning sets (GALE Dev, P1R6 Dev, and Dev 12) initially released under the DARPA GALE program. The development set is 4491 sentences in total. Our test sets are NIST MT06 (1664 sentences) , MT08 news (691 sentences), and MT08 web (666 sentences).

For all NMT systems, the full vocabulary size of the training set is $300k$. In the training procedure, we use AdaDelta (Zeiler, 2012) to update model parameters with a mini-batch size 80. Following Mi et al. (2016a), the output vocabulary for each mini-batch or sentence is a sub-set of the full vocabulary. For each source sentence, the sentence-level target vocabularies are union of top $2k$ most frequent target words and the top 10 candidates of the word-to-word/phrase translation tables learned

single system			MT06			MT08						avg.
						News			Web			
			BP	BLEU	T-B	BP	BLEU	T-B	BP	BLEU	T-B	T-B
Tree-to-string			0.95	34.93	9.45	0.94	31.12	12.90	0.90	23.45	17.72	13.36
Cov. LVNMT (Mi et al., 2016b)			0.92	35.59	10.71	0.89	30.18	15.33	0.97	27.48	16.67	14.24
+Alignment	**Zh → En**	**A → J**	0.95	35.71	10.38	0.93	30.73	14.98	0.96	27.38	16.24	13.87
		A → T	0.95	28.59	16.99	0.92	24.09	20.89	0.97	20.48	23.31	20.40
		A → T → J	0.95	35.95	10.24	0.92	30.95	14.62	0.97	26.76	17.04	13.97
		J	0.96	36.76	9.67	0.94	31.24	14.80	0.96	28.35	**15.61**	13.36
	GDFA	**J**	0.96	36.44	10.16	0.94	30.66	15.01	0.96	26.67	16.72	13.96
	MaxEnt	**J**	0.95	36.80	**9.49**	0.93	31.74	14.02	0.96	27.53	16.21	13.24
		J + Gau.	0.96	**36.95**	9.71	0.94	**32.43**	**13.61**	0.97	**28.63**	15.80	**13.04**

Table 1: Single system results in terms of (TER-BLEU)/2 (T-B, the lower the better) on 5 million Chinese to English training set. BP denotes the brevity penalty. NMT results are on a large vocabulary ($300k$) and with UNK replaced. The second column shows different alignments (**Zh → En** (one direction), **GDFA** ("grow-diag-final-and"), and **MaxEnt** (Ittycheriah and Roukos, 2005). **A**, **T**, and **J** mean optimize alignment only, translation only, and jointly. **Gau.** denotes the smoothed transformation.

from 'fast_align' (Dyer et al., 2013). The maximum length of a source phrase is 4. In the training time, we add the reference in order to make the translation reachable.

The Cov. LVNMT system is a re-implementation of the enhanced NMT system of Mi et al. (2016a), which employs a coverage embedding model and achieves better performance over large vocabulary NMT Jean et al. (2015). The coverage embedding dimension of each source word is 100.

Following Jean et al. (2015), we dump the alignments, attentions, for each sentence, and replace UNKs with the word-to-word translation model or the aligned source word.

Our SMT system is a hybrid syntax-based tree-to-string model (Zhao and Al-onaizan, 2008), a simplified version of the joint decoding (Liu et al., 2009; Cmejrek et al., 2013). We parse the Chinese side with Berkeley parser, and align the bilingual sentences with GIZA++ and MaxEnt. and extract Hiero and tree-to-string rules on the training set. Our two 5-gram language models are trained on the English side of the parallel corpus, and on monolingual corpora (around 10 billion words from Gigaword (LDC2011T07), respectively.As suggested by Zhang (2016), NMT systems can achieve better results with the help of those monolingual corpora. In this paper, our NMT systems only use the bilingual data. We tune our system with PRO (Hopkins and May, 2011) to minimize (TER- BLEU)/2 [1] on the development set.

5.2 Translation Results

Table 1 shows the translation results of all systems. The syntax-based statistical machine translation model achieves an average (TER-BLEU)/2 of 13.36 on three test sets. The Cov. LVNMT system achieves an average (TER-BLEU)/2 of 14.24, which is about 0.9 points worse than Tree-to-string SMT system. Please note that all systems are single systems. It is highly possible that ensemble of NMT systems with different random seeds can lead to better results over SMT.

We test three different alignments:

- **Zh → En** (one direction of GIZA++),
- **GDFA** (the "grow-diag-final-and" heuristic merge of both directions of GIZA++),
- **MaxEnt** (trained on $67k$ hand-aligned sentences).

[1]The metric used for optimization in this work is (TER-BLEU)/2 to prevent the system from using sentence length alone to impact BLEU or TER. Typical SMT systems use target word count as a feature and it has been observed that BLEU can be optimized by tweaking the weighting of the target word count with no improvement in human assessments of translation quality. Conversely, in order to optimize TER shorter sentences can be produced. Optimizing the combination of metrics alleviates this effect (Arne Mauser and Ney, 2008).

The alignment quality improves from **Zh** $\rightarrow$ **En** to **MaxEnt**. We also test different optimization strategies: **J** (jointly), **A** (alignment only), and **T** (translation model only). A combination, **A** $\rightarrow$ **T**, shows that we optimize **A** only first, then we fix **A** and only update **T** part. **Gau.** denotes the smoothed transformation (Section 3.2). Only the last row uses the smoothed transformation, all others use the simple transformation.

Experimental results in Table 1 show some interesting results. First, with the same alignment, **J** joint optimization works best than other optimization strategies (lines 3 to 6). Unfortunately, breaking down the network into two separate parts (**A** and **T**) and optimizing them separately do not help (lines 3 to 5). We have to conduct joint optimization **J** in order to get a comparable or better result (lines 3, 5 and 6) over the baseline system.

Second, when we change the training alignment seeds (**Zh** $\rightarrow$ **En**, **GDFA**, and **MaxEnt**) NMT model does not yield significant different results (lines 6 to 8).

Third, the smoothed transformation (**J + Gau.**) gives some improvements over the simple transformation (the last two lines), and achieves the best result (1.2 better than LVNMT, and 0.3 better than Tree-to-string). In terms of BLEU scores, we conduct the statistical significance tests with the sign-test of Collins et al. (2005), the results show that the improvements of our **J + Gau.** over LVNMT are significant on three test sets ($p < 0.01$).

At last, the brevity penalty (BP) consistently gets better after we add the alignment cost to NMT objective. Our alignment objective adjusts the translation length to be more in line with the human references accordingly.

5.3 Alignment Results

Table 2 shows the alignment F1 scores on the alignment test set (447 hand aligned sentences). The MaxEnt model is trained on $67k$ hand-aligned sentences, and achieves an F1 score of 75.96. For NMT systems, we dump the alignment matrixes and convert them into alignments with following steps. For each target word, we sort the alphas and add the max probability link if it is higher than 0.2. If we only tune the alignment component (**A** in line 3), we improve the alignment F1 score from 45.76 to 47.87.

system			pre.	rec.	F1
MaxEnt			74.86	77.10	75.96
Cov LVNMT (Mi et al., 2016b)			51.11	41.42	45.76
+Alignment	**Zh** $\rightarrow$ **En**	**A**	50.88	45.19	47.87
		A $\rightarrow$ **J**	53.18	49.37	51.21
		A $\rightarrow$ **T**	50.29	44.90	47.44
		A $\rightarrow$ **T** $\rightarrow$ **J**	53.71	49.33	51.43
		J	**54.29**	48.02	50.97
	GDFA	**J**	53.88	48.25	50.91
	MaxEnt	**J**	44.42	55.25	49.25
		J + Gau.	48.90	**55.38**	**51.94**

Table 2: Alignment F1 scores of different models.

And we further boost the score to 50.97 by tuning alignment and translation jointly (**J** in line 7). Interestingly, the system using **MaxEnt** produces more alignments in the output, and results in a higher recall. This suggests that using **MaxEnt** can lead to a sharper attention distribution, as we pick the alignment links based on the probabilities of attentions, the sharper the distribution is, more links we can pick. We believe that a sharp attention distribution is a great property of NMT.

Again, the best result is **J + Gau.** in the last row, which significantly improves the F1 by 5 points over the baseline Cov. LVNMT system. When we use **MaxEnt** alignments, **J + Gau.** smoothing gives us about 1.7 points gain over **J** system. So it looks interesting to run another **J + Gau.** over **GDFA** alignment.

Together with the results in Table 1, we conclude that adding the alignment cost to the training objective helps both translation and alignment significantly.

6 Conclusion

In this paper, we utilize the "supervised" alignments, and put the alignment cost to the NMT objective function. In this way, we directly optimize the attention model in a supervised way. Experiments show significant improvements in both translation and alignment tasks over a very strong LVNMT system.

Acknowledgment

We thank the anonymous reviewers for their useful comments.

References

Sasa Hasan Arne Mauser and Hermann Ney. 2008. Automatic evaluation measures for statistical machine translation system optimization. In *Proceedings of LREC 2008*, Marrakech, Morocco, may.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv e-prints*, September.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based joint training for bidirectional attention-based neural machine translation. In *Proceedings of IJCAI*, New York, USA, July.

Martin Cmejrek, Haitao Mi, and Bowen Zhou. 2013. Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 545–555, Seattle, Washington, USA, October. Association for Computational Linguistics.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *HLT '05: Proceedings of the HLT and EMNLP*, pages 89–96.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*, pages 1–10, Beijing, China, July.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proceedings of IJCAI 2016*, pages 2852–2858, New York, NY, USA, July.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *North American Association for Computational Linguistics (NAACL)*, pages 104–111.

P. Liang, D. Klein, and M. I. Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint decoding with multiple translation models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 576–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016a. A coverage embedding model for neural machine translation. *ArXiv e-prints*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016b. Vocabulary manipulation for neural machine translation. In *Proceedings of ACL*, Berlin, Germany, August.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 440–447, Stroudsburg, PA, USA. Association for Computational Linguistics.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. 2016. Coverage-based Neural Machine Translation. *ArXiv e-prints*, January.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*.

Jiajun Zhang. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP 2016*, Austin, Texas, USA, November.

Bing Zhao and Yaser Al-onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 572–581, Stroudsburg, PA, USA. Association for Computational Linguistics.

Learning principled bilingual mappings of word embeddings while preserving monolingual invariance

Mikel Artetxe, Gorka Labaka, Eneko Agirre
IXA NLP Group, University of the Basque Country (UPV/EHU)
{mikel.artetxe, gorka.labaka, e.agirre}@ehu.eus

Abstract

Mapping word embeddings of different languages into a single space has multiple applications. In order to map from a source space into a target space, a common approach is to learn a linear mapping that minimizes the distances between equivalences listed in a bilingual dictionary. In this paper, we propose a framework that generalizes previous work, provides an efficient exact method to learn the optimal linear transformation and yields the best bilingual results in translation induction while preserving monolingual performance in an analogy task.

1 Introduction

Bilingual word embeddings have attracted a lot of attention in recent times (Zou et al., 2013; Kočiský et al., 2014; Chandar A P et al., 2014; Gouws et al., 2014; Gouws and Søgaard, 2015; Luong et al., 2015; Wick et al., 2016). A common approach to obtain them is to train the embeddings in both languages independently and then learn a mapping that minimizes the distances between equivalences listed in a bilingual dictionary. The learned transformation can also be applied to words missing in the dictionary, which can be used to induce new translations with a direct application in machine translation (Mikolov et al., 2013b; Zhao et al., 2015).

The first method to learn bilingual word embedding mappings was proposed by Mikolov et al. (2013b), who learn the linear transformation that minimizes the sum of squared Euclidean distances for the dictionary entries. Subsequent work has proposed alternative optimization objectives to learn better mappings. Xing et al. (2015) incorporate length normalization in the training of word embeddings and try to maximize the cosine similarity instead, introducing an orthogonality constraint to preserve the length normalization after the projection. Faruqui and Dyer (2014) use canonical correlation analysis to project the embeddings in both languages to a shared vector space.

Beyond linear mappings, Lu et al. (2015) apply deep canonical correlation analysis to learn a non-linear transformation for each language. Finally, additional techniques have been used to address the hubness problem in Mikolov et al. (2013b), both through the neighbor retrieval method (Dinu et al., 2015) and the training itself (Lazaridou et al., 2015). We leave the study of non-linear transformation and other additions for further work.

In this paper, we propose a general framework to learn bilingual word embeddings. We start with a basic optimization objective (Mikolov et al., 2013b) and introduce several meaningful and intuitive constraints that are equivalent or closely related to previously proposed methods (Faruqui and Dyer, 2014; Xing et al., 2015). Our framework provides a more general view of bilingual word embedding mappings, showing the underlying connection between the existing methods, revealing some flaws in their theoretical justification and providing an alternative theoretical interpretation for them. Our experiments on an existing English-Italian word translation induction and an English word analogy task give strong empirical evidence in favor of our theoretical reasoning, while showing that one of our models clearly outperforms previous alternatives.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2289–2294,
Austin, Texas, November 1-5, 2016.

2 Learning bilingual mappings

Let X and Z denote the word embedding matrices in two languages for a given bilingual dictionary so that their ith row X_{i*} and Z_{i*} are the word embeddings of the ith entry in the dictionary. Our goal is to find a linear transformation matrix W so that XW best approximates Z, which we formalize minimizing the sum of squared Euclidean distances following Mikolov et al. (2013b):

$$\arg\min_W \sum_i \|X_{i*}W - Z_{i*}\|^2$$

Alternatively, this is equivalent to minimizing the (squared) Frobenius norm of the residual matrix:

$$\arg\min_W \|XW - Z\|_F^2$$

Consequently, W will be the so called least-squares solution of the linear matrix equation $XW = Z$. This is a well-known problem in linear algebra and can be solved by taking the Moore-Penrose pseudoinverse $X^+ = \left(X^T X\right)^{-1} X^T$ as $W = X^+ Z$, which can be computed using SVD.

2.1 Orthogonality for monolingual invariance

Monolingual invariance is needed to preserve the dot products after mapping, avoiding performance degradation in monolingual tasks (e.g. analogy). This can be obtained requiring W to be an orthogonal matrix ($W^T W = I$). The exact solution under such orthogonality constraint is given by $W = VU^T$, where $Z^T X = U\Sigma V^T$ is the SVD factorization of $Z^T X$ (cf. Appendix A). Thanks to this, the optimal transformation can be efficiently computed in linear time with respect to the vocabulary size. Note that orthogonality enforces an intuitive property, and as such it could be useful to avoid degenerated solutions and learn better bilingual mappings, as we empirically show in Section 3.

2.2 Length normalization for maximum cosine

Normalizing word embeddings in both languages to be unit vectors guarantees that all training instances contribute equally to the optimization goal. As long as W is orthogonal, this is equivalent to maximizing the sum of cosine similarities for the dictionary entries, which is commonly used for similarity computations:

$$\begin{aligned}&\arg\min_W \sum_i \left\| \frac{X_{i*}}{\|X_{i*}\|} W - \frac{Z_{i*}}{\|Z_{i*}\|} \right\|^2 \\ &= \arg\max_W \sum_i \cos\left(X_{i*}W, Z_{i*}\right)\end{aligned}$$

This last optimization objective coincides with Xing et al. (2015), but their work was motivated by an hypothetical inconsistency in Mikolov et al. (2013b), where the optimization objective to learn word embeddings uses dot product, the objective to learn mappings uses Euclidean distance and the similarity computations use cosine. However, the fact is that, as long as W is orthogonal, optimizing the squared Euclidean distance of length-normalized embeddings is equivalent to optimizing the cosine, and therefore, the mapping objective proposed by Xing et al. (2015) is equivalent to that used by Mikolov et al. (2013b) with orthogonality constraint and unit vectors. In fact, our experiments show that orthogonality is more relevant than length normalization, in contrast to Xing et al. (2015), who introduce orthogonality only to ensure that unit length is preserved after mapping.

2.3 Mean centering for maximum covariance

Dimension-wise mean centering captures the intuition that two randomly taken words would not be expected to be semantically similar, ensuring that the expected product of two random embeddings in any dimension and, consequently, their cosine similarity, is zero. As long as W is orthogonal, this is equivalent to maximizing the sum of dimension-wise covariance for the dictionary entries:

$$\begin{aligned}&\arg\min_W \|C_m X W - C_m Z\|_F^2 \\ &= \arg\max_W \sum_i \operatorname{cov}\left(XW_{*i}, Z_{*i}\right)\end{aligned}$$

where C_m denotes the centering matrix

This equivalence reveals that the method proposed by Faruqui and Dyer (2014) is closely related to our framework. More concretely, Faruqui and Dyer (2014) use Canonical Correlation Analysis (CCA) to project the word embeddings in both languages to a shared vector space. CCA maximizes

the dimension-wise covariance of both projections (which is equivalent to maximizing the covariance of a single projection if the transformations are constrained to be orthogonal, as in our case) but adds an implicit restriction to the two mappings, making different dimensions have the same variance and be uncorrelated among themselves[1]:

$$\arg\max_{A,B} \sum_i \mathrm{cov}\left(XA_{*i}, ZB_{*i}\right)$$
$$\text{s.t.} \quad A^T X^T C_m X A = B^T Z^T C_m Z B = I$$

Therefore, the only fundamental difference between both methods is that, while our model enforces monolingual invariance, Faruqui and Dyer (2014) do change the monolingual embeddings to meet this restriction. In this regard, we think that the restriction they add could have a negative impact on the learning of the bilingual mapping, and it could also degrade the quality of the monolingual embeddings. Our experiments (cf. Section 3) show empirical evidence supporting this idea.

3 Experiments

In this section, we experimentally test the proposed framework and all its variants in comparison with related methods. For that purpose, we use the translation induction task introduced by Mikolov et al. (2013b), which learns a bilingual mapping on a small dictionary and measures its accuracy on predicting the translation of new words. Unfortunately, the dataset they use is not public. For that reason, we use the English-Italian dataset on the same task provided by Dinu et al. (2015)[2]. The dataset contains monolingual word embeddings trained with the word2vec toolkit using the CBOW method with negative sampling (Mikolov et al., 2013a)[3]. The English embeddings were trained on a 2.8 billion word corpus (ukWaC + Wikipedia + BNC), while the 1.6 billion word corpus itWaC was used to train the Italian embeddings. The dataset also contains a bilingual dictionary learned from Europarl, split into a training set of 5,000 word pairs and a test set of 1,500 word pairs, both of them uniformly distributed in frequency bins. Accuracy is the evaluation measure.

Apart from the performance of the projected embeddings in bilingual terms, we are also interested in the monolingual quality of the source language embeddings after the projection. For that purpose, we use the word analogy task proposed by Mikolov et al. (2013a), which measures the accuracy on answering questions like "what is the word that is similar to *small* in the same sense as *biggest* is similar to *big*?" using simple word vector arithmetic. The dataset they use consists of 8,869 semantic and 10,675 syntactic questions of this type, and is publicly available[4]. In order to speed up the experiments, we follow the authors and perform an approximate evaluation by reducing the vocabulary size according to a frequency threshold of 30,000 (Mikolov et al., 2013a). Since the original embeddings are the same in all the cases and it is only the transformation that is applied to them that changes, this affects all the methods in the exact same way, so the results are perfectly comparable among themselves. With these settings, we obtain a coverage of 64.98%.

We implemented the proposed method in Python using NumPy, and make it available as an open source project[5]. The code for Mikolov et al. (2013b) and Xing et al. (2015) is not publicly available, so we implemented and tested them as part of the proposed framework, which only differs from the original systems in the optimization method (exact solution instead of gradient descent) and the length normalization approach in the case of Xing et al. (2015) (postprocessing instead of constrained training). As for the method by Faruqui and Dyer (2014), we used their original implementation in Python and MATLAB[6], which we extended to cover cases where the dictionary contains more than one entry for the same word.

[1] While CCA is typically defined in terms of correlation (thus its name), correlation is invariant to the scaling of variables, so it is possible to constrain the canonical variables to have a fixed variance, as we do, in which case correlation and covariance become equivalent

[2] http://clic.cimec.unitn.it/~georgiana.dinu/down/

[3] The context window was set to 5 words, the dimension of the embeddings to 300, the sub-sampling to 1e-05 and the number of negative samples to 10

[4] https://code.google.com/archive/p/word2vec/

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

[6] https://github.com/mfaruqui/crosslingual-cca

	EN-IT	EN AN.
Original embeddings	-	76.66%
Unconstrained mapping	34.93%	73.80%
+ length normalization	33.80%	73.61%
+ mean centering	38.47%	73.71%
Orthogonal mapping	36.73%	76.66%
+ length normalization	36.87%	76.66%
+ mean centering	39.27%	76.59%

Table 1: Our results in bilingual and monolingual tasks.

	EN-IT	EN AN.
Original embeddings	-	76.66%
Mikolov et al. (2013b)	34.93%	73.80%
Xing et al. (2015)	36.87%	76.66%
Faruqui and Dyer (2014)	37.80%	69.64%
Our method	39.27%	76.59%

Table 2: Comparison of our method to other work.

3.1 Results of our framework

The rows in Table 1 show, respectively, the results for the original embeddings, the basic mapping proposed by Mikolov et al. (2013b) (cf. Section 2) and the addition of orthogonality constraint (cf. Section 2.1), with and without length normalization and, incrementally, mean centering. In all the cases, length normalization and mean centering were applied to all embeddings, even if missing from the dictionary.

The results show that the orthogonality constraint is key to preserve monolingual performance, and it also improves bilingual performance by enforcing a relevant property (monolingual invariance) that the transformation to learn should intuitively have. The contribution of length normalization alone is marginal, but when followed by mean centering we obtain further improvements in bilingual performance without hurting monolingual performance.

3.2 Comparison to other work

Table 2 shows the results for our best performing configuration in comparison to previous work. As discussed before, (Mikolov et al., 2013b) and (Xing et al., 2015) were implemented as part of our framework, so they correspond to our uncostrained mapping with no preprocessing and orthogonal mapping with length normalization, respectively.

As it can be seen, the method by Xing et al. (2015) performs better than that of Mikolov et al. (2013b) in the translation induction task, which is in line with what they report in their paper. Moreover, thanks to the orthogonality constraint their monolingual performance in the word analogy task does not degrade, whereas the accuracy of Mikolov et al. (2013b) drops by 2.86% in absolute terms with respect to the original embeddings.

Since Faruqui and Dyer (2014) take advantage of CCA to perform dimensionality reduction, we tested several values for it and report the best (180 dimensions). This beats the method by Xing et al. (2015) in the bilingual task, although it comes at the price of a considerable degradation in monolingual quality.

In any case, it is our proposed method with the orthogonality constraint and a global preprocessing with length normalization followed by dimension-wise mean centering that achieves the best accuracy in the word translation induction task. Moreover, it does not suffer from any considerable degradation in monolingual quality, with an anecdotal drop of only 0.07% in contrast with 2.86% for Mikolov et al. (2013b) and 7.02% for Faruqui and Dyer (2014).

When compared to Xing et al. (2015), our results in Table 1 reinforce our theoretical interpretation for their method (cf. Section 2.2), as it empirically shows that its improvement with respect to Mikolov et al. (2013b) comes solely from the orthogonality constraint, and not from solving any inconsistency.

It should be noted that the implementation by Faruqui and Dyer (2014) also length-normalizes the word embeddings in a preprocessing step. Following the discussion in Section 2.3, this means that our best performing configuration is conceptually very close to the method by Faruqui and Dyer (2014), as they both coincide on maximizing the average dimension-wise covariance and length-normalize the embeddings in both languages first, the only difference being that our model enforces monolingual invariance after the normalization while theirs does change the monolingual embeddings to make different dimensions have the same variance and be uncorrelated among themselves. However, our model performs considerably better than any configuration from Faruqui and Dyer (2014) in both the monolingual and the bilingual task, supporting our hypothesis that these two constraints that are implicit in their method are not only conceptually confusing,

but also have a negative impact.

4 Conclusions

This paper develops a new framework to learn bilingual word embedding mappings, generalizing previous work and providing an efficient exact method to learn the optimal transformation. Our experiments show the effectiveness of the proposed model and give strong empirical evidence in favor of our reinterpretation of Xing et al. (2015) and Faruqui and Dyer (2014). It is the proposed method with the orthogonality constraint and a global preprocessing with length normalization and dimension-wise mean centering that achieves the best overall results both in monolingual and bilingual terms, surpassing those previous methods. In the future, we would like to study non-linear mappings (Lu et al., 2015) and the additional techniques in (Lazaridou et al., 2015).

Acknowledgments

This research was partially supported by the European Commision (QTLeap FP7-ICT-2013-10-610516), a Google Faculty Award, and the Spanish Ministry of Economy and Competitiveness (TADEEP TIN2015-70214-P). Mikel Artetxe enjoys a doctoral grant from the Spanish Ministry of Education, Culture and Sports.

A Proof of solution under orthogonality

Constraining W to be orthogonal ($W^TW = I$), the original minimization problem can be reformulated as follows (cf. Section 2.1):

$$\begin{aligned}
&\arg\min_W \sum_i \|X_{i*}W - Z_{i*}\|^2 \\
&= \arg\min_W \sum_i \left(\|X_{i*}W\|^2 + \|Z_{i*}\|^2 - 2X_{i*}WZ_{i*}^T\right) \\
&= \arg\max_W \sum_i X_{i*}WZ_{i*}^T \\
&= \arg\max_W \operatorname{Tr}\left(XWZ^T\right) \\
&= \arg\max_W \operatorname{Tr}\left(Z^TXW\right)
\end{aligned}$$

In the above expression, $\operatorname{Tr}(\cdot)$ denotes the trace operator (the sum of all the elements in the main diagonal), and the last equality is given by its cyclic property. At this point, we can take the SVD of Z^TX as $Z^TX = U\Sigma V^T$, so $\operatorname{Tr}\left(Z^TXW\right) = \operatorname{Tr}\left(U\Sigma V^TW\right) = \operatorname{Tr}\left(\Sigma V^TWU\right)$. Since V^T, W and U are orthogonal matrices, their product V^TWU will also be an orthogonal matrix. In addition to that, given that Σ is a diagonal matrix, its trace after an orthogonal transformation will be maximal when the values in its main diagonal are preserved after the mapping, that is, when the orthogonal transformation matrix is the identity matrix. This will happen when $V^TWU = I$ in our case, so the optimal solution will be $W = VU^T$.

References

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems 27*, pages 1853–1861.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR2015), workshop track*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 224–229.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 270–280.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Min-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, pages 151–159.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Michael Wick, Pallika Kanani, and Adam Pocock. 2016. Minimally-constrained multilingual embeddings via artificial code-switching. In *Thirtieth AAAI conference on Artificial Intelligence (AAAI)*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1527–1536.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Measuring the behavioral impact of machine translation quality improvements with A/B testing

Benjamin Russell and **Duncan Gillespie**
Etsy
{brussell, dgillespie}@etsy.com

Abstract

In this paper we discuss a process for quantifying the behavioral impact of a domain-customized machine translation system deployed on a large-scale e-commerce platform. We discuss several machine translation systems that we trained using aligned text from product listing descriptions written in multiple languages. We document the quality improvements of these systems as measured through automated quality measures and crowdsourced human quality assessments. We then measure the effect of these quality improvements on user behavior using an automated A/B testing framework. Through testing we observed an increase in key e-commerce metrics, including a significant increase in purchases.

1 Introduction

Quality evaluation is an essential task when training a machine translation (MT) system. While automatic evaluation methods like BLEU (Papineni et al., 2002) can be useful for estimating translation quality, a higher score is no guarantee of quality improvement (Callison-Burch et al., 2006). Previous studies (e.g. Coughlin, 2003) have compared human evaluations of MT to metrics like BLEU and found close correspondence between the two. Koehn (2004) argued that relatively small differences in BLEU can indicate significant MT quality differences and suggested that human evaluation, the traditional alternative to automated metrics like BLEU, is therefore unnecessarily time-consuming and costly. Callison-Burch (2009) explored the use of crowdsourcing platforms for evaluating MT quality, with good results. However, we are not aware of any research that investigates the effect of improved MT on human behavior. In a commercial application, like an e-commerce platform, it is desirable to have a high degree of confidence in the material effect of MT quality differences: any MT system change should positively impact user experiences.

Etsy is an online marketplace for handmade and vintage items, with over 40 million active listings and a community of buyers and sellers located around the world. Visitors can use MT to translate the text of product descriptions, product reviews, and private messages, making it possible for members to communicate effectively with one another, even when they speak different languages. These multilingual interactions facilitated by MT, such as reading nonnative listing descriptions or conversing with a foreign seller, are integral to the user experience.

However, due to the unique nature of the products available in the marketplace, a generic third party MT system[1] often falls short when translating user-generated content. One challenging lexical item is "clutch." A generic engine, trained on commonly available parallel text, translates clutch as an "automotive clutch." In this marketplace, however, clutch almost always means "purse." A mistake like this is problematic: a user who sees this incorrect machine translation may lose confidence in that listing and possibly in the marketplace as a whole.

[1] We use Microsoft's Bing Translator for our machine translations.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2295–2299,
Austin, Texas, November 1-5, 2016.

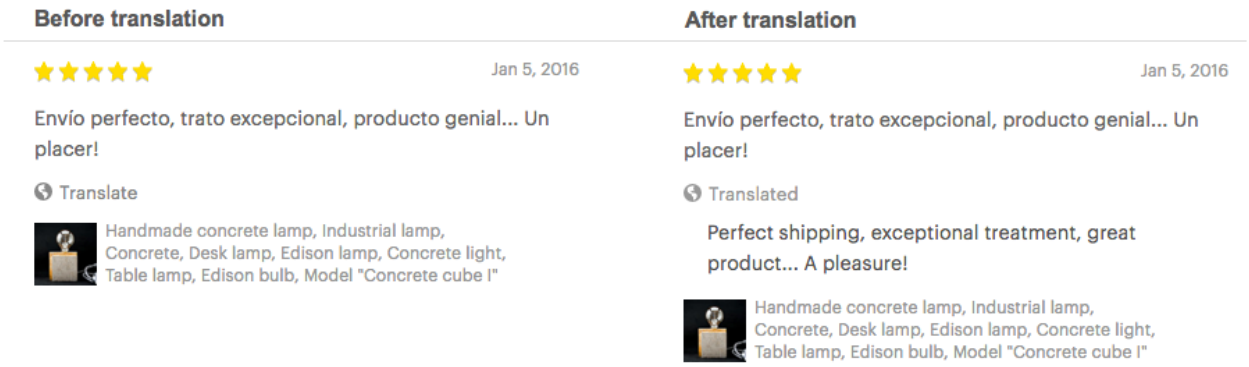


Figure 1: An example review translation on the website.

To improve the translation quality for terms like clutch, we used an interface provided by a third party machine translation service[2] to train a custom MT engine for English to French translations. To validate that the retrained MT systems were materially improved, we used a two step validation process, first using crowd-sourced evaluations with Amazon's Mechanical Turk, and secondly using A/B testing, a way of conducting randomized experiments on web sites, to measure the effect of the trained system on user behavior.

2 Data Collection

Our online marketplace contains millions of listing descriptions posted by tens of thousands of multilingual sellers. We conducted an MT system training using aligned texts from these product listings. We used our third-party translation service's automated retraining framework to train multiple MT systems that were specifically tuned to the marketplace's corpus. To gather this data, we used a Hadoop job to parse through 130 million active and expired product listings to find listing descriptions that were written in both English and French. Once we found these listing descriptions, we tokenized the text on sentence boundary. We removed any descriptions where there was a mismatch in the number of sentences between the source and target descriptions.

Next, we used a language detection service to ensure the source and target strings were the correct languages (source: English, target: French). After language detection, we removed all sentences where the ratio of alphabetic characters to total characters was below 70%. This 70% threshold was determined through manual assessment of the result set, and was used to eliminate strings with low numbers of alphabetic characters, such as "25.5 in x 35.5 in".

After these preliminary filtering steps, our training set consisted of 885,732 aligned sentences. To supplement the aligned text, we also collected 2,625,162 monolingual French sentences for the training. The monolingual text was parsed and cleaned in the same manner as the aligned sentences.

The commercial MT system's automatic training framework provides tools for the upload of bilingual and monolingual training data, tuning data, and testing data for customization of the underlying statistical MT system. Bilingual training data is used to modify the base translation model; monolingual data customizes the language model; the system is optimized for the tuning data; and the testing data is used to calculate a BLEU score. We trained over a dozen systems with a variety of datasets and selected the three systems that had the highest BLEU scores.

System 1 was trained using the aligned sentences, along with the 2.6 million monolingual sentences. The system was tuned using 2,500 sentences automatically separated from the training sentences by the third party's training system, and used an additional 2,500 automatically separated sentences for testing.

For System 2, we used a variation of the Gale-Church alignment algorithm (1993) to remove sentences predicted to be misaligned based on their length differences. The subject of sentence alignment in parallel texts has been researched exten-

[2] http://hub.microsofttranslator.com

Training Data	Sys. 1	Sys. 2	Sys. 3
886K aligned sentences	x		
766K aligned sentences after Gale-Church applied		x	x
2.6M monolingual segments	x	x	x
Auto tuning*	x	x	
Tuning with 2K in-domain sentences			x

Table 1: Data sets used for three MT system retrainings. *The third party's training platform automatically sets aside data to use for the parameter tuning.

sively (e.g. Brown et al., 1991; Gale and Church, 1993). Although more sophisticated methods exist (e.g. Chen, 1993; Wu, 1994; Melamed, 1996; Munteanu and Marcu, 2005), we used Gale-Church due to its relatively high accuracy and low implementation overhead. Misalignment between the same listing descriptions written in multiple languages could be caused by several factors, the most common problem being that sellers do not translate descriptions sentence for sentence from one language to the next. We detected possible misalignments in 13.5% of the original 886K aligned sentences, leaving 776K sentences to use for training System 2. We used auto-tuning and auto-testing for this engine, as we did for System 1.

System 3 was trained using the same training data as the second engine, but was tuned using 2,000 professionally-translated sentences taken from listing descriptions. Two hundred of these sentences were drawn semi-randomly to represent a general sample of listing description text; the remaining 1,800 contained terms, like "clutch," that were being mistranslated by the generic system. This system used the same automatically-generated testing data as the other two to calculate a BLEU score. Table 1 shows the training and tuning data used for the three systems.

3 Crowdsourced Evaluation

For evaluation of the trained translation systems, we generated translations of sentences drawn randomly from our monolingual English corpus (product listings that sellers had not translated into languages other than English). We excluded segments that were translated the same by both the trained and generic systems. (For System 1, 48 of 2,000 test sentences had the same translation as the generic system, for System 2 that number was 42, and for System 3 that number was 148.)

	BLEU Score	BLEU Score Improvement Over Generic System
System 1	48.16	+9.82
System 2	50.36	+12.02
System 3	46.85	+8.51

Table 2: BLEU score improvements for three translation systems over a baseline BLEU for the generic system of 38.34.

To obtain judgments about the quality of these translations, we used Mechanical Turk to obtain human evaluations of our candidate translation systems (Callison-Burch, 2009). To recruit Mechanical Turk workers with bilingual competence, we required workers to achieve at least 80% accuracy in a binary translation judgment task (workers were asked to judge whether each of 20 translations was "Good" or "Bad"; their answers were compared with those of professional translators).

Qualified workers completed a survey indicating their preference for the translation of a particular trained system compared to the generic commercial translation system. Translation pairs were presented in random order with no indication of whether a translation was produced by a human, a generic translation system, or an untrained translation system. Workers were asked to choose the better of the two translations or to indicate, "Neither is better". Workers were offered $2.00 to complete a 50-question survey. Each survey contained five hidden questions with known answers (translation pairs judged by professional translators) for quality control (we excluded responses from workers who did not answer the hidden questions with at least 80% accuracy).

4 Results

4.1 BLEU Evaluation

We used the automated BLEU calculation provided by the third-party translation service to obtain scores for each of the three translation systems.

All three systems had significant BLEU improvements after retraining, as shown in Table 2. We be-

	Trained	Generic	Neither	Ratio
Sys. 1	129 (34%)	109 (29%)	138 (36%)	1.18
Sys. 2	71 (25%)	85 (31%)	123 (44%)	0.84
Sys. 3	203 (36%)	150 (27%)	205 (37%)	1.35

Table 3: Results from crowdsourced evaluations of three translation systems. Columns labeled **Trained**, **Generic**, and **Neither** include the number of responses and percentage of total responses for each response type. The **Ratio** column shows the number of responses that favored the trained system to the number of responses that favored the generic system.

lieve System 3 has a lower BLEU score than the others because it was tuned on a different data set: the professionally-translated, in-domain sentences from product listing descriptions. This made the system's output less like the automatically-selected test set than the others, but closer, presumably, to the high-quality, low-noise tuning translations sourced from professional translators.

4.2 Crowdsourced evaluation

The crowdsourced evaluation of the three systems favored System 3. Table 3 provides a summary of the results. Neither System 1 nor System 2 showed a significant difference between selection of translations provided by the trained or untrained system: chi-squared tests did not detect a significant difference between number of responses favoring the trained system and number of responses favoring the generic system ($p = 0.1948$ and $p = 0.26$, respectively, for the two systems). However, a chi-squared test indicated a significant preference for System 3, which was chosen 35% more often than the generic system ($p = 0.0048$). Based on the crowd-sourced results, we proceeded to A/B test System 3 against the generic translation system baseline.

The lack of improvements for System 1 and System 2 detected using the crowd-sourcing methods was somewhat surprising, given the large BLEU score improvements observed for all three systems. We believe this lends further support to Callison-Burch, et al.'s (2006) critiques of BLEU as a standalone machine translation quality metric. In this case, it is possible that Systems 1 and 2 achieved high BLEU improvements due to over-fitting the training data from which the test set was drawn. We might speculate that this is due to the presence of low-quality translations from limited-bilingual sellers, or the presence of MT generated by a different online tool in some sellers' translations. By tuning the system using a high-quality, professionally-translated test set, we reduced overall BLEU but increased quality as judged by bilingual evaluators.

4.3 A/B testing

A/B testing is a strategy for comparing two different versions of a website to see which one performs better. Traditionally, one of these experiences is the existing, A, control experience, and the other experience is a new, B, variant experience. By randomly grouping users into one of the two experiences, and measuring the on-site behavior (e.g., clicks on a listing or items purchased) of each group, we can make data-driven decisions about whether new experiences are actually an improvement for our users. For our use case, the control experience is showing users content machine translated with the generic engine, and the variant experience is showing content translated with the retrained engine. A/B testing allows us to answer the following question: will users who read a product description translated by a domain-customized translation engine be more or less likely to purchase a product?

To test the effects of the quality improvement obtained, we used our in-house automated A/B testing framework to compare the behavioral effects on users who translated text using the generic engine and those who translated using System 3. Visitors to the online marketplace were randomly "bucketed" into an experimental group or a control group. Random bucketing was achieved via a hash of a user's browser ID, which allows users who return to the site during the experimental period to be bucketed consistently across visits. For visitors who requested translations from English into French, the generic system's translations were displayed to visitors in the control group, and System 3 translations were displayed to visitors in the experimental group.

The experiment ran for 66 days for a total of 88,106 visitors (43,306 control and 44,800 experimental). The key metrics tracked were pages per visit (the number of pages seen in one user session), conversion rate (the percent of visits that include at least one purchase), and add-to-cart rate (the percent of visits in which a user adds an item to their shopping cart). We observed a significant positive

Metric	Trained engine
Conversion rate	+8.72%
Visit add-to-cart rate	+2.92%
Pages per visit	+3.37%

Table 4: The trained translation system's (System 3) improvement over the generic engine on key business metrics. All differences are statistically significant ($p < 0.05$). Base rates are omitted for data privacy reasons.

effect of the trained system on all three metrics, as shown in Table 4: a 3.37% increase in pages per visit ($p = 0.00153$ 95% CI $[1.29, 5.46]$), an 8.72% increase in purchase rate ($p = 0.00513$ 95% CI $[2.61, 14.82]$), and a 2.92% increase in add-to-cart ($p = 0.04689$ 95% CI $[0.04, 5.8]$).

5 Conclusion

Numerous studies have shown that automatic machine translation quality estimates, such as BLEU, are correlated with human evaluations of translation quality. Our work shows that those improvements in translation quality can have a positive effect on user behavior in a commercial setting, as measured through conversion rate. These considerations suggest that, in domains where machine translation conveys information upon which individuals base decisions, the effort needed to gather and process data to customize a machine translation system can be worthwhile. Additionally, our experiments show A/B testing can be a valuable tool to evaluate machine translation quality. A/B testing goes beyond measuring the quality of translation improvements: it allows us to see the positive impact that quality improvements are having on users' purchase behavior in a measurable way.

References

Peter F. Brown, Jennifer C. Lai, and Robert L Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, pages 169–176.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of EACL*, volume 6, pages 249–256.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 286–295.

Stanley F. Chen. 1993. Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 9–16.

Deborah Coughlin. 2003. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*, pages 63–70.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

I Dan Melamed. 1996. A geometric approach to mapping bitext correspondence. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Dekai Wu. 1994. Aligning a parallel English-Chinese corpus statistically with lexical criteria. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, pages 80–87.

Creating a Large Benchmark for Open Information Extraction

Gabriel Stanovsky and **Ido Dagan**
Computer Science Department,
Bar-Ilan University, Ramat Gan, Israel
gabriel.satanovsky@gmail.com
dagan@cs.biu.ac.il

Abstract

Open information extraction (Open IE) was presented as an unrestricted variant of traditional information extraction. It has been gaining substantial attention, manifested by a large number of automatic Open IE extractors and downstream applications. In spite of this broad attention, the Open IE task definition has been lacking – there are no formal guidelines and no large scale gold standard annotation. Subsequently, the various implementations of Open IE resorted to small scale post-hoc evaluations, inhibiting an objective and reproducible cross-system comparison. In this work, we develop a methodology that leverages the recent QA-SRL annotation to create a first independent and large scale Open IE annotation,[1] and use it to automatically compare the most prominent Open IE systems.

1 Introduction

Open Information Extraction (Open IE) was originally formulated as a function from a document to a set of tuples indicating a semantic relation between a predicate phrase and its arguments (Banko et al., 2007). Wu and Weld (2008) further defined that an Open IE extractor should "produce one triple for every relation stated explicitly in the text, but is not required to infer implicit facts". For example, given the sentence "*John managed to open the door*" an Open IE extractor should produce the tuple *(John;* **managed to open***; the door)* but is *not* required to produce the extraction *(John;* **opened***; the door)*.

[1]Publicly available at `http://www.cs.biu.ac.il/nlp/resources/downloads`

Following this initial presentation of the task, Open IE has gained substantial and consistent attention. Many automatic extractors were created (e.g., (Fader et al., 2011; Mausam et al., 2012; Del Corro and Gemulla, 2013)) and were put to use in various downstream applications.

In spite of this wide attention, Open IE's formal definition is lacking. There are no clear guidelines as to what constitutes a valid proposition to be extracted, and subsequently there is no large scale benchmark annotation. Open IE evaluations therefore usually consist of a post-hoc manual evaluation of a small output sample.

This evaluation practice lacks in several respects: (1) Most works provide a precision oriented metric, whereas recall is often not measured, (2) the numbers are not comparable across systems, as they use different guidelines and datasets, and (3) the experiments are hard to replicate.

In this work, we aim to contribute to the standardization of Open IE evaluation by providing a large gold benchmark corpus. For that end, we first identify consensual guiding principles across prominent Open IE systems, resulting in a clearer formulation of the Open IE task. Following, we find that the recent formulation of QA-SRL (He et al., 2015) in fact subsumes these requirements for Open IE. This enables us to automatically convert the annotations of QA-SRL to a high-quality Open IE corpus of more than 10K extractions, 13 times larger than the previous largest Open IE annotation.

Finally, we automatically evaluate the performance of various Open IE systems against our corpus, using a soft matching criterion. This is the first

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2300–2305,
Austin, Texas, November 1-5, 2016.

time such a comparative evaluation is performed on a large scale gold corpus.

Future Open IE systems (and its applicative users) can use this large benchmark, along with the automatic evaluation measure, to easily compare their performance against previous baselines, alleviating the current need for ad-hoc evaluation.

2 Background

2.1 Open IE

Open Information Extraction (Open IE) was introduced as an open variant of traditional Information Extraction (Etzioni et al., 2008). As mentioned in the Introduction, its primary goal is to extract coherent propositions from a sentence, each comprising of a relation phrase and two or more argument phrases (e.g., (Barack Obama, **born in**, Hawaii)). Since its inception, Open IE has gained consistent attention, mostly used as a component within larger frameworks (Christensen et al., 2013; Balasubramanian et al., 2013).

In parallel, many Open IE extractors were developed. TextRunner (Banko et al., 2007) and WOE (Wu and Weld, 2010) take a self-supervised approach over automatically produced dependency parses. Perhaps more dominant is the rule based approach taken by ReVerb (Fader et al., 2011), OLLIE (Mausam et al., 2012), KrakeN (Akbik and Löser, 2012) and ClausIE (Del Corro and Gemulla, 2013).

Two recent systems take a semantically-oriented approach. Open IE-4[2] uses semantic role labeling to extract tuples, while Stanford Open Information Extraction (Angeli et al., 2015) uses natural logic inference to arrive at shorter, more salient, arguments.

Recently, Stanovsky et al. (2016b) presented PropS, a proposition oriented representation, obtained via conversion rules from dependency trees. Performing Open IE extraction over PropS structures is straightforward – follow the clearly marked predicated nodes to their direct arguments.

Contrary to the vast interest in Open IE, its task formulation has been largely overlooked. There are currently no common guidelines defining a valid extraction, which consequently hinders the creation of an evaluation benchmark for the task. Most Open IE extractors[3] evaluate performance by manually examining a small sample of their output. Table 1 summarizes the evaluations taken by the most prominent Open IE systems.

2.2 QA-SRL

Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) is typically perceived as answering **argument role questions**, such as *who, what, to whom, when*, or *where*, regarding a target predicate. For instance, PropBank's ARG0 for the predicate **say** answers the question "*who* **said** *something?*".

QA-SRL (He et al., 2015) suggests that answering explicit role questions is an intuitive means to solicit predicate-argument structures from non-expert annotators. Annotators are presented with a sentence in which a target predicate[4] was marked, and are requested to annotate argument role questions and corresponding answers.

Consider the sentence "*Giles Pearman, Microsoft's director of marketing, left his job*" and the target predicate **left**. The QA-SRL annotation consists of the following pairs: (1) *Who left something?* {**Giles Pearman**; **Microsoft's director of marketing**} and (2) *what did someone leave?* **his job**.[5]

He et al. assessed the validity of QA-SRL by annotating 3200 sentences from PropBank and Wikipedia, showing high agreement with the PropBank annotations. In the following section we automatically derive an Open IE benchmark from this QA-SRL annotation.

3 Creating an Open IE Benchmark

3.1 Open IE Guidelines

Before creating a generic benchmark for evaluating Open IE systems, it is first needed to obtain a clearer specification of the common task that they address. Despite some nuances, we identified the following core aspects of the Open IE task as consensual across all systems mentioned in Section 2:

[2] https://github.com/knowitall/openie

[3] Except for (Wu and Weld, 2010) who evaluated recall.

[4] Currently consisting of automatically annotated verbs.

[5] Three cases give rise to multiple answers for the same question: appositives (as illustrated in this example), co-reference ("***Jimmy Hendrix*** *played the guitar,* ***he*** *was really good at it*"), and distributive coordinations ("***Bob*** *and* ***Mary*** *were born in America*").

System	#Sentences	Genre	Metric	#Annot.	Agreement
TextRunner	400	Web	% Correct	3	-
WOE	300	Web, Wiki, News	Precision / Recall	5	-
ReVerb	500	Web	Precision / AUC	2	86%, .68 k
KrakeN	500	Web	% Correct	2	87%
Ollie	300	News, Wiki, Biology	Precision/Yield AUC	2	96%
ClauseIE	300	Web, Wiki, News	Precision/Yield	2	57% / 68% / 63%

Table 1: The post-hoc evaluation metrics taken by the different systems described in Section 2. In contrast, Stanford Open IE and PropS took an extrinsic evaluation approach.

Assertedness Extracted propositions should be asserted by the original sentence. For example, given the sentence "*Sam succeeded in convincing John*", ReVerb and ClausIE produce the extraction: *(Sam;* **succeeded in convincing***; John)*. Most Open IE systems do not attempt to recover implied embedded propositions (e.g., *(Sam;* **convinced***; John)*), but rather include matrix verbs (e.g., **succeeded**) in the predicate slot. Other elements that affect assertedness, like negations and modals, are typically included in the predicate slot as well (e.g. *(John;* **could not join***; the band)*).

Minimal propositions Open IE systems aim to "break down" a sentence into a set of small isolated propositions. Accordingly, the span of each individual proposition, and hence the span of each of its predicate and argument slots, should be as minimal as possible, as long as the original information (truth conditions) is preserved. For example, this leads to splitting distributive coordination in the sentence "*Bell distributes electronic and building products*", for which ClausIE produces: *(Bell,* **distributes***, electronic products)* and *(Bell,* **distributes***, building products)*. Having shorter entities as Open IE arguments was further found to be useful in several semantic tasks (Angeli et al., 2015; Stanovsky et al., 2015).

Completeness and open lexicon Open IE systems aim to extract all asserted propositions from a sentence. In practice, most current Open IE systems limit their scope to extracting verbal predicates, but consider all possible verbs without being bound to a pre-specified lexicon.

3.2 From QA-SRL to Open IE

SRL and Open IE have been defined with different objectives. Particularly, SRL identifies argument role labels, which is not addressed in Open IE. Yet, the two tasks overlap as they both need to recover predicate-argument structures in sentences. We now examine the above Open IE requirements and suggest that while they are only partly embedded within SRL structures, they can be fully recovered from QA-SRL.

Asserted (matrix) propositions appear in SRL as non-embedded predicates (e.g., **succeeded** in the "*Sam succeeded to convince John*"). However, SRL's predicates are grounded to a lexicon such as PropBank (Palmer et al., 2005) or FrameNet (Baker et al., 1998), which violates the *completeness and open lexicon* principle. Further, in contrast to the *minimal propositions* principle, arguments in SRL annotations are inclusive, each marked as full subtrees in a syntactic parse.

Yet, QA-SRL seems to bridge this gap between traditional SRL structures and Open IE requirements. Its predicate vocabulary is open, and its question-answer format solicits *minimal propositions*, as was found in a recent study by (Stanovsky et al., 2016a). This correlation suggests that the QA-SRL methodology is in fact also an attractive means for soliciting Open IE extractions from non-experts annotators. Evidently, it enables automatically deriving high quality Open IE annotations from (current or future) QA-SRL gold annotations, as described in the following section

3.3 Generating Open-IE Extractions

Formally, we extract an Open-IE dataset from the QA-SRL dataset by the following algorithm, which is illustrated in more detail further below:

1. Given:
   - s - a sentence from the QA-SRL dataset.
   - p - a predicate in s.
   - $\{q_1, ..., q_n\}$ - a list of questions over p.
   - $\{\{a_{1,1}, ..., a_{1,l_1}\}, ...\{a_{n,1}, ..., a_{n,l_n}\}\}$ - a list of sets of corresponding answers, where question q_i has l_i answers.
2. If p is a non-embedded (matrix) verb:
   (a) Remove answers which are composed only of pronouns, as these are not expected to be extracted by Open-IE (and accordingly adjust the l_i's).
   (b) Return extractions composed of p and every combination of answers in $\{\{a_{1,1}, ..., a_{1,l_1}\} \times ... \times \{a_{n,1}, ..., a_{n,l_n}\}\}$ (the Cartesian product of the answers). This process results in a list of $l_1 \cdot l_2 \cdot ... \cdot l_n$ Open IE extractions.

For example, consider the sentence: "*Barack Obama, the U.S. president, was* **determined** *to win the majority vote in Washington and Arizona*". The questions corresponding to the predicate **determine** are: {*who was determined?*, *what was someone determined to do?*}, and the corresponding answer sets are: {{"Barack Obama", "the U.S president"}, {"win the majority vote in Washington", "win the majority vote in Arizona"}}.

Following, our algorithm will produce these Open IE extractions: *(Barack Obama;* **was determined***; to win the majority vote in Washington)*, *(the U.S. president;* **was determined***; to win the majority vote in Washington)*, *(Barack Obama;* **was determined***; to win the majority vote in Arizona)*, and *(the U.S. president;* **was determined***; to win the majority vote in Arizona)*.

Note that we do not produce extractions for embedded predicates (e.g., **win**) to conform with the *assertedness* principle, as discussed earlier.

With respect to pronoun removal (step 2(a)), we would remove the pronoun "he" as the answer to the question *who was tired?* in "*John went home,* ***he*** *was tired*". Notice that in this sentence "John" would be a second answer for the above question, yielding the extraction *(John;* ***was tired****)*. When the only answer to a question is a pronoun this question will be ignored in the extraction process, since the QA-SRL corpus does not address cross-sentence co-references. This issue may be addressed in future work.

Corpus	WSJ	WIKI	ALL
#Sentences	1241	1959	3200
#Predicates	2020	5690	7710
#Questions	8112	10798	18910
#Extractions	**4481**	**5878**	**10359**

Table 2: Corpus statistics.

System	#Extractions		
	WSJ	WIKI	ALL
Stanford	6423	14104	20527
ClausIE	5295	8265	13560
Open IE4	3634	5113	8747
OLLIE	2976	5250	8226
PropS	2852	4990	7842
ReVerb	1624	2552	4716

Table 3: The yield of the different Open IE systems.

Applying this process to the QA-SRL corpus yielded a total of 10,359 Open IE extractions over 3200 sentences from 2 domains (see Table 2). This corpus is about 13 times larger than the previous largest annotated Open IE corpus (Fader et al., 2011). The corpus is available at: `http://www.cs.biu.ac.il/nlp/resources/downloads`.

Corpus validation We assess the validity of our dataset by performing expert annotation[6] of Open IE extractions, following the principles discussed in Section 3.1, for 100 random sentences. We find that our benchmark extractions, derived automatically from QA-SRL, highly agree with the expert annotation, reaching 95.8 F1 by the head-agreement criterion defined in the next section.

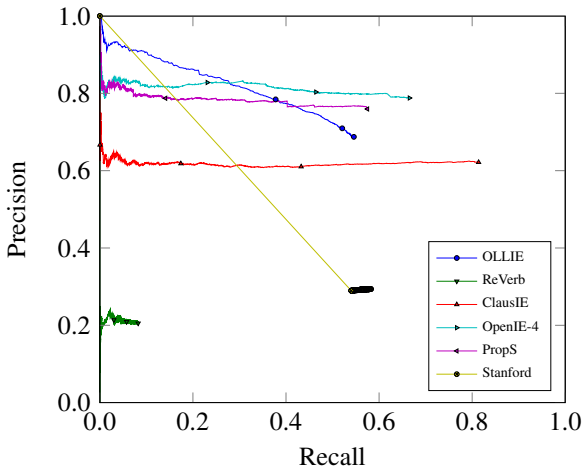


Figure 1: Precision-recall curve for the different Open IE systems on our corpus (see discussion in Section 4).

4 Comparative Evaluation

In this section, we illustrate the utility of our new corpus by testing the performance of 6 prominent Open IE systems: OpenIE-4, ClausIE, OLLIE, PropS, Stanford, and ReVerb (see Section 2).[7]

In order to evaluate these systems in terms of precision and recall, we need to match between their automated extractions and the benchmark extractions. To allow some flexibility (e.g., omissions of prepositions or auxiliaries), we follow (He et al., 2015) and match an automated extraction with a gold proposition if both agree on the grammatical head of all of their elements (predicate and arguments). We then analyze the recall and precision of Open IE systems on different confidence thresholds (Figure 1). Furthermore, we calculate the area under the PR curve for each of the different corpora (Figure 2) and the explicit yield per system (Table 3).

To the best of our knowledge, this is the first objective comparative evaluation of prominent Open-IE systems, over a large and independently created dataset. This comparison gives rise to several observations; which can be useful for future research and for choosing a preferred system for a particular application setting, such as:

[6]Carried by the first author.

[7]Currently, we test only the common case of verbal predicates.

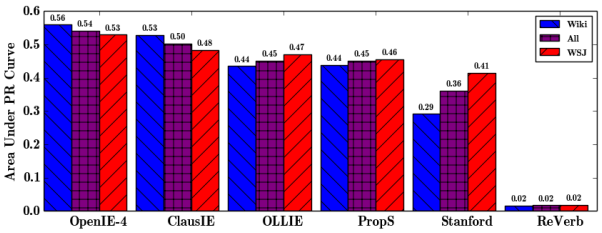


Figure 2: Area Under the PR Curve (AUC) measure for the evaluated systems.

1. *Open IE-4* achieves best precision above 3% recall ($\geqslant$ 78.67) and best AUC score (54.02),
2. *ClausIE* is best at recall (81.38), and
3. *Stanford Open IE* assigns confidence of 1 to 94% of its extractions, explaining its low precision.

5 Conclusions

We presented the first independent and large scale Open IE benchmark annotation, and tested the most prominent systems against it. We hope that future Open IE systems can make use of this new resource to easily and objectively measure and compare their performance.

Acknowledgments

We would like to thank Mausam for fruitful discussions, and the anonymous reviewers for their helpful comments.

This work was supported in part by grants from the MAGNET program of the Israeli Office of the Chief Scientist (OCS), the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *NAACL-HLT 2012: Proceedings of the The Knowledge Extraction Workshop*.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL*, pages 86–90. Association for Computational Linguistics.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*, pages 1721–1731.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CONLL*, pages 152–164.

Janara Christensen, Stephen Soderland Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *HLT-NAACL*, pages 1163–1173. Citeseer.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. International World Wide Web Conferences Steering Committee.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the Web. *Communications of the ACM*, 51(12):68–74.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea, July. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Gabriel Stanovsky, Ido Dagan, and Meni Adler. 2016a. Specifying and annotating reduced argument span via qa-srl. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016b. Getting more out of syntax with props. *arXiv preprint*.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden, July. Association for Computational Linguistics.

Fei Wu, Raphael Hoffmann, and Daniel S Weld. 2008. Information extraction from wikipedia: Moving down the long tail. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 731–739. ACM.

Bilingually-constrained Synthetic Data for Implicit Discourse Relation Recognition

Changxing Wu[1,2], **Xiaodong Shi**[1,2]*, **Yidong Cheng**[1,2], **Yanzhou Huang**[1,2], **Jinsong Su**[3]
Fujian Key Lab of the Brain-like Intelligent Systems, Xiamen University, China[1]
School of Information Science and Technology, Xiamen University, China[2]
School of Software, Xiamen University, China[3]
{wcxnlp, huangyanzhou163}@163.com
{mandel, ydchen, jssu}@xmu.edu.cn

Abstract

To alleviate the shortage of labeled data, we propose to use bilingually-constrained synthetic implicit data for implicit discourse relation recognition. These data are extracted from a bilingual sentence-aligned corpus according to the implicit/explicit mismatch between different languages. Incorporating these data via a multi-task neural network model achieves significant improvements over baselines, on both the English PDTB and Chinese CDTB data sets.

1 Introduction

Discovering the discourse relation between two sentences is crucial to understanding the meaning of a coherent text, and also beneficial to many downstream NLP applications, such as question answering and machine translation. Implicit discourse relation recognition (DRR_{imp}) remains a challenging task due to the absence of strong surface clues like discourse connectives (e.g. but). Most work resorts to large amounts of manually designed features (Soricut and Marcu, 2003; Pitler et al., 2009; Lin et al., 2009; Louis et al., 2010; Rutherford and Xue, 2014), or distributed features learned via neural network models (Braud and Denis, 2015; Zhang et al., 2015; Ji and Eisenstein, 2015). The above methods usually suffer from limited labeled data.

Marcu and Echihabi (2002) attempt to create labeled implicit data automatically by removing connectives from explicit instances, as additional training data. These data are usually called as *synthetic implicit data* (hereafter $SynData$). However, Sporleder and Lascarides (2008) argue that $SynData$ has two drawbacks: 1) meaning shifts in some cases when removing connectives, and 2) a different word distribution with the *real implicit data*. They also show that using $SynData$ directly degrades the performance. Recent work seeks to derive valuable information from $SynData$ while filtering noise, via domain adaptation (Braud and Denis, 2014; Ji et al., 2015), classifying connectives (Rutherford and Xue, 2015) or multi-task learning (Lan et al., 2013; Liu et al., 2016), and shows promising results.

ch: [社会 认为 有 青少年 问题,]$_{Arg1}$
society reckon existence youth problems,
implicit=但是 [很多 青少年 认为 自己 没 问题.]$_{Arg2}$
but many young people think themselves no problems.

en: [society reckons the existence of youth problems,]$_{Arg1}$
but [many young people do not think there is anything wrong with them.]$_{Arg2}$

Figure 1: An example illustrating the implicit/explicit mismatch between Chinese (ch) and English (en). A Chinese implicit instance is translated into an English explicit one. In the PDTB, a discourse instance is defined as a connective (e.g. but) taking two arguments ($Arg1$ and $Arg2$).

Different from previous work, we propose to construct *bilingually-constrained synthetic implicit data* (called $BiSynData$) for DRR_{imp}, which can alleviate the drawbacks of $SynData$. Our method is inspired by the findings that a discourse instance expressed implicitly in one language may be expressed explicitly in another. For example, Zhou and Xue

*Corresponding author.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2306–2312,
Austin, Texas, November 1-5, 2016.

(2012) show that the connectives in Chinese omit much more frequently than those in English with about 82.0% vs. 54.5%. Li et al. (2014a) further argue that there are about 23.3% implicit/explicit mismatchs between Chinese/English instances. As illustrated in Figure 1, a Chinese implicit instance where the connective 但是 is absent, is translated into an English explicit one with the connective but. Intuitively, the Chinese instance is a *real* implicit one which can be signaled by but. Hence, it could potentially serve as additional training data for the Chinese DRR_{imp}, avoiding the different word distribution problem of $SynData$. Meanwhile, for the English explicit instance, it is very likely that removing but would not lose any information since its Chinese counterpart 但是 can be omitted. Therefore it could be used for the English DRR_{imp}, alleviating the meaning shift problem of $SynData$.

We extract our $BiSynData$ from a Chinese-English sentence-aligned corpus (Section 2). Then we design a multi-task neural network model to incorporate the $BiSynData$ (Section 3). Experimental results, on both the English PDTB (Prasad et al., 2008) and Chinese CDTB (Li et al., 2014b), show that $BiSynData$ is more effective than $SynData$ used in previous work (Section 4). Finally, we review the related work (Section 5) and draw conclusions (Section 6).

2 BiSynData

Formally, given a Chinese-English sentence pair (S_{ch}, S_{en}), we try to find an English explicit instance $(Arg1_{en}, Arg2_{en}, Conn_{en})$ in S_{en}[1], and a Chinese implicit instance $(Arg1_{ch}, Arg2_{ch})$ in S_{ch}, where $(Arg1_{en}, Arg2_{en}, Conn_{en})$ is the translation of $(Arg1_{ch}, Arg2_{ch})$. In most cases, discourse relations should be preserved during translating, so the connective $Conn_{en}$ is potentially a strong indicator of the discourse relation between not only $Arg1_{en}$ and $Arg2_{en}$, but also $Arg1_{ch}$ and $Arg2_{ch}$. Therefore, we can construct two synthetic implicit instances labeled by $Conn_{en}$, denoted as $\langle (Arg1_{en}, Arg2_{en}), Conn_{en} \rangle$ and $\langle (Arg1_{ch}, Arg2_{ch}), Conn_{en} \rangle$, respectively. We refer to these synthetic instances as $BiSynData$ because they are constructed according to the bilingual implicit/explicit mismatch.

Conn.	Freq.	Conn.	Freq.
and	14294	$while$	1031
if	2580	$before$	822
as	1951	$also$	552
$when$	1521	$since$	511
but	1122	$because$	503

Table 1: Top 10 most frequent connectives in our $BiSynData$.

In our experiments, we extract our $BiSynData$ from a combined corpus (FBIS and HongKong Law), with about 2.38 million Chinese-English sentence pairs. We generate 30,032 synthetic English instances and the same number of Chinese instances, with 80 connectives, as our $BiSynData$. Table 1 lists the top 10 most frequent connectives in our $BiSynData$, which are roughly consistent with the statistics of Chinese/English implicit/explicit mismatches in (Li et al., 2014a). According to connectives and their related relations in the PDTB, in most cases, and and $also$ indicate the $Expansion$ relation, if and $because$ the $Contigency$ relation, $before$ the $Temporal$ relation, and but the $Comparison$ relation. Connectives as, $when$, $while$ and $since$ are ambiguous. For example, $while$ can indicate the $Comparison$ or $Temporal$ relation. Overall, our constructed $BiSynData$ covers all four main discourse relations defined in the PDTB.

With our $BiSynData$, we define two connective classification tasks: 1) given $(Arg1_{en}, Arg2_{en})$ to predict the connective $Conn_{en}$, and 2) given $(Arg1_{ch}, Arg2_{ch})$ to predict $Conn_{en}$. We incorporate the first task to help the English DRR_{imp}, and the second for the Chinese DRR_{imp}. It is worthy to note that we use English connectives themselves as classification labels rather than mapping them to relations in both tasks.

3 Multi-Task Neural Network Model

We design a Multi-task Neural Network Model (denoted as MTN), which incorporates a connective classification task on $BiSynData$ (auxiliary task) to benefit DRR_{imp} (main task). In general, the more related two tasks are, the more powerful a multi-task learning method will be. In the current problem, the

[1]In our experiments, we use the pdtb-parser toolkit (Lin et al., 2014) to identify English explicit instances.

two tasks are essentially the same, just with different output labels. Therefore, as illustrated in Figure 2, MTN shares parameters in all feature layers (L_1-L_3) and uses two separate classifiers in the classifier layer (L_4). For each task, given an instance (Arg_1, Arg_2), MTN simply averages embeddings of words to represent arguments, as v_{Arg_1} and v_{Arg_2}. These two vectors are then concatenated and transformed through two non-linear hidden layers. Finally, the corresponding $softmax$ layer is used to perform classification.

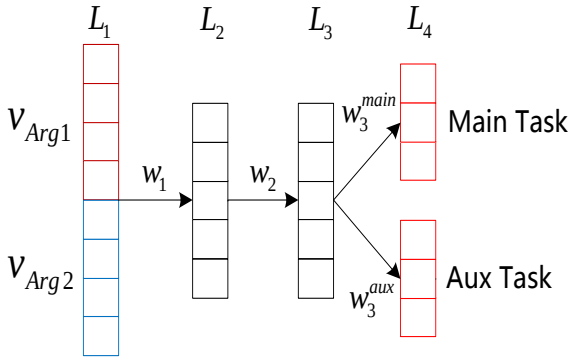


Figure 2: MTN with four layers L_1-L_4.

MTN ignores the word order in arguments and uses two hidden layers to capture the interactions between two arguments. The idea behind MTN is borrowed from (Iyyer et al., 2015), where a deep averaging network achieves close to the state-of-the-art performance on text classification. Though MTN is simple, it is easy to train and efficient on both memory and computational cost. In addition, the simplicity of MTN allows us to focus on measuring the quality of $BiSynData$.

We use the cross-entropy loss function and mini-batch AdaGrad (Duchi et al., 2011) to optimize parameters. Pre-trained word embeddings are fixed. We find that fine-tuning word embeddings during training leads to severe overfitting in our experiments. Following Liu et al. (2016), we alternately use two tasks to train the model, one task per epoch. For tasks on both the PDTB and CDTB, we use the same hyper-parameters. The dimension of word embedding is 100. We set the size of L_2 to 200, and L_3 to 100. $ReLU$ is used as the non-linear function. Different learning rates 0.005 and 0.001 are used in the main and auxiliary tasks, respectively. To avoid overfitting, we randomly drop out 20% words in each argument following Iyyer et al. (2015). All hyper-parameters are tuned on the development set.

4 Experiments

We evaluate our method on both the English PDTB and Chinese CDTB data sets. We tokenize English data and segment Chinese data using the Stanford CoreNLP toolkit (Manning et al., 2014). The English/Chinese Gigaword corpus (3rd edition) is used to train the English/Chinese word embeddings via $word2vec$ (Mikolov et al., 2013), respectively. Due to the skewed class distribution of test data (see Section 4.1), we use the macro-averaged F_1 for performance evaluation.

4.1 On the PDTB

Following Rutherford and Xue (2015), we perform a 4-way classification on the top-level discourse relations: $Temporal$ ($Temp$), $Comparison$ ($Comp$), $Contigency$ ($Cont$) and $Expansion$ ($Expa$). Sections 2-20 are used as training set, sections 0-1 as development set and sections 21-22 as test set. The training/test set contains 582/55 instances for $Temp$, 1855/145 for $Comp$, 3235/273 for $Cont$ and 6673/538 for $Expa$. The top 20 most frequent connectives in our $BiSynData$ are considered in the auxiliary task, with 28,013 synthetic English instances in total.

		STN	MTN_{bi}
$Temp$	P	33.33	**34.48**
	R	14.55	**18.18**
	F_1	20.25	**23.81**
$Comp$	P	38.54	**42.11**
	R	25.52	**33.10**
	F_1	30.71	**37.07**
$Cont$	P	38.36	**44.22**
	R	**41.03**	40.66
	F_1	39.65	**42.37**
$Expa$	P	59.60	**62.56**
	R	66.36	**71.75**
	F_1	62.80	**66.84**
$macro\ F_1$		38.35	**42.52**

Table 2: Results of 4-way classification on the PDTB.

Table 2 shows the results of MTN combining our $BiSynData$ (denoted as MTN_{bi}) on the PDTB.

STN means we train MTN with only the main task. On the $macro\ F_1$, MTN_{bi} gains an improvement of 4.17% over STN. The improvement is significant under one-tailed t-test (p<0.05). A closer look into the results shows that MTN_{bi} performs better across all relations, on the precision, recall and F_1 score, except a little drop on the recall of $Cont$. The reason for the recall drop of $Cont$ is not clear. The greatest improvement is observed on $Comp$, up to 6.36% F_1 score. The possible reason is that only $while$ is ambiguous about $Comp$ and $Temp$, while as, $when$ and $since$ are all ambiguous about $Temp$ and $Cont$, among top 10 connectives in our $BiSynData$. Meanwhile the amount of labeled data for $Comp$ is relatively small. Overall, using $BiSynData$ under our multi-task model achieves significant improvements on the English DRR_{imp}. We believe the reasons for the improvements are twofold: 1) the added synthetic English instances from our $BiSynData$ can alleviate the meaning shift problem, and 2) a multi-task learning method is helpful for addressing the different word distribution problem between implicit and explicit data.

Considering some of the English connectives (e.g., $while$) are highly ambiguous, we compare our method with ones that uses only unambiguous connectives. Specifically, we first discard as, $when$, $while$ and $since$ in top 20 connectives, and get 22,999 synthetic instances. Then, we leverage these instances in two different ways: 1) using them in our multi-task model as above, and 2) using them as additional training data directly after mapping unambiguous connectives into relations. Both methods using only unambiguous connectives do not achieve better performance. One possible reason is that these synthetic instances become more unbalanced after discarding ones with ambiguous connectives.

We also compare MTN_{bi} with recent systems using additional training data. Rutherford and Xue (2015) select explicit instances that are similar to the implicit ones via connective classification, to enrich the training data. Liu et al. (2016) use a multi-task model with three auxiliary tasks: 1) $conn$: connective classification on explicit instances, 2) exp: relation classification on the labeled explicit instances in the PDTB, and 3) rst: relation classification on the labeled RST corpus (William and Thompson, 1988), which defines different discourse relations with that in the PDTB. The results are shown in Table 3. Although Liu et al. (2016) achieve the state-of-the-art performance (Line 5), they use two additional labeled corpora. We can find that MTN_{bi} (Line 6) yields better results than those systems incorporating $SynData$ (Line 1, 2 and 3), or even the labeled RST (Line 4). These results confirm that $BiSynData$ can indeed alleviate the disadvantages of $SynData$ effectively.

	System	$macro\ F_1$
1	Rutherford and Xue (2015)	40.50
2	Liu et al. (2016) $_{conn}$	38.09
3	Liu et al. (2016) $_{exp}$	39.03
4	Liu et al. (2016) $_{rst}$	40.67
5	Liu et al. (2016) $_{conn+exp+rst}$	**44.98**
6	MTN_{bi}	42.52

Table 3: Comparison with recent systems on the PDTB. $conn+exp+rst$ means using three auxiliary tasks simultaneously.

4.2 On the CDTB

Four top-level relations are defined in the CDTB, including $Transition$ ($Tran$), $Causality$ ($Caus$), $Explanation$ ($Expl$) and $Coordination$ ($Coor$). We use instances in the first 50 documents as test set, second 50 documents as development set and remaining 400 documents as training set. We conduct a 3-way classification because of only 39 instances for $Tran$. The training/test set contains 682/95 instances for $Caus$, 1143/126 for $Expl$ and 2300/347 for $Coor$. The top 20 most frequent connectives (excluding and)[2] in our $BiSynData$ are considered in the auxiliary task, with 13,899 synthetic Chinese instances in total. The results are shown in Table 4. Compared with STN, MTN_{bi} raises the $macro\ F_1$ from 55.44% to 58.28%. The improvement is significant under one-tailed t-test (p<0.05). Therefore, $BiSynData$ is also helpful for the Chinese DRR_{imp}.

Because of no reported results on the CDTB, we use MTN with two different auxiliary tasks as baselines: 1) exp: relation classification on the labeled

[2]Including and degrades the performance slightly. A possible reason is that and can be related to both the $Expl$ and $Coor$ relations in the CDTB, and instances marked by and account for about half of our $BiSynData$.

		STN	MTN_{bi}
$Caus$	P	47.92	**52.94**
	R	24.21	**28.42**
	F_1	32.17	**36.99**
$Expl$	P	**54.62**	53.47
	R	56.35	**61.11**
	F_1	55.47	**57.04**
$Coor$	P	74.36	**78.02**
	R	83.57	**83.86**
	F_1	78.70	**80.83**
$macro\ F_1$		55.44	**58.28**

Table 4: Results of 3-way classification on the CDTB.

explicit instances in the CDTB, including 466 instances for $Caus$, 201 for $Expl$ and 974 for $Coor$. 2) $conn$: connective classification on explicit instances from the Xinhua part of the Chinese Gigaword corpus. We collect explicit instances with the top 20 most frequent Chinese connectives and sample 20,000 instances for the experiment. Both exp and $conn$ can be considered as tasks on $SynData$. The results in Table 5 show that MTN incorporating $BiSynData$ (Line 3) performs better than using $SynData$ (Line 1 and 2), for the task on the CDTB.

	System	$macro\ F_1$
1	MTN_{exp}	56.42
2	MTN_{conn}	56.86
3	MTN_{bi}	**58.28**

Table 5: MTN with different auxiliary tasks on the CDTB.

5 Related Work

One line of research related to DRR_{imp} tries to take advantage of explicit discourse data. Zhou et al. (2010) predict the absent connectives based on a language model. Using these predicted connectives as features is proven to be helpful. Biran and McKeown (2013) aggregate word-pair features that are collected around the same connectives, which can effectively alleviate the feature sparsity problem. More recently, Braud and Denis (2014) and Ji et al. (2015) consider explicit data from a different domain, and use domain adaptation methods to explore the effect of them. Rutherford and Xue (2015) propose to gather weakly labeled data from explicit instances via connective classification, which are used as additional training data directly. Lan et al. (2013) and Liu et al. (2016) combine explicit and implicit data using multi-task learning models and gain improvements. Different from all the above work, we construct additional training data from a bilingual corpus.

Multi-task neural networks have been successfully used for many NLP tasks. For example, Collobert et al. (2011) jointly train models for the Part-of-Speech tagging, chunking, named entity recognition and semantic role labeling using convolutional network. Liu et al. (2015) successfully combine the tasks of query classification and ranking for web search using a deep multi-task neural network. Luong et al. (2016) explore multi-task sequence to sequence learning for constituency parsing, image caption generation and machine translation.

6 Conclusion

In this paper, we introduce bilingually-constrained synthetic implicit data ($BiSynData$), which are generated based on the bilingual implicit/explicit mismatch, into implicit discourse relation recognition for the first time. On both the PDTB and CDTB, using $BiSynData$ as the auxiliary task significantly improves the performance of the main task. We also show that $BiSynData$ is more beneficial than the synthetic implicit data typically used in previous work. Since the lack of labeled data is a major challenge for implicit discourse relation classification, our proposed $BiSynData$ can enrich the training data and then benefit future work.

Acknowledgments

We would like to thank all the reviewers for their constructive and helpful suggestions on this paper. This work is partially supported by the Natural Science Foundation of China (Grant Nos. 61573294, 61303082, 61672440), the Ph.D. Programs Foundation of Ministry of Education of China (Grant No. 20130121110040), the Fund of Research Project of Tibet Autonomous Region of China (Grant No. Z2014A18G2-13), and the Natural Science Foundation of Fujian Province (Grant No. 2016J05161).

References

Or Biran and Kathleen McKeown. 2013. Aggregated Word Pair Features for Implicit Discourse Relation Disambiguation. In *Proceedings of ACL (Volume 2: Short Papers)*, pages 69–73, Sofia, Bulgaria.

Chloé Braud and Pascal Denis. 2014. Combining Natural and Artificial Examples to Improve Implicit Discourse Relation Identification. In *Proceedings of COLING*, pages 1694–1705, Dublin, Ireland.

Chloé Braud and Pascal Denis. 2015. Comparing Word Representations for Implicit Discourse Relation Classification. In *Proceedings of EMNLP*, pages 2201–2211, Lisbon, Portugal.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of ACL-IJCNLP*, pages 1681–1691, Beijing, China.

Yangfeng Ji and Jacob Eisenstein. 2015. One Vector is Not Enough: Entity-Augmented Distributed Semantics for Discourse Relations. In *Transactions of the Association for Computational Linguistics*, volume 3, pages 329–344.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the Gap: Domain Adaptation from Explicit to Implicit Discourse Relations. In *Proceedings of EMNLP*, pages 2219–2224, Lisbon, Portugal.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging Synthetic Discourse Data via Multi-task Learning for Implicit Discourse Relation Recognition. In *Proceedings of ACL*, pages 476–485, Sofia, Bulgaria.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014a. Cross-lingual Discourse Relation Analysis: A Corpus Study and a Semi-supervised Classification System. In *Proceedings of COLING : Technical Papers*, pages 577–587, Dublin, Ireland.

Yancui Li, Wenhe Feng, Jing Sun, Fang Kong, and Guodong Zhou. 2014b. Building Chinese Discourse Corpus with Connective-driven Dependency Tree Structure. In *Proceedings of EMNLP*, pages 2105–2114, Doha, Qatar.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing Implicit Discourse Relations in the Penn Discourse Treebank. In *Proceedings of EMNLP*, pages 343–351, PA, USA.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled End-to-end Discourse Parser. *Natural Language Engineering*, 20(02):151–184.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation Learning Using Multi-task Deep Neural Networks for Semantic Classification and Information Retrieval. In *Proceedings of NAACL*, pages 912–921, Denver, Colorado.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit Discourse Relation Classification via Multi-Task Neural Networks. In *Proceedings of AAAI*, pages 2750–2756, Arizona, USA.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using Entity Features to Classify Implicit Discourse Relations. In *Proceedings of SIGDIAL*, pages 59–62, PA, USA.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proceedings of ICLR*, pages 1–10, San Juan, Puerto Rico.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL (System Demonstrations)*, pages 55–60, Maryland, USA.

Daniel Marcu and Abdessamad Echihabi. 2002. An Unsupervised Approach to Recognizing Discourse Relations. In *Proceedings of ACL*, pages 368–375, PA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic Sense Prediction for Implicit Discourse Relations in Text. In *Proceedings of ACL-IJCNLP*, pages 683–691, PA, USA.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC*, volume 24, pages 2961–2968, Marrakech, Morocco.

Attapol T. Rutherford and Nianwen Xue. 2014. Discovering Implicit Discourse Relations Through Brown Cluster Pair Representation and Coreference Patterns. In *Proceedings of EACL*, pages 645–654, Gothenburg, Sweden.

Attapol Rutherford and Nianwen Xue. 2015. Improving the Inference of Implicit Discourse Relations via Classifying Explicit Discourse Connectives. In *Proceedings of NAACL*, pages 799–808, Denver, Colorado.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of NAACL*, pages 149–156, PA, USA.

Caroline Sporleder and Alex Lascarides. 2008. Using Automatically Labelled Examples to Classify Rhetorical Relations: An Assessment. *Natural Language Engineering*, 14(3):369–416.

Mann William and Sandra Thompson. 1988. Rhetorical structure theory: Towards a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition. In *Proceedings of EMNLP*, pages 2230–2235, Lisbon, Portugal.

Yuping Zhou and Nianwen Xue. 2012. PDTB-style discourse annotation of Chinese text. In *Proceedings of ACL*, pages 69–77, Jeju Island, Korea.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting Discourse Connectives for Implicit Discourse Relation Recognition. In *Proceedings of COLING*, pages 1507–1514, PA, USA.

Transition-Based Dependency Parsing with Heuristic Backtracking

Jacob Buckman[♣] **Miguel Ballesteros**[♢] **Chris Dyer**[♠♣]
[♣]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[♢]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♠]Google DeepMind, London, UK
jacobbuckman@cmu.edu, miguel.ballesteros@upf.edu, cdyer@google.com

Abstract

We introduce a novel approach to the decoding problem in transition-based parsing: heuristic backtracking. This algorithm uses a series of partial parses on the sentence to locate the best candidate parse, using confidence estimates of transition decisions as a heuristic to guide the starting points of the search. This allows us to achieve a parse accuracy comparable to beam search, despite using fewer transitions. When used to augment a Stack-LSTM transition-based parser, the parser shows an unlabeled attachment score of up to 93.30% for English and 87.61% for Chinese.

1 Introduction

Transition-based parsing, one of the most prominent dependency parsing techniques, constructs a dependency structure by reading words sequentially from the sentence, and making a series of local decisions (called transitions) which incrementally build the structure. Transition-based parsing has been shown to be both fast and accurate; the number of transitions required to fully parse the sentence is linear relative to the number of words in the sentence.

In recent years, the field has seen dramatic improvements in the ability to correctly predict transitions. Recent models include the greedy Stack-LSTM model of Dyer et al. (2015) and the globally normalized feed-forward networks of Andor et al. (2016). These models output a local decision at each transition point, so searching the space of possible paths to the predicted tree is an important component of high-accuracy parsers.

One common search technique is beam search. (Zhang and Clark, 2008; Zhang and Nivre, 2011; Bohnet and Nivre, 2012; Zhou et al., 2015; Weiss et al., 2015; Yazdani and Henderson, 2015) In beam-search, a fixed number of candidate transition sequences are generated, and the highest-scoring sequence is chosen as the answer. One downside to beam search is that it often results in a significant amount of wasted predictions. A constant number of beams are explored at all points throughout the sentence, leading to some unnecessary exploration towards the beginning of the sentence, and potentially insufficient exploration towards the end.

One way that this problem can be mitigated is by using a dynamically-sized beam (Mejia-Lavalle and Ramos, 2013). When using this technique, at each step, prune all beams whose scores are below some value s, where s is calculated based upon the distribution of scores of available beams. Common methods for pruning are removing all beams below some percentile, or any beams which scored below some constant percentage of the highest-scoring beam.

Another approach to solving this issue is given by Choi and McCallum (2013). They introduced selectional branching, which involves performing an initial greedy parse, and then using confidence estimates on each prediction to spawn additional beams. Relative to standard beam-search, this reduces the average number of predictions required to parse a sentence, resulting in a speed-up.

In this paper, we introduce heuristic backtracking, which expands on the ideas of selectional branching by integrating a search strategy based on a heuristic function (Pearl, 1984): a function which estimates

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2313–2318,
Austin, Texas, November 1-5, 2016.

the future cost of taking a particular decision. When paired with a good heuristic, heuristic backtracking maintains the property of reducing wasted predictions, but allows us to more fully explore the space of possible transition sequences (as compared to selectional branching). In this paper, we use a heuristic based on the confidence of transition predictions.

We also introduce a new optimization: heuristic backtracking with cutoff. Since heuristic backtracking produces results incrementally, it is possible to stop the search early if we have found an answer that we believe to be the gold parse, saving time proportional to the number of backtracks remaining.

We compare the performance of these various decoding algorithms with the Stack-LSTM parser (Dyer et al., 2015), and achieve slightly higher accuracy than beam search, in significantly less time.

2 Transition-Based Parsing With Stack-LSTM

Our starting point is the model described by Dyer et al. (2015).[1] The parser implements the arc-standard algorithm (Nivre, 2004) and it therefore makes use of a stack and a buffer. In (Dyer et al., 2015), the stack and the buffer are encoded with Stack-LSTMs, and a third sequence with the history of actions taken by the parser is encoded with another Stack-LSTM. The three encoded sequences form the parser state $\mathbf{p}_t$ defined as follows,

$$\mathbf{p}_t = \max\left\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\right\}, \quad (1)$$

where $\mathbf{W}$ is a learned parameter matrix, $\mathbf{b}_t$, $\mathbf{s}_t$ and $\mathbf{a}_t$ are the stack LSTM encoding of buffer, stack and the history of actions, and $\mathbf{d}$ is a bias term. The output $\mathbf{p}_t$ (after a component-wise rectified linear unit (ReLU) nonlinearity (Glorot et al., 2011)) is then used to compute the probability of the parser action at time t as:

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)}, \quad (2)$$

where $\mathbf{g}_z$ is a column vector representing the (output) embedding of the parser action z, and q_z is a bias term for action z. The set $\mathcal{A}(S, B)$ represents the valid transition actions that may be taken in the current state. The objective function is:

$$\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w}, \boldsymbol{z}) = \sum_{t=1}^{|\boldsymbol{z}|} \log p(z_t \mid \mathbf{p}_t) \quad (3)$$

where $\boldsymbol{z}$ refers to parse transitions.

[1] We refer to the original work for details.

3 Heuristic Backtracking

Using the Stack-LSTM parsing model of Dyer et al. (2015) to predict each decision greedily yields very high accuracy; however, it can only explore one path, and it therefore can be improved by conducting a larger search over the space of possible parses. To do this, we introduce a new algorithm, heuristic backtracking. We also introduce a novel cutoff approach to further increase speed.

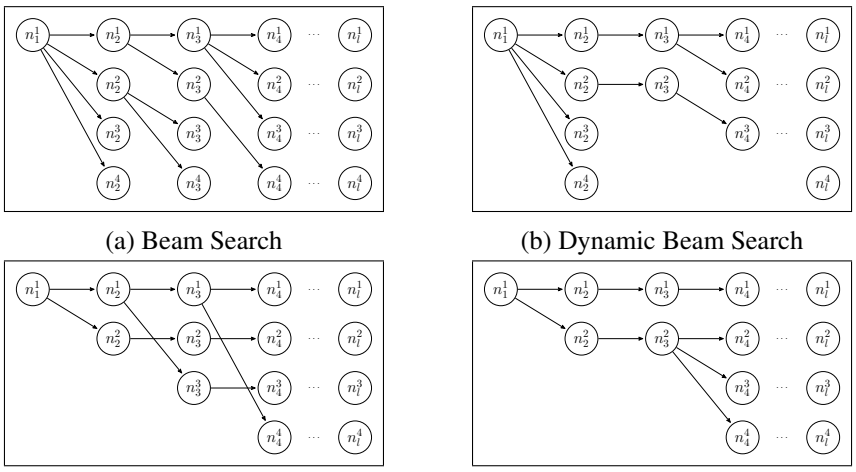
3.1 Decoding Strategy

We model the space of possible parses as a tree, where each node represents a certain parse state (with complete values for stack, buffer, and action history). Transitions connect nodes of the tree, and leaves of the tree represent final states.

During the first iteration, we start at the root of the tree, and greedily parse until we reach a leaf. That is, for each node, we use the Stack-LSTM model to calculate scores for each transition (as described in Section 2), and then execute the highest-scoring transition, generating a child node upon which we repeat the procedure. Additionally, we save an ordered list of the transition scores, and calculate the confidence of the node (as described in Section 3.2).

When we reach the leaf node, we backtrack to the location that is most likely to fix a mistake. To find this, we look at all explored nodes that still have at least one unexplored child, and choose the node with the lowest heuristic confidence (see Section 3.2). We rewind our stack, buffer, and action history to that state, and execute the highest-scoring transition from that node that has not yet been explored. At this point, we are again in a fully-unexplored node, and can greedily parse just as before until we reach another leaf.

Once we have generated b leaves, we score them all and return the transition sequence leading up to the highest-scoring leaf as the answer. Just as in previous studies (Collins and Roark, 2004), we use the



(a) Beam Search (b) Dynamic Beam Search

(c) Selectional Branching (d) Heuristic Backtracking

Figure 1: Visualization of various decoding algorithms

sum of the log probabilities of all individual transitions as the overall score for the parse.

3.2 Calculating Error Likelihood

Let n indicate a node, which consists of a state, a buffer, and an action history. We may refer to a specific node as n_i^j, which means it has i actions in its action history and it is part of the history of the jth leaf (and possibly subsequent leaves). Let the function $T(n)$ represent a sorted vector containing all possible transitions from n, and $S(n)$ represent a sorted vector containing the scores of all of these transitions, in terms of log probabilities of each score. We can index the scores in order of value, so $T_1(n)$ is the highest-scoring transition and $S_1(n)$ is its score, $T_2(n)$ is the second-highest-scoring transition, etc. Here, let u_n indicate the ranking of the transition leading to the first unexplored child of a node n. Also, let $V(n)$ represent the total score of all nodes in the history of n, i.e. the sum of all the scores of individual transitions that allowed us to get to n.

To calculate the confidence of an individual node, Choi and McCallum (2013) simply found the score margin, or difference in probability between the top-scoring transition and the second-highest scoring transition: $C(n) = S_1(n) - S_2(n)$. In selectional branching, the only states for which the confidence was relevant were the states in the first greedy parse, i.e. states n_i^1 for all i. For heuristic backtracking, we wish to generalize this to any state n_i^j for all i and j. We do this in the following way:

$$H(n_i^j) = (V(n_i^1) - V(n_i^j)) + (S_{(u_{n_i^j})-1}(n_i^j) + S_{(u_{n_i^j})}(n_i^j)) \quad (4)$$

Intuitively, this formula means that the node that will be explored first is the node that will yield a parse that scores as close to the greedy choice as possible. The first term ensures that it has a history of good choices, and the second term ensures that the new child node being explored will be nearly as good as the prior child.

3.3 Number of Predictions

As discussed earlier, we use number of predictions made by the model as a proxy for the speed; execution speed may vary based on system and algorithmic implementation, but prediction count gives a good estimate of the overall work done by the algorithm.

Consider a sentence of length l, which requires at most $2l$ transitions with the greedy decoder (Nivre, 2004). The number of predictions required for heuristic backtracking for b leaves is guaranteed to be less than or equal to a beam search with b beams.

When doing a beam search, the first transition will require 1 prediction, and then every subsequent transition will require 1 prediction per beam, or b predictions. This results in a total of $b(2l-1)+1$ predictions.

When doing heuristic backtracking, the first greedy search will require $2l$ predictions. Every

subsequent prediction will require a number of predictions dependent on the target of the backtrack: backtracking to n_i^j will require $2l - (i + 1)$ predictions. Note that $0 < i < 2l$. Thus, each backtrack will require at maximum $2l - 1$ predictions. Therefore, the maximum total amount of predictions is $2l + (b - 1)(2l - 1) = b(2l - 1) + 1$.

However, note that on average, there are significantly fewer. Assuming that all parts of a sentence have approximately equal score distributions, the average backtrack will be where $i = l$, and reduce predictions by 50%.

An intuitive understanding of this difference can be gained by viewing the graphs of various decoding methods in Figure 1. Beam search has many nodes which never yield children that reach an end-state; dynamic beam search has fewer, but still several. Selectional branching has none, but suffers from the restriction that every parse candidate can be no more than one decision away from the greedy parse. With heuristic backtracking, there is no such restriction, but yet every node explored is directly useful for generating a candidate parse.

3.4 Early Cutoff

Another inefficiency inherent to beam search is the fact that all b beams are always fully explored. Since the beams are calculated in parallel, this is inevitable. However, with heuristic backtracking, the beams are calculated incrementally; this gives us the opportunity to cut off our search at any point. In order to leverage this into more efficient parsing, we constructed a second Stack-LSTM model, which we call the cutoff model. The cutoff model uses a single Stack-LSTM[2] that takes as input the sequence of parser states (see Eq 1), and outputs a boolean variable predicting whether the entire parse is correct or incorrect.

To train the cutoff model, we used stochastic gradient descent over the training set. For each training example, we first parse it greedily using the Stack-LSTM parser. Then, for as long as the parse has at least one mistake, we pass it to the cutoff model as a negative training example. Once the parse is completely correct, we pass it to the cutoff model as a positive training example. The loss function that we use is:

$$\mathcal{L}_{\boldsymbol{\theta}} = -\log\ p(\mathbf{t} \mid \boldsymbol{s}) \tag{5}$$

where **s** is the LSTM encoded vector and **t** is the truth (parse correct/incorrect).

When decoding using early cutoff, we follow the exact same procedure as for normal heuristic backtracking, but after every candidate parse is generated, we use it as input to our cutoff model. When our cutoff model returns our selection as correct, we stop backtracking and return it as the answer. If we make b attempts without finding a correct parse, we follow the same procedure as before.

4 Experiments and Results

To test the effectiveness of heuristic backtracking, we compare it with other decoding techniques: greedy, beam search,[3], dynamic beam search (Mejia-Lavalle and Ramos, 2013), and selectional branching (Choi and McCallum, 2013). We then try heuristic backtracking (see Section 3.1), and heuristic backtracking with cutoff (see Section 3.4). Note that beam search was *not* used for early-update training (Collins and Roark, 2004). We use the same greedy training strategy for all models, and we only change the decoding strategy.

We tested the performance of these algorithms on the English SD and Chinese CTB.[4] A single model was trained using the techniques described in Section 2, and used as the transition model for all decoding algorithms. Each decoding technique was tested with varying numbers of beams; as b increased, both the predictions per sentence and accuracy trended upwards. The results are summarized in Table 1.[5] Note that we report results for only the highest-accuracy b (in the development set) for each.

We also report the results of the cutoff model in Table 2. The same greedily-trained model as above was used to generate candidate parses and confidence estimates for each transition, and then the cutoff model was trained to use these confidence esti-

[2] 2 layers and 300 dimensions.

[3] Greedy and beam-search were already explored by Dyer et al. (2015)

[4] Using the exact same settings as Dyer et al. (2015) with pretrained embeddings and part-of-speech tags.

[5] The development sets are used to set the model parameters; results on the development sets are similar to the ones obtained in the test sets.

English			
Decoding	**Pred/Sent**	**UAS**	**LAS**
Greedy – Dyer et al.	47.92	93.04%	90.87%
Beam Search	542.09	93.32%	91.19%
Dynamic Beam Search	339.42	93.32%	91.19%
Sel. Branching	59.66	93.24%	91.12%
Heur. Backtr.	198.03	93.30%	91.18%
Heur. Backtr. w/ Cutoff	108.32	93.27%	91.16%
Chinese			
Decoding	**Pred/Sent**	**UAS**	**LAS**
Greedy – Dyer et al.	53.79	87.31%	85.88%
Beam Search	815.65	87.62%	86.17%
Dynamic Beam Search	282.32	87.62%	86.17%
Sel. Branching	91.51	87.53%	86.08%
Heur. Backtr.	352.30	87.61%	86.16%
Heur. Backtr. w/ Cutoff	162.37	87.60%	86.15%

Table 1: UAS and LAS of various decoding methods. Pred/Sent refers to number of predictions made by the Stack-LSTM per sentence.

mates to discriminate between correctly-parsed and incorrectly-parsed sentences.

5 Discussion

In Table 1 we see that in both English and Chinese, the best heuristic backtracking performs approximately as well as the best beam search, while making less than half the predictions. This supports our hypothesis that heuristic backtracking can perform at the same level as beam search, but with increased efficiency.

Dynamic beam search also performed as well as full beam search, despite demonstrating a reduction in predictions on par with that of heuristic backtracking. Since the implementation of dynamic beam search is very straightforward for systems which have already implemented beam search, we believe this will prove to be a useful finding.

Heuristic backtracking with cutoff outperformed greedy decoding, and reduced transitions by an additional 50%. However, it increased accuracy slightly less than full heuristic backtracking. We believe this difference could be mitigated with an improved cutoff model; as can be seen in Table 2, the cutoff model was only able to discriminate between correct and incorrect parses around 75% of the time. Also, note that while predictions per sentence were low, the overall runtime was increased due to running the cutoff LSTM multiple times per sentence.

Language	**Cutoff Accuracy**
English	72.43%
Chinese	75.18%

Table 2: Test-set accuracy of cutoff model on English and Chinese.

6 Related Work

Heuristic backtracking is most similar to the work of Choi and McCallum (2013), but is distinguished from theirs by allowing new beams to be initialized from any point in the parse, rather than only from points in the initial greedy parse. Heuristic backtracking also bears similarity to greedy-best-first-search (Pearl, 1984), but is unique in that it guarantees that b candidate solutions will be found within $b(2l-1)+1$ predictions. Our work also relates to beam-search parsers (Zhang and Clark, 2008, *inter alia*).

7 Conclusions

We have introduced a novel decoding algorithm, called heuristic backtracking, and presented evidence that it performs at the same level as beam search for decoding, while being significantly more efficient. We have demonstrated this for both English and Chinese, using a parser with strong results with a greedy decoder. We expect that heuristic backtracking could be applied to any other transition-based parser with similar benefits.

We plan on experimenting with various heuristics and cutoff models, such as adapting the attention-based models of Bahdanau et al. (2014) to act as a guide for both the heuristic search and cutoff.

Acknowledgments

Miguel Ballesteros was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA).

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1455–1465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria, August. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 334–343. The Association for Computer Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proc. AISTATS*.

Manuel Mejia-Lavalle and Cesar Geovani Pereyra Ramos. 2013. Beam search with dynamic pruning for artificial intelligence hard problems. In *Proceedings of the 2013 International Conference on Mechatronics, Electronics and Automotive Engineering*, November.

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

Judea Pearl. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

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

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *CoNLL*, pages 142–152. ACL.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 562–571, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 188–193, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July. Association for Computational Linguistics.

Word Ordering Without Syntax

Allen Schmaltz and **Alexander M. Rush** and **Stuart M. Shieber**
Harvard University
{schmaltz@fas,srush@seas,shieber@seas}.harvard.edu

Abstract

Recent work on word ordering has argued that syntactic structure is important, or even required, for effectively recovering the order of a sentence. We find that, in fact, an n-gram language model with a simple heuristic gives strong results on this task. Furthermore, we show that a long short-term memory (LSTM) language model is even more effective at recovering order, with our basic model outperforming a state-of-the-art syntactic model by 11.5 BLEU points. Additional data and larger beams yield further gains, at the expense of training and search time.

1 Introduction

We address the task of recovering the original word order of a shuffled sentence, referred to as bag generation (Brown et al., 1990), shake-and-bake generation (Brew, 1992), or more recently, linearization, as standardized in a recent line of research as a method useful for isolating the performance of text-to-text generation models (Zhang and Clark, 2011; Liu et al., 2015; Liu and Zhang, 2015; Zhang and Clark, 2015). The predominant argument of the more recent works is that jointly recovering explicit syntactic structure is crucial for determining the correct word order of the original sentence. As such, these methods either generate or rely on given parse structure to reproduce the order.

Independently, Elman (1990) explored linearization in his seminal work on recurrent neural networks. Elman judged the capacity of early recurrent neural networks via, in part, the network's ability to predict word order in simple sentences. He notes,

> The order of words in sentences reflects a number of constraints... Syntactic structure, selective restrictions, subcategorization, and discourse considerations are among the many factors which join together to fix the order in which words occur... [T]here is an abstract structure which underlies the surface strings and it is this structure which provides a more insightful basis for understanding the constraints on word order.... It is, therefore, an interesting question to ask whether a network can learn any aspects of that underlying abstract structure (Elman, 1990).

Recently, recurrent neural networks have reemerged as a powerful tool for learning the latent structure of language. In particular, work on long short-term memory (LSTM) networks for language modeling has provided improvements in perplexity.

We revisit Elman's question by applying LSTMs to the word-ordering task, without any explicit syntactic modeling. We find that language models are in general effective for linearization relative to existing syntactic approaches, with LSTMs in particular outperforming the state-of-the-art by 11.5 BLEU points, with further gains observed when training with additional text and decoding with larger beams.

2 Background: Linearization

The task of linearization is to recover the original order of a shuffled sentence. We assume a vocabulary $\mathcal{V}$ and are given a sequence of out-of-order phrases $x_1, \ldots, x_N$, with $x_n \in \mathcal{V}^+$ for $1 \leq n \leq N$. Define M as the total number of tokens (i.e., the sum of the lengths of the phrases). We consider two varieties of the task: (1) WORDS, where each x_n consists of a single word and $M = N$, and (2) WORDS+BNPS,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2319–2324,
Austin, Texas, November 1-5, 2016.

where base noun phrases (noun phrases not containing inner noun phrases) are also provided and $M \geq N$. The second has become a standard formulation in recent literature.

Given input x, we define the output set $\mathcal{Y}$ to be all possible permutations over the N elements of x, where $\hat{y} \in \mathcal{Y}$ is the permutation generating the true order. We aim to find $\hat{y}$, or a permutation close to it. We produce a linearization by (approximately) optimizing a learned scoring function f over the set of permutations, $y^* = \arg\max_{y \in \mathcal{Y}} f(x, y)$.

3 Related Work: Syntactic Linearization

Recent approaches to linearization have been based on reconstructing the syntactic structure to produce the word order. Let $\mathcal{Z}$ represent all projective dependency parse trees over M words. The objective is to find $y^*, z^* = \arg\max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(x, y, z)$ where f is now over both the syntactic structure and the linearization. The current state of the art on the Penn Treebank (PTB) (Marcus et al., 1993), without external data, of Liu et al. (2015) uses a transition-based parser with beam search to construct a sentence and a parse tree. The scoring function is a linear model $f(x, y) = \theta^\top \Phi(x, y, z)$ and is trained with an early update structured perceptron to match both a given order and syntactic tree. The feature function Φ includes features on the syntactic tree. This work improves upon past work which used best-first search over a similar objective (Zhang and Clark, 2011).

In follow-up work, Liu and Zhang (2015) argue that syntactic models yield improvements over pure surface n-gram models for the WORDS+BNPS case. This result holds particularly on longer sentences and even when the syntactic trees used in training are of low quality. The n-gram decoder of this work utilizes a single beam, discarding the probabilities of internal, non-boundary words in the BNPs when comparing hypotheses. We revisit this comparison between syntactic models and surface-level models, utilizing a surface-level decoder with heuristic future costs and an alternative approach for scoring partial hypotheses for the WORDS+BNPS case.

Additional previous work has also explored n-gram models for the word ordering task. The work of de Gispert et al. (2014) demonstrates improvements over the earlier syntactic model of Zhang et al. (2012) by applying an n-gram language model over the space of word permutations restricted to concatenations of phrases seen in a large corpus. Horvat and Byrne (2014) models the search for the highest probability permutation of words under an n-gram model as a Travelling Salesman Problem; however, direct comparisons to existing works are not provided.

4 LM-Based Linearization

In contrast to the recent syntax-based approaches, we use an LM directly for word ordering. We consider two types of language models: an n-gram model and a long short-term memory network (Hochreiter and Schmidhuber, 1997). For the purpose of this work, we define a common abstraction for both models. Let $\boldsymbol{h} \in \mathcal{H}$ be the current state of the model, with $\boldsymbol{h}_0$ as the initial state. Upon seeing a word $w_i \in \mathcal{V}$, the LM advances to a new state $\boldsymbol{h}_i = \delta(w_i, \boldsymbol{h}_{i-1})$. At any time, the LM can be queried to produce an estimate of the probability of the next word $q(w_i, \boldsymbol{h}_{i-1}) \approx p(w_i \mid w_1, \ldots, w_{i-1})$. For n-gram language models, $\mathcal{H}$, δ, and q can naturally be defined respectively as the state space, transition model, and edge costs of a finite-state machine.

LSTMs are a type of recurrent neural network (RNN) that are conducive to learning long-distance dependencies through the use of an internal memory cell. Existing work with LSTMs has generated state-of-the-art results in language modeling (Zaremba et al., 2014), along with a variety of other NLP tasks.

In our notation we define $\mathcal{H}$ as the hidden states and cell states of a multi-layer LSTM, δ as the LSTM update function, and q as a final affine transformation and softmax given as $q(*, \boldsymbol{h}_{i-1}; \theta_q) = \mathrm{softmax}(\boldsymbol{W}\boldsymbol{h}_{i-1}^{(L)} + \boldsymbol{b})$ where $\boldsymbol{h}_{i-1}^{(L)}$ is the top hidden layer and $\theta_q = (\boldsymbol{W}, \boldsymbol{b})$ are parameters. We direct readers to the work of Graves (2013) for a full description of the LSTM update.

For both models, we simply define the scoring function as

$$f(x, y) = \sum_{n=1}^{N} \log p(x_{y(n)} \mid x_{y(1)}, \ldots, x_{y(n-1)})$$

where the phrase probabilities are calculated word-by-word by our language model.

Algorithm 1 LM beam-search word ordering

procedure ORDER($x_1 \dots x_N, K, g$)
  $B_0 \leftarrow \langle(\langle\rangle, \{1, \dots, N\}, 0, \boldsymbol{h}_0)\rangle$
  for $m = 0, \dots, M-1$ **do**
    for $k = 1, \dots, |B_m|$ **do**
      $(y, \mathcal{R}, s, \boldsymbol{h}) \leftarrow B_m^{(k)}$
      for $i \in \mathcal{R}$ **do**
        $(s', \boldsymbol{h}') \leftarrow (s, \boldsymbol{h})$
        for word w in phrase x_i **do**
          $s' \leftarrow s' + \log q(w, \boldsymbol{h}')$
          $\boldsymbol{h}' \leftarrow \delta(w, \boldsymbol{h}')$
        $j \leftarrow m + |x_i|$
        $B_j \leftarrow B_j + (y + x_i, \mathcal{R} - i, s', \boldsymbol{h}')$
        keep top-K of B_j by $f(x, y) + g(\mathcal{R})$
  return B_M

Searching over all permutations $\mathcal{Y}$ is intractable, so we instead follow past work on linearization (Liu et al., 2015) and LSTM generation (Sutskever et al., 2014) in adapting beam search for our generation step. Our work differs from the beam search approach for the WORDS+BNPS case of previous work in that we maintain multiple beams, as in stack decoding for phrase-based machine translation (Koehn, 2010), allowing us to incorporate the probabilities of internal, non-boundary words in the BNPs. Additionally, for both WORDS and WORDS+BNPS, we also include an estimate of future cost in order to improve search accuracy.

Beam search maintains $M + 1$ beams, $B_0, \dots, B_M$, each containing at most the top-K partial hypotheses of that length. A partial hypothesis is a 4-tuple $(y, \mathcal{R}, s, \boldsymbol{h})$, where y is a partial ordering, $\mathcal{R}$ is the set of remaining indices to be ordered, s is the score of the partial linearization $f(x, y)$, and $\boldsymbol{h}$ is the current LM state. Each step consists of expanding all next possible phrases and adding the next hypothesis to a later beam. The full beam search is given in Algorithm 1.

As part of the beam search scoring function we also include a future cost g, an estimate of the score contribution of the remaining elements in $\mathcal{R}$. Together, $f(x, y) + g(\mathcal{R})$ gives a noisy estimate of the total score, which is used to determine the K best elements in the beam. In our experiments we use a very simple unigram future cost estimate, $g(\mathcal{R}) = \sum_{i \in \mathcal{R}} \sum_{w \in x_i} \log p(w)$.

Model	WORDS	WORDS+BNPS
ZGEN-64	30.9	49.4
ZGEN-64+POS	–	50.8
NGRAM-64 (NO g)	32.0	51.3
NGRAM-64	37.0	54.3
NGRAM-512	38.6	55.6
LSTM-64	40.5	60.9
LSTM-512	42.7	63.2
ZGEN-64+LM+GW+POS	–	52.4
LSTM-64+GW	41.1	63.1
LSTM-512+GW	44.5	65.8

Table 1: BLEU score comparison on the PTB test set. Results from previous works (for ZGEN) are those provided by the respective authors, except for the WORDS task. The final number in the model identifier is the beam size, +GW indicates additional Gigaword data. Models marked with +POS are provided with a POS dictionary derived from the PTB training set.

5 Experiments

Setup Experiments are on PTB with sections 2-21 as training, 22 as validation, and 23 as test[1]. We utilize two UNK types, one for initial uppercase tokens and one for all other low-frequency tokens; end sentence tokens; and start/end tokens, which are treated as words, to mark BNPs for the WORDS+BNPS task. We also use a special symbol to replace tokens that contain at least one numeric character. We otherwise train with punctuation and the original case of each token, resulting in a vocabulary containing around 16K types from around 1M training tokens.

For experiments marked GW we augment the PTB with a subset of the Annotated Gigaword corpus (Napoles et al., 2012). We follow Liu and Zhang (2015) and train on a sample of 900k Agence France-Presse sentences combined with the full PTB training set. The GW models benefit from both additional data and a larger vocabulary of around 25K types, which reduces unknowns in the validation and test sets.

We compare the models of Liu et al. (2015)

[1]In practice, the results in Liu et al. (2015) and Liu and Zhang (2015) use section 0 instead of 22 for validation (author correspondence).

BNP	g	GW	1	10	64	128	256	512
					LSTM			
•			41.7	53.6	58.0	59.1	60.0	60.6
•	•		47.6	59.4	62.2	62.9	63.6	64.3
•	•	•	48.4	60.1	64.2	64.9	65.6	66.2
			15.4	26.8	33.8	35.3	36.5	38.0
	•		25.0	36.8	40.7	41.7	42.0	42.5
	•	•	23.8	35.5	40.7	41.7	42.9	43.7
					NGRAM			
•			40.6	49.7	52.6	53.2	54.0	54.7
•	•		45.7	53.6	55.6	56.2	56.6	56.6
			14.6	27.1	32.6	33.8	35.1	35.8
	•		27.1	34.6	37.5	38.1	38.4	38.7

Table 2: BLEU results on the validation set for the LSTM and NGRAM model with varying beam sizes, future costs, additional data, and use of base noun phrases.

(known as ZGEN), a 5-gram LM using Kneser-Ney smoothing (NGRAM)[2], and an LSTM. We experiment on the WORDS and WORDS+BNPS tasks, and we also experiment with including future costs (g), the Gigaword data (GW), and varying beam size. We retrain ZGEN using publicly available code[3] to replicate published results.

The LSTM model is similar in size and architecture to the medium LSTM setup of Zaremba et al. (2014)[4]. Our implementation uses the Torch[5] framework and is publicly available[6].

We compare the performance of the models using the BLEU metric (Papineni et al., 2002). In generation if there are multiple tokens of identical UNK type, we randomly replace each with possible unused tokens in the original source before calculating BLEU. For comparison purposes, we use the BLEU script distributed with the publicly available ZGEN code.

Results Our main results are shown in Table 1. On the WORDS+BNPS task the NGRAM-64 model scores nearly 5 points higher than the syntax-based model ZGEN-64. The LSTM-64 then surpasses

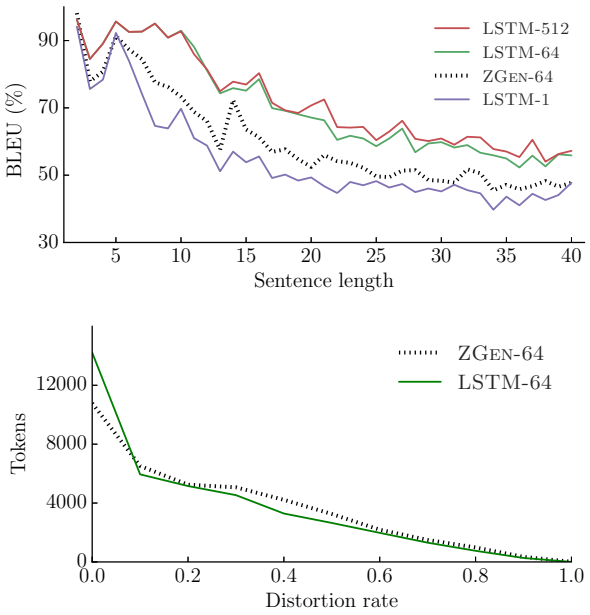


Figure 1: Experiments on the PTB validation on the WORDS+BNPS models: (a) Performance on the set by length on sentences from length 2 to length 40. (b) The distribution of token distortion, binned at 0.1 intervals.

NGRAM-64 by more than 5 BLEU points. Differences on the WORDS task are smaller, but show a similar pattern. Incorporating Gigaword further increases the result another 2 points. Notably, the NGRAM model outperforms the combined result of ZGEN-64+LM+GW+POS from Liu and Zhang (2015), which uses a 4-gram model trained on Gigaword. We believe this is because the combined ZGEN model incorporates the n-gram scores as discretized indicator features instead of using the probability directly.[7] A beam of 512 yields a further improvement at the cost of search time.

To further explore the impact of search accuracy, Table 2 shows the results of various models with beam widths ranging from 1 (greedy search) to 512, and also with and without future costs g. We see that for the better models there is a steady increase in accuracy even with large beams, indicating that search errors are made even with relatively large beams.

[2]We use the KenLM Language Model Toolkit (`https://kheafield.com/code/kenlm/`).

[3]`https://github.com/SUTDNLP/ZGen`

[4]We hypothesize that additional gains are possible via a larger model and model averaging, ceteris paribus.

[5]`http://torch.ch`

[6]`https://github.com/allenschmaltz/word_ordering`

[7]In work of Liu and Zhang (2015), with the given decoder, N-grams only yielded a small further improvement over the syntactic models when discretized versions of the LM probabilities were incorporated as indicator features in the syntactic models.

Model	WORDS	WORDS+BNPS
ZGEN-64(z^*)	39.7	64.9
ZGEN-64	40.8	65.2
NGRAM-64	46.1	67.0
NGRAM-512	47.2	67.8
LSTM-64	51.3	71.9
LSTM-512	52.8	73.1

Table 3: Unlabeled attachment scores (UAS) on the PTB validation set after parsing and aligning the output. For ZGEN we also include a result using the tree z^* produced directly by the system. For WORDS+BNPS, internal BNP arcs are always counted as correct.

One proposed advantage of syntax in linearization models is that it can better capture long-distance relationships. Figure 1 shows results by sentence length and distortion, which is defined as the absolute difference between a token's index position in y^* and $\hat{y}$, normalized by M. The LSTM model exhibits consistently better performance than existing syntax models across sentence lengths and generates fewer long-range distortions than the ZGEN model.

Finally, Table 3 compares the syntactic fluency of the output. As a lightweight test, we parse the output with the Yara Parser (Rasooli and Tetreault, 2015) and compare the unlabeled attachment scores (UAS) to the trees produced by the syntactic system. We first align the gold head to each output token. (In cases where the alignment is not one-to-one, we randomly sample among the possibilities.) The models with no knowledge of syntax are able to recover a higher proportion of gold arcs.

6 Conclusion

Strong surface-level language models recover word order more accurately than the models trained with explicit syntactic annotations appearing in a recent series of papers. This has implications for the utility of costly syntactic annotations in generation models, for both high- and low- resource languages and domains.

Acknowledgments

We thank Yue Zhang and Jiangming Liu for assistance in using ZGen, as well as verification of the task setup for a valid comparison. Jiangming Liu also assisted in pointing out a discrepancy in the implementation of an earlier version of our NGRAM decoder, the resolution of which improved BLEU performance.

References

[Brew1992] Chris Brew. 1992. Letting the cat out of the bag: Generation for shake-and-bake MT. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 610–616, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brown et al.1990] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

[de Gispert et al.2014] Adrià de Gispert, Marcus Tomalin, and Bill Byrne. 2014. Word ordering with phrase-based grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 259–268, Gothenburg, Sweden, April. Association for Computational Linguistics.

[Elman1990] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179 – 211.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

[Horvat and Byrne2014] Matic Horvat and William Byrne. 2014. A graph-based approach to string regeneration. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 85–95, Gothenburg, Sweden, April. Association for Computational Linguistics.

[Koehn2010] Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

[Liu and Zhang2015] Jiangming Liu and Yue Zhang. 2015. An empirical comparison between n-gram and syntactic language models for word ordering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 369–378, Lisbon, Portugal, September. Association for Computational Linguistics.

[Liu et al.2015] Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado, May–June. Association for Computational Linguistics.

[Marcus et al.1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

[Napoles et al.2012] Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Rasooli and Tetreault2015] Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *CoRR*, abs/1503.06733.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

[Zaremba et al.2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

[Zhang and Clark2011] Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1147–1157, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zhang and Clark2015] Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Comput. Linguist.*, 41(3):503–538, September.

[Zhang et al.2012] Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 736–746, Stroudsburg, PA, USA. Association for Computational Linguistics.

Morphological Segmentation Inside-Out

Ryan Cotterell
Department of Computer Science
Johns Hopkins University
ryan.cotterell@jhu.edu

Arun Kumar
Faculty of Arts and Humanities
Universitat Oberta de Catalunya

Hinrich Schütze
CIS
LMU Munich

Abstract

Morphological segmentation has traditionally been modeled with non-hierarchical models, which yield flat segmentations as output. In many cases, however, proper morphological analysis requires hierarchical structure—especially in the case of derivational morphology. In this work, we introduce a discriminative joint model of morphological segmentation along with the orthographic changes that occur during word formation. To the best of our knowledge, this is the first attempt to approach discriminative segmentation with a context-free model. Additionally, we release an annotated treebank of 7454 English words with constituency parses, encouraging future research in this area.

1 Introduction

In NLP, supervised morphological segmentation has typically been viewed as either a sequence labeling or a segmentation task (Ruokolainen et al., 2016). In contrast, we consider a hierarchical approach, employing a context-free grammar (CFG). CFGs provide a richer model of morphology: they capture (i) the intuition that words themselves have internal constituents, which belong to different categories, as well as (ii) the order in which affixes are attached. Moreover, many morphological processes, e.g., compounding and reduplication, are best modeled as hierarchical; thus, context-free models are expressively more appropriate.

The purpose of morphological segmentation is to decompose words into smaller units, known as morphemes, which are typically taken to be the smallest meaning bearing units in language. This work concerns itself with modeling hierarchical structure over these morphemes. Note a simple flat morphological segmentation can also be straightforwardly derived from the CFG parse tree. Segmentations have found use in a diverse set of NLP applications, e.g., automatic speech recognition (Afify et al., 2006), keyword spotting (Narasimhan et al., 2014), machine translation (Clifton and Sarkar, 2011) and parsing (Seeker and Çetinoğlu, 2015). In contrast to prior work, we focus on **canonical segmentation**, i.e., we seek to jointly model orthographic changes and segmentation. For instance, the canonical segmentation of *untestably* is *un*+*test*+*able*+*ly*, where we map *ably* to *able*+*ly*, restoring the letters *le*.

We make two contributions: (i) We introduce a joint model for canonical segmentation with a CFG backbone. We experimentally show that this model outperforms a semi-Markov model on flat segmentation. (ii) We release the first morphology treebank, consisting of 7454 English word types, each annotated with a full constituency parse.

2 The Case For Hierarchical Structure

Why should we analyze morphology hierarchically? It is true that we can model much of morphology with finite-state machinery (Beesley and Karttunen, 2003), but there are, nevertheless, many cases where hierarchical structure appears requisite. For instance, the flat segmentation of the word *untestably*↦*un*+*test*+*able*+*ly* is missing important information about how the word was derived. The correct *parse* [[*un*[[*test*]*able*]]*ly*], on the other hand, does tell us that this is the order in which the com-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2325–2330,
Austin, Texas, November 1-5, 2016.

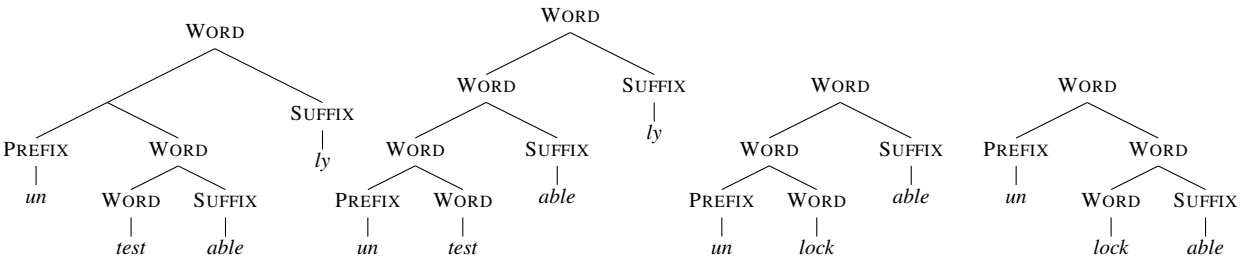


Figure 1: Canonical segmentation parse trees for *untestably* and *unlockable*. For both words, the scope of *un* is ambiguous. Arguably, (a) is the only correct parse tree for *untestably*; the reading associated with (b) is hard to get. On the other hand, *unlockable* is truly ambiguous between "able to be unlocked" (c) and "unable to be locked" (d).

plex form was derived:

$$\textit{test} \xmapsto{\text{able}} \textit{testable} \xmapsto{\text{un}} \textit{untestable} \xmapsto{\text{ly}} \textit{untestably}.$$

This gives us clear insight into the structure of the lexicon—we should expect that the segment *testable* exists as an independent word, but *ably* does not.

Moreover, a flat segmentation is often semantically ambiguous. There are two potentially valid readings of *untestably* depending on how the negative prefix *un* scopes. The correct tree (see Figure 1) yields the reading "in the manner of not able to be tested". A second—likely infelicitous reading—where the segment *untest* forms a constituent yields the reading "in a manner of being able to untest". Recovering the hierarchical structure allows us to select the correct reading; note there are even cases of true ambiguity; e.g., *unlockable* has two readings: "unable to be locked" and "able to be unlocked".

Theoretical linguists often implicitly assume a context-free treatment of word formation, e.g., by employing brackets to indicate different levels of affixation. Others have explicitly modeled word-internal structure with grammars (Selkirk, 1982; Marvin, 2002).

3 Parsing the Lexicon

A novel component of this work is the development of a discriminative parser (Finkel et al., 2008; Hall et al., 2014) for morphology. The goal is to define a probability distribution over all trees that could arise from the input word, after reversal of orthographic and phonological processes. We employ the simple grammar shown in Table 1. Despite its simplicity, it models the *order in which morphemes are attached*.

More formally, our goal is to map a surface form w (e.g., w=*untestably*) into its underlying canonical form u (e.g., u=*untestablely*) and then into a parse tree t over its morphemes. We assume $u, w \in \Sigma^*$, for some discrete alphabet Σ.[1] Note that a parse tree over the string implicitly defines a flat segmentation given our grammar—one can simply extract the characters spanned by all preterminals in the resulting tree. Before describing the joint model in detail, we first consider its pieces individually.

3.1 Restoring Orthographic Changes

To extract a *canonical segmentation* (Naradowsky and Goldwater, 2009; Cotterell et al., 2016), we restore orthographic changes that occur during word formation. To this end, we define the score function,

$$\text{score}_{\boldsymbol{\eta}}(u, w) = \sum_{a \in A(u,w)} \exp\left(\boldsymbol{g}(u, a, w)^\top \boldsymbol{\eta}\right), \quad (1)$$

where $A(u, w)$ is the set of all monotonic alignments between the strings u and w. The goal is for $\text{score}_{\boldsymbol{\eta}}$ to assign higher values to better matched pairs, e.g., (w=*untestably*, u=*untestablely*).

We have left these out of the equation for simplicity, but we refer to the reader Dreyer et al. (2008) for a more thorough exposition. To ensure that the partition function $Z_{\boldsymbol{\eta}}(w)$ is finite, we cap the maximum string length u, which yields

$$Z_{\boldsymbol{\eta}}(w) = \sum_{u' \in \Sigma^{|w|+k}} \exp\left(\boldsymbol{g}(u', w)^\top \boldsymbol{\eta}\right), \quad (2)$$

[1] For efficiency, we assume $u \in \Sigma^{|w|+k}$, $k = 5$.

ROOT	$\rightarrow$	WORD
WORD	$\rightarrow$	PREFIX WORD
WORD	$\rightarrow$	WORD SUFFIX
WORD	$\rightarrow$	Σ^+
PREFIX	$\rightarrow$	Σ^+
SUFFIX	$\rightarrow$	Σ^+

Table 1: The context-free grammar used in this work to model word formation. The productions closely resemble those of Johnson et al. (2006)'s Adaptor Grammar.

where $|w| + k$ is the maximum length for u.

For ease of computation, we can encode this function as a weighted finite-state machine (WFST) (Mohri et al., 2002). This requires, however, that the feature function $\boldsymbol{g}$ factors over the topology of the finite-state encoding. Since our model conditions on the word w, the feature function $\boldsymbol{g}$ can extract features from *any* part of this string. Features on the output string, u, however, are more restricted. In this work, we employ a bigram model over output characters. This implies that each state remembers exactly one character, the previous one. See Cotterell et al. (2014) for details. We can compute the score for two strings u and w using a weighted generalization of the Levenshtein algorithm. Computing the partition function requires a different dynamic program, which runs in $\mathcal{O}(|w|^2 \cdot |\Sigma|^2)$ time. Note that since $|\Sigma| \approx 26$ (lower case English letters), it takes roughly $26^2 = 676$ times longer to compute the partition function than to score a pair of strings.

Our model includes several simple feature templates, including features that fire on individual edit actions as well as conjunctions of edit actions and characters in the surrounding context. See Cotterell et al. (2016) for details.

3.2 Morphological Analysis as Parsing

Next, we need to score an underlying canonical form (e.g., u=*untestablely*) together with a parse tree (e.g., t=[[*un*[[*test*]*able*]]*ly*]). Thus, we define the parser score

$$\text{score}_{\boldsymbol{\omega}}(t, u) = \exp\left(\sum_{\pi \in \Pi(t)} \boldsymbol{f}(\pi, u)^\top \boldsymbol{\omega}\right), \quad (3)$$

where $\Pi(t)$ is the set of *anchored productions* in the tree t. An anchored production π is a grammar rule in Chomsky normal form attached to a span, e.g., $A_{i,k} \rightarrow B_{i,j} C_{j+1,k}$. Each π is then assigned a weight by the linear function $\boldsymbol{f}(\pi, u)^\top \boldsymbol{\omega}$, where the function $\boldsymbol{f}$ extracts relevant features from the anchored production as well as the corresponding span of the underlying form u. This model is typically referred to as a weighted CFG (WCFG) (Smith and Johnson, 2007) or a CRF parser.

Luckily, we can exploit the structure of the problem to efficiently compute the partition function,

$$Z_{\boldsymbol{\omega}}(u) = \sum_{t \in \mathcal{T}(u)} \exp\left(\sum_{\pi \in \Pi(t)} \boldsymbol{f}(\pi, u)^\top \boldsymbol{\omega}\right), \quad (4)$$

where $\mathcal{T}(u)$ is the set of all trees under the grammar that have yield u. Specifically, we make use of the inside algorithm, which is just CKY (Aho and Ullman, 1979) in the $(+, \times)$ semiring (Goodman, 1998), which runs in $\mathcal{O}(|\mathcal{G}| \cdot |u|^3)$ time, where $|\mathcal{G}|$ is the size of the grammar.

For $\boldsymbol{f}$, we define three span features: (i) indicator features on the span's segment, (ii) an indicator feature that fires if the segment appears in an external corpus[2] and (iii) the conjunction of the segment with the label (e.g., PREFIX) of the subtree root. Following Hall et al. (2014), we employ an indicator feature for each production as well as production backoff features.

4 A Joint Model

Our complete model is a joint CRF (Koller and Friedman, 2009) where each of the above scores are factors. We define the likelihood as

$$p_{\boldsymbol{\theta}}(t, u \mid w) = \frac{1}{Z_{\boldsymbol{\theta}}} \text{score}_{\boldsymbol{\omega}}(t, u) \cdot \text{score}_{\boldsymbol{\eta}}(u, w), \quad (5)$$

where $\boldsymbol{\theta} = \{\boldsymbol{\omega}, \boldsymbol{\eta}\}$ is the parameter vector and

$$Z_{\boldsymbol{\theta}} = \sum_{u' \in \Sigma^*} \sum_{t' \in \mathcal{T}_{u'}} \text{score}_{\boldsymbol{\omega}}(t', u') \cdot \text{score}_{\boldsymbol{\eta}}(u', w) \quad (6)$$

is the partition function and $\mathcal{T}_{u'}$ is set of all parse trees for the string u'. We see now that both WFST and WCFG are just structured factors in the model.

[2] We use the Wikipedia dump from 2016-05-01.

The joint approach has the advantage that it allows both factors to work together to influence the choice of the underlying form u. This is useful as the parser now has access to which words are attested in the language; this helps guide the relatively weak transduction model. On the downside, the partition function $Z_{\boldsymbol{\theta}}$ now involves a sum over *both* all strings in $\Sigma^{|w|+k}$ *and* all possible parses of each string! Inference in this joint model is intractable, so we resort to approximate methods.

4.1 Learning and Inference

We use stochastic gradient descent to optimize the log-probability of the training data $\sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(t^{(i)}, u^{(i)} \mid w^{(i)})$; this requires the computation of the gradient of the partition function $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$, which is intractable. As in all CRFs, this gradient is in fact an expectation:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \mathbb{E}_{(u,t)\sim p_{\boldsymbol{\theta}}} \left[\log \left(\text{score}_{\boldsymbol{\omega}}(t,u) \cdot \text{score}_{\boldsymbol{\eta}}(u,w)\right)\right]. \quad (7)$$

To approximate this expectation, we use an importance sampling routine. Roughly speaking, we approximate the hard-to-sample-from distribution $p_{\boldsymbol{\theta}}$ by taking samples from an easy-to-sample-from proposal distribution q. We then *reweight* the samples using the *unnormalized* score from $p_{\boldsymbol{\theta}}$. Due to a lack of space, we omit the derivation of the approximate gradient.

4.2 Decoding

We also decode by importance sampling. Given w, we sample canonical forms u and then run the CKY algorithm to get the highest scoring tree.

5 Related Work

We believe our attempt to train discriminative grammars for morphology is novel. Nevertheless, other researchers have described parsers for morphology. Most of this work is unsupervised: Johnson et al. (2007) applied a Bayesian PCFG to unsupervised morphological segmentation. Similarly, Adaptor Grammars (Johnson et al., 2006), a non-parametric Bayesian generalization of PCFGs, have been applied to the unsupervised version of the task (Botha and Blunsom, 2013; Sirts and Goldwater, 2013). Relatedly, Schmid (2005) performed unsupervised disambiguation of a German morphological analyzer (Schmid et al., 2004) using a PCFG, using the inside-outside algorithm (Baker, 1979). Also, discriminative parsing approaches have been applied to the related problem of Chinese word segmentation (Zhang et al., 2014).

6 Morphological Treebank

Supervised morphological segmentation has historically been treated as a segmentation problem, devoid of hierarchical structure. A core reason behind this is that—to the best of our knowledge—there are no hierarchically annotated corpora for the task. To remedy this, we provide tree annotations for a subset of the English portion of CELEX (Baayen et al., 1993). We reannotated 7454 English types with a full constituency parse.[3] The resource will be freely available for future research.

6.1 Annotation Guidelines

The annotation of the morphology treebank was guided by three core principles. The first principle concerns **productivity**: we exclusively annotate *productive* morphology. In the context of morphology, productivity refers to the degree that native speakers actively employ the affix to create new words (Aronoff, 1976). We believe that for NLP applications, we should focus on productive affixation. Indeed, this sets our corpus apart from many existing morphologically annotated corpora such as CELEX. For example, CELEX contains *warmth*↦*warm*+*th*, but *th* is not a productive suffix and cannot be used to create new words. Thus, we *do not* want to analyze *hearth*↦*hear*+*th* or, in general, allow *wug*↦*wug*+*th*. Second, we annotate for **semantic coherence**. When there are several candidate parses, we choose the one that is best compatible with the compositional semantics of the derived form.

Interestingly, multiple trees can be considered valid depending on the linguistic tier of interest. Consider the word *unhappier*. From a semantic perspective, we have the parse [[*un* [*happy*]] *er*] which gives us the correct meaning "not happy to a greater degree". However, since the suffix *er* only attaches to mono- and bisyllabic words, we get [*un*[[*happy*] *er*]] from a phonological perspective. In the linguis-

[3] In many cases, we corrected the flat segmentation as well.

	Segmentation			Tree
	Morph. F_1	*Edit*	*Acc.*	*Const.* F_1
Flat	78.89 (0.9)	0.72 (0.04)	72.88 (1.21)	N/A
Hier	**85.55** (0.6)	**0.55** (0.03)	**73.19** (1.09)	79.01 (0.5)

Table 2: Results for the 10 splits of the treebank. Segmentation quality is measured by morpheme F_1, edit distance and accuracy; tree quality by constituent F_1.

tics literature, this problem is known as the bracketing paradox (Pesetsky, 1985; Embick, 2015). We annotate exclusively at the syntactic-semantic tier.

Thirdly, in the context of derivational morphology, we *force spans to be words* themselves. Since derivational morphology—by definition—forms new words from existing words (Lieber and Štekauer, 2014), it follows that each span rooted with WORD or ROOT in the correct parse corresponds to a word in the lexicon. For example, consider *unlickable*. The correct parse, under our scheme, is [*un* [[*lick*] *able*]]. Each of the spans (*lick*, *lickable* and *unlickable*) exists as a word. By contrast, the parse [[*un* [*lick*]] *able*] contains the span *unlick*, which is not a word in the lexicon. The span in the segmented form may involve changes, e.g., [*un* [[*achieve*] *able*]], where *achieveable* is not a word, but *achievable* (after deleting *e*) is.

7 Experiments

We run a simple experiment to show the empirical utility of parsing words—we compare a WCFG-based canonical segmenter with the semi-Markov segmenter introduced in Cotterell et al. (2016). We divide the corpus into 10 distinct train/dev/test splits with 5454 words for train and 1000 for each of dev and test. We report three evaluation metrics: full form accuracy, morpheme F_1 (Van den Bosch and Daelemans, 1999) and average edit distance to the gold segmentation with boundaries marked by a distinguished symbol. For the WCFG model, we also report constituent F_1—typical for sentential constituency parsing— as a baseline for future systems. This F_1 measures how well we predict the whole tree (not just a segmentation). For all models, we use L_2 regularization and run 100 epochs of ADAGRAD (Duchi et al., 2011) with early stopping. We tune the regularization coefficient by grid search considering $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

7.1 Results and Discussion

Table 2 shows the results. The hierarchical WCFG model outperforms the flat semi-Markov model on all metrics on the segmentation task. This shows that modeling structure among the morphemes, indeed, does help segmentation. The largest improvements are found under the morpheme F_1 metric (≈ 6.5 points). In contrast, accuracy improves by $< 1\%$. Edit distance is in between with an improvement of 0.2 characters. Accuracy, in general, is an all or nothing metric since it requires getting every canonical segment correct. Morpheme F_1, on the other hand, gives us partial credit. Thus, what this shows us is that the WCFG gets a lot more of the morphemes in the held-out set correct, even if it only gets a few complete forms correct. We provide additional results evaluating the entire tree with constituency F_1 as a future baseline.

8 Conclusion

We presented a discriminative CFG-based model for canonical morphological segmentation and showed empirical improvements on its ability to segment words under three metrics. We argue that our hierarchical approach to modeling morphemes is more often appropriate than the traditional flat segmentation. Additionally, we have annotated 7454 words with a morphological constituency parse. The corpus is available online at `http://ryancotterell.github.io/data/morphological-treebank` to allow for exact comparison and to spark future research.

Acknowledgements

The first author was supported by a DAAD Long-Term Research Grant and an NDSEG fellowship. The third author was supported by DFG (SCHU 2246/10-1).

References

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal Arabic speech recognition. In *INTERSPEECH*.

Alfred W Aho and Jeffrey Ullman. 1979. *Introduction to Automata theory, Languages and Computation*. Addison-Wesley.

Mark Aronoff. 1976. *Word Formation in Generative Grammar*. MIT Press.

R Harald Baayen, Richard Piepenbrock, and Rijn van H. 1993. The CELEX lexical data base on CD-ROM.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Kenneth R Beesley and Lauri Karttunen. 2003. *Finite-state Morphology: Xerox Tools and Techniques*. CSLI, Stanford.

Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non-concatenative morphology. In *EMNLP*, pages 345–356.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *ACL*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *ACL*.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *NAACL*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

David Embick. 2015. *The Morpheme: A Theoretical Introduction*, volume 31. Walter de Gruyter GmbH & Co KG.

Jenny Rose Finkel, Alex Kleeman, and Christopher D Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*, volume 46, pages 959–967.

Joshua Goodman. 1998. *Parsing Inside-out*. Ph.D. thesis, Harvard University.

David Leo Wright Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *ACL*, pages 228–237.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS*, pages 641–648.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov Chain Monte Carlo. In *HLT-NAACL*, pages 139–146.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Rochelle Lieber and Pavol Štekauer. 2014. *The Oxford Handbook of Derivational Morphology*. Oxford University Press, USA.

Tatjana Marvin. 2002. *Topics in the Stress and Syntax of Words*. Ph.D. thesis, Massachusetts Institute of Technology.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *IJCAI*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *EMNLP*.

David Pesetsky. 1985. Morphology and logical form. *Linguistic Inquiry*, 16(2):193–246.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. Comparative study of minimally supervised morphological segmentation. *Computational Linguistics*.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *LREC*.

Helmut Schmid. 2005. Disambiguation of morphological structure using a PCFG. In *EMNLP*, pages 515–522. Association for Computational Linguistics.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL*.

Elisabeth Selkirk. 1982. *The Syntax of Words*. Number 7 in Linguistic Inquiry Monograph Series. MIT Press.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *TACL*, 1:255–266.

Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Antal Van den Bosch and Walter Daelemans. 1999. Memory-based morphological analysis. In *ACL*.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level Chinese dependency parsing. In *ACL*, pages 1326–1336.

Parsing as Language Modeling

Do Kook Choe
Brown University
Providence, RI
dc65@cs.brown.edu

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

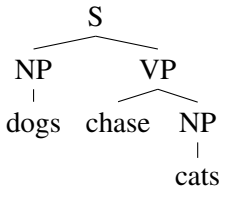
We recast syntactic parsing as a language modeling problem and use recent advances in neural network language modeling to achieve a new state of the art for constituency Penn Treebank parsing — 93.8 F_1 on section 23, using 2-21 as training, 24 as development, plus tri-training. When trees are converted to Stanford dependencies, UAS and LAS are 95.9% and 94.1%.

1 Introduction

Recent work on deep learning syntactic parsing models has achieved notably good results, e.g., Dyer et al. (2016) with 92.4 F_1 on Penn Treebank constituency parsing and Vinyals et al. (2015) with 92.8 F_1. In this paper we borrow from the approaches of both of these works and present a neural-net parse reranker that achieves very good results, 93.8 F_1, with a comparatively simple architecture.

In the remainder of this section we outline the major difference between this and previous work — viewing parsing as a language modeling problem. Section 2 looks more closely at three of the most relevant previous papers. We then describe our exact model (Section 3), followed by the experimental setup and results (Sections 4 and 5).

(a)



(b)

(S (NP dogs)$_{\mathrm{NP}}$ (VP chase (NP cats)$_{\mathrm{NP}}$)$_{\mathrm{VP}}$)$_{\mathrm{S}}$

Figure 1: A tree (a) and its sequential form (b). There is a one-to-one mapping between a tree and its sequential form. (Part-of-speech tags are not used.)

1.1 Language Modeling

Formally, a language model (LM) is a probability distribution over strings of a language:

$$P(\boldsymbol{x}) = P(x_1, \cdots, x_n) = \prod_{t=1}^{n} P(x_t | x_1, \cdots, x_{t-1}), \quad (1)$$

where $\boldsymbol{x}$ is a sentence and t indicates a word position. The efforts in language modeling go into computing $P(x_t|x_1, \cdots, x_{t-1})$, which as described next is useful for parsing as well.

1.2 Parsing as Language Modeling

A generative parsing model parses a sentence $(\boldsymbol{x})$ into its phrasal structure $(\boldsymbol{y})$ according to

$$\operatorname*{argmax}_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} P(\boldsymbol{x}, \boldsymbol{y}'),$$

where $\mathcal{Y}(\boldsymbol{x})$ lists all possible structures of $\boldsymbol{x}$. If we think of a tree $(\boldsymbol{x}, \boldsymbol{y})$ as a sequence $(\boldsymbol{z})$ (Vinyals et

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2331–2336,
Austin, Texas, November 1-5, 2016.

al., 2015) as illustrated in Figure 1, we can define a probability distribution over $(\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{z}) = P(z_1, \cdots, z_m) = \prod_{t=1}^{m} P(z_t | z_1, \cdots, z_{t-1}), \quad (2)$$

which is equivalent to Equation (1). We have reduced parsing to language modeling and can use language modeling techniques of estimating $P(z_t | z_1, \cdots, z_{t-1})$ for parsing.

2 Previous Work

We look here at three neural net (NN) models closest to our research along various dimensions. The first (Zaremba et al., 2014) gives the basic language modeling architecture that we have adopted, while the other two (Vinyals et al., 2015; Dyer et al., 2016) are parsing models that have the current best results in NN parsing.

2.1 LSTM-LM

The LSTM-LM of Zaremba et al. (2014) turns $(x_1, \cdots, x_{t-1})$ into h_t, a hidden state of an LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2003; Graves, 2013), and uses h_t to guess x_t:

$$P(x_t | x_1, \cdots, x_{t-1}) = P(x_t | h_t) = \mathrm{softmax}(W h_t)[x_t],$$

where W is a parameter matrix and $[i]$ indexes ith element of a vector. The simplicity of the model makes it easily extendable and scalable, which has inspired a character-based LSTM-LM that works well for many languages (Kim et al., 2016) and an ensemble of large LSTM-LMs for English with astonishing perplexity of 23.7 (Jozefowicz et al., 2016). In this paper, we build a parsing model based on the LSTM-LM of Zaremba et al. (2014).

2.2 MTP

Vinyals et al. (2015) observe that a phrasal structure $(\boldsymbol{y})$ can be expressed as a sequence and build a machine translation parser (MTP), a sequence-to-sequence model, which translates $\boldsymbol{x}$ into $\boldsymbol{y}$ using a conditional probability:

$$P(\boldsymbol{y} | \boldsymbol{x}) = P(y_1, \cdots, y_l | \boldsymbol{x}) = \prod_{t=1}^{l} P(y_t | \boldsymbol{x}, y_1, \cdots, y_{t-1}),$$

where the conditioning event $(\boldsymbol{x}, y_1, \cdots, y_{t-1})$ is modeled by an LSTM encoder and an LSTM decoder. The encoder maps $\boldsymbol{x}$ into $\boldsymbol{h}^e$, a set of vectors that represents $\boldsymbol{x}$, and the decoder obtains a summary vector (h'_t) which is concatenation of the decoder's hidden state (h^d_t) and weighted sum of word representations $(\sum_{i=1}^{n} \alpha_i h^e_i)$ with an alignment vector $(\boldsymbol{\alpha})$. Finally the decoder predicts y_t given h'_t. Inspired by MTP, our model processes sequential trees.

2.3 RNNG

Recurrent Neural Network Grammars (RNNG), a generative parsing model, defines a joint distribution over a tree in terms of actions the model takes to generate the tree (Dyer et al., 2016):

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{a}) = \prod_{t=1}^{m} P(a_t | a_1, \cdots, a_{t-1}), \quad (3)$$

where $\boldsymbol{a}$ is a sequence of actions whose output precisely matches the sequence of symbols in $\boldsymbol{z}$, which implies Equation (3) is the same as Equation (2). RNNG and our model differ in how they compute the conditioning event $(z_1, \cdots, z_{t-1})$: RNNG combines hidden states of three LSTMs that keep track of actions the model has taken, an incomplete tree the model has generated and words the model has generated whereas our model uses one LSTM's hidden state as shown in the next section.

3 Model

Our model, the model of Zaremba et al. (2014) applied to sequential trees and we call LSTM-LM from now on, is a joint distribution over trees:

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{z}) = \prod_{t=1}^{m} P(z_t | z_1, \cdots, z_{t-1}) = \prod_{t=1}^{m} P(z_t | h_t) = \prod_{t=1}^{m} \mathrm{softmax}(W h_t)[z_t],$$

where h_t is a hidden state of an LSTM. Due to lack of an algorithm that searches through an exponentially large phrase-structure space, we use an n-best parser to reduce $\mathcal{Y}(\boldsymbol{x})$ to $\mathcal{Y}'(\boldsymbol{x})$, whose size is polynomial, and use LSTM-LM to find $\boldsymbol{y}$ that satisfies

$$\underset{\boldsymbol{y}' \in \mathcal{Y}'(\boldsymbol{x})}{\operatorname{argmax}} P(\boldsymbol{x}, \boldsymbol{y}'). \quad (4)$$

3.1 Hyper-parameters

The model has three LSTM layers with 1,500 units and gets trained with truncated backpropagation through time with mini-batch size 20 and step size 50. We initialize starting states with previous mini-batch's last hidden states (Sutskever, 2013). The forget gate bias is initialized to be one (Jozefowicz et al., 2015) and the rest of model parameters are sampled from $\mathcal{U}(-0.05, 0.05)$. Dropout is applied to non-recurrent connections (Pham et al., 2014) and gradients are clipped when their norm is bigger than 20 (Pascanu et al., 2013). The learning rate is $0.25 \cdot 0.85^{\max(\epsilon - 15,\, 0)}$ where ϵ is an epoch number. For simplicity, we use vanilla softmax over an entire vocabulary as opposed to hierarchical softmax (Morin and Bengio, 2005) or noise contrastive estimation (Gutmann and Hyvärinen, 2012).

4 Experiments

We describe datasets we use for evaluation, detail training and development processes.[1]

4.1 Data

We use the Wall Street Journal (WSJ) of the Penn Treebank (Marcus et al., 1993) for training (2-21), development (24) and testing (23) and millions of auto-parsed "silver" trees (McClosky et al., 2006; Huang et al., 2010; Vinyals et al., 2015) for tri-training. To obtain silver trees, we parse the entire section of the New York Times (NYT) of the fifth Gigaword (Parker et al., 2011) with a product of eight Berkeley parsers (Petrov, 2010)[2] and ZPar (Zhu et al., 2013) and select 24 million trees on which both parsers agree (Li et al., 2014). We do not resample trees to match the sentence length distribution of the NYT to that of the WSJ (Vinyals et al., 2015) because in preliminary experiments Charniak parser (Charniak, 2000) performed better when trained on all of 24 million trees than when trained on resampled two million trees.

Given $\boldsymbol{x}$, we produce $\mathcal{Y}'(\boldsymbol{x})$, 50-best trees, with Charniak parser and find $\boldsymbol{y}$ with LSTM-LM as Dyer et al. (2016) do with their discriminative and generative models.[3]

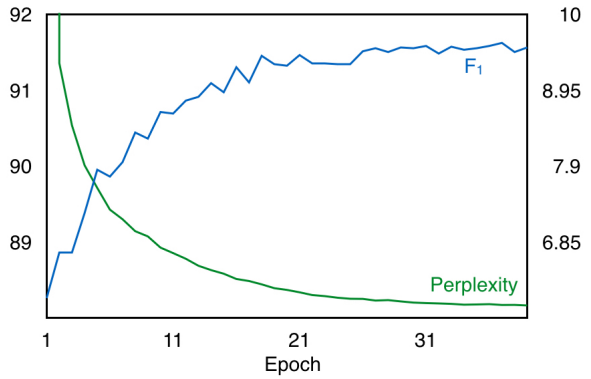


Figure 2: Perplexity and F_1 on the development set at each epoch during training.

n	Oracle	Final	Exact
10	94.0	91.2	39.8
50	96.7	91.7	40.0
51^o	100	93.9	49.7
100	96.3	91.7	39.9
500	97.0	91.8	40.0

Table 1: The performance of LSTM-LM (G) with varying n-best parses on the dev set. Oracle refers to Charniak parser's oracle F_1. Final and Exact report LSTM-LM (G)'s F_1 and exact match percentage respectively. To simulate an optimal scenario, we include gold trees to 50-best trees and rerank them with LSTM-LM (G) (51^o).

4.2 Training and Development

4.2.1 Supervision

We unk words that appear fewer than 10 times in the WSJ training (6,922 types) and drop activations with probability 0.7. At the beginning of each epoch, we shuffle the order of trees in the training data. Both perplexity and F_1 of LSTM-LM (G) improve and then plateau (Figure 2). Perplexity, the

[1] The code and trained models used for experiments are available at github.com/cdg720/emnlp2016.

[2] We use the reimplementation by Huang et al. (2010).

[3] Dyer et al. (2016)'s discriminative model performs comparably to Charniak (89.8 vs. 89.7).

	Base	Final
Vinyals et al. (2015)	88.3	90.5
Dyer et al. (2016)	89.8	92.4
LSTM-LM (G)	89.7	**92.6**

Table 2: F_1 of models trained on WSJ. Base refers to performance of a single base parser and Final that of a final parser.

model's training objective, nicely correlates with F_1, what we care about. Training takes 12 hours (37 epochs) on a Titan X. We also evaluate our model with varying n-best trees including optimal 51-best trees that contain gold trees (51^o). As shown in Table 1, the LSTM-LM (G) is robust given sufficiently large n, i.e. 50, but does not exhibit its full capacity because of search errors in Charniak parser. We address this problem in Section 5.3.

4.2.2 Semi-supervision

We unk words that appear at most once in the training (21,755 types). We drop activations with probability 0.45, smaller than 0.7, thanks to many silver trees, which help regularization. We train LSTM-LM (GS) on the WSJ and a different set of 400,000 NYT trees for each epoch except for the last one during which we use the WSJ only. Training takes 26 epochs and 68 hours on a Titan X. LSTM-LM (GS) achieves 92.5 F_1 on the development.

5 Results

5.1 Supervision

As shown in Table 2, with 92.6 F_1 LSTM-LM (G) outperforms an ensemble of five MTPs (Vinyals et al., 2015) and RNNG (Dyer et al., 2016), both of which are trained on the WSJ only.

5.2 Semi-supervision

We compare LSTM-LM (GS) to two very strong semi-supervised NN parsers: an ensemble of five MTPs trained on 11 million trees of the high-confidence corpus[4] (HC) (Vinyals et al., 2015); and an ensemble of six one-to-many sequence models trained on the HC and 4.5 millions of English-German translation sentence pairs (Luong et al., 2016). We also compare LSTM-LM (GS) to best performing non-NN parsers in the literature. Parsers' parsing performance along with their training data is reported in Table 3. LSTM-LM (GS) outperforms all the other parsers with 93.1 F_1.

5.3 Improved Semi-supervision

Due to search errors – good trees are missing in 50-best trees – in Charniak (G), our supervised and semi-supervised models do not exhibit their full potentials when Charniak (G) provides $\mathcal{Y}'(\boldsymbol{x})$. To mitigate the search problem, we tri-train Charniak (GS) on all of 24 million NYT trees in addition to the WSJ, to yield $\mathcal{Y}'(\boldsymbol{x})$. As shown in Table 3, both LSTM-LM (G) and LSTM-LM (GS) are affected by the quality of $\mathcal{Y}'(\boldsymbol{x})$. A single LSTM-LM (GS) together with Charniak (GS) reaches 93.6 and an ensemble of eight LSTM-LMs (GS) with Charniak (GS) achieves a new state of the art, 93.8 F_1. When trees are converted to Stanford dependencies,[5] UAS and LAS are 95.9% and 94.1%,[6] more than 1% higher than those of the state of the art dependency parser (Andor et al., 2016). Why an indirect method (converting trees to dependencies) is more accurate than a direct one (dependency parsing) remains unanswered (Kong and Smith, 2014).

6 Conclusion

The generative parsing model we presented in this paper is very powerful. In fact, we see that a generative parsing model, LSTM-LM, is more effective than discriminative parsing models (Dyer et al., 2016). We suspect building large models with character embeddings would lead to further improvement as in language modeling (Kim et al., 2016; Jozefowicz et al., 2016). We also wish to develop a complete parsing model using the LSTM-LM framework.

Acknowledgments

We thank the NVIDIA corporation for its donation of a Titan X GPU, Tstaff of Computer Science

[4]The HC consists of 90,000 gold trees, from the WSJ, English Web Treebank and Question Treebank, and 11 million silver trees, whose sentence length distribution matches that of the WSJ, parsed and agreed on by Berkeley parser and ZPar.

[5]Version 3.3.0.

[6]We use the CoNLL evaluator available through the CoNLL website: ilk.uvt.nl/conll/software/eval.pl. Following the convention, we ignore punctuation.

	Base	Oracle	Final	Gold	Silver
Huang et al. (2010)	-	-	92.8	WSJ (40K)	BLLIP (1.8M)
Shindo et al. (2012)	-	-	92.4	WSJ (40K)	-
Choe et al. (2015)	-	-	92.6	WSJ (40K)	NYT (2M)
Vinyals et al. (2015)	-	-	92.8	HC (90K)	HC (11M)
Luong et al. (2016)	-	-	93.0	HC (90K)	HC (11M)
Charniak (G) + LSTM-LM (G)	89.7	96.7	92.6	WSJ (40K)	-
Charniak (G) + LSTM-LM (GS)	89.7	96.7	93.1	WSJ (40K)	NYT (0/10M)
Charniak (GS) + LSTM-LM (G)	91.2	97.1	92.9	WSJ (40K)	NYT (24M/0)
Charniak (GS) + LSTM-LM (GS)	91.2	97.1	93.6	WSJ (40K)	NYT (24M/10M)
Charniak (GS) + E(LSTM-LMs (GS))	91.2	97.1	**93.8**	WSJ (40K)	NYT (24M/11.2M)

Table 3: Evaluation of models trained on the WSJ and additional resources. Note that the numbers of Vinyals et al. (2015) and Luong et al. (2016) are not directly comparable as their models are evaluated on OntoNotes-style trees instead of PTB-style trees. E(LSTM-LMs (GS)) is an ensemble of eight LSTM-LMs (GS). X/Y in Silver column indicates the number of silver trees used to train Charniak parser and LSTM-LM. For the ensemble model, we report the maximum number of trees used to train one of LSTM-LMs (GS).

at Brown University for setting up GPU machines and David McClosky for helping us train Charniak parser on millions trees.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Do Kook Choe, David McClosky, and Eugene Charniak. 2015. Syntactic parse fusion. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Lingpeng Kong and Noah A Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. *arXiv preprint arXiv:1404.4314*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task se-

quence to sequence learning. *International Conference on Learning Representations*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition. *Linguistic Data Consortium, LDC2011T07*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Ilya Sutskever. 2013. *Training recurrent neural networks*. Ph.D. thesis, University of Toronto.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Human-in-the-Loop Parsing

Luheng He **Julian Michael** **Mike Lewis** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA
{luheng,julianjm,mlewis,lsz}@cs.washington.edu

Abstract

This paper demonstrates that it is possible for a parser to improve its performance with a human in the loop, by posing simple questions to non-experts. For example, given the first sentence of this abstract, if the parser is uncertain about the subject of the verb "pose," it could generate the question *What would pose something?* with candidate answers *this paper* and *a parser*. Any fluent speaker can answer this question, and the correct answer resolves the original uncertainty. We apply the approach to a CCG parser, converting uncertain attachment decisions into natural language questions about the arguments of verbs. Experiments show that crowd workers can answer these questions quickly, accurately and cheaply. Our human-in-the-loop parser improves on the state of the art with less than 2 questions per sentence on average, with a gain of 1.7 F1 on the 10% of sentences whose parses are changed.

1 Introduction

The size of labelled datasets has long been recognized as a bottleneck in the performance of natural language processing systems (Marcus et al., 1993; Petrov and McDonald, 2012). Such datasets are expensive to create, requiring expert linguists and extensive annotation guidelines. Even relatively large datasets, such as the Penn Treebank, are much smaller than required—as demonstrated by improvements from semi-supervised learning (Søgaard and Rishøj, 2010; Weiss et al., 2015).

We take a step towards cheap, reliable annotations by introducing human-in-the-loop parsing, where non-experts improve parsing accuracy by answering questions automatically generated from the parser's output. We develop the approach for CCG parsing, leveraging the link between CCG syntax and semantics to convert uncertain attachment decisions into natural language questions. The answers are used as soft constraints when re-parsing the sentence.

Temple also said Sea Containers' plan raises numerous legal, regulatory, financial and fairness issues, but didn't ***elaborate***.	
Q:	What didn't ***elaborate***?
[1] ****	Temple
[2] *	Sea Containers' plan
[3]	None of the above.

Table 1: An automatically generated query from CCGbank. 4 out of 5 annotators correctly answered *Temple*, providing a signal that can be used to improve parse predictions.

Previous work used crowdsourcing for less structured tasks such as named entity recognition (Werling et al., 2015) and prepositional phrase attachment (Jha et al., 2010). Our work is most related to that of Duan et al. (2016), which automatically generates paraphrases from n-best parses and gained significant improvement by re-training from crowdsourced judgments on two out-of-domain datasets. Choe and McClosky (2015) improve a parser by creating paraphrases of sentences, and then parsing the sentence and its paraphrase jointly. Instead of using paraphrases, we build on the approach of QA-SRL (He et al., 2015), which shows that untrained crowd workers can annotate predicate–argument structures by writing question–answer pairs.

Our experiments for newswire and biomedical

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2337–2342,
Austin, Texas, November 1-5, 2016.

text demonstrate improvements to parsing accuracy of 1.7 F1 on the sentences changed by re-parsing, while asking only less than 2 questions per sentence. The annotations we collected[1] are a representation-independent resource that could be used to develop new models or human-in-the-loop algorithms for related tasks, including semantic role labeling and syntactic parsing with other formalisms.

2 Mapping CCG Parses to Queries

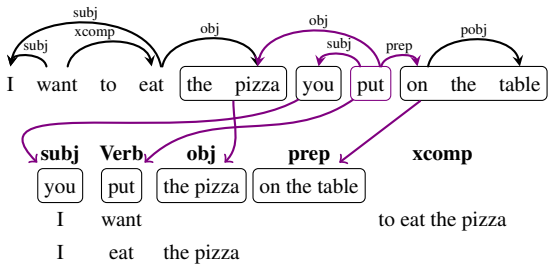
Our annotation task consists of multiple-choice *what*-questions that admit multiple answers. To generate them, we produce question–answer (QA) pairs from each parse in the 100-best scored output of a CCG parser and aggregate the results together.

We designed the approach to generate queries with high **question confidence**—questions should be simple and grammatical, so annotators are more likely to answer them correctly—and high **answer uncertainty**—the parser should be uncertain about the answers, so there is potential for improvement.

Our questions only apply to core arguments of verbs where the argument phrase is an NP, which account for many of the parser's mistakes. Prepositional phrase attachment mistakes are also a large source of errors—we tried several approaches to generate questions for these, but the greater ambiguity and inconsistency among both annotators and the gold parses made it difficult to extract meaningful signal from the crowd.

Generating Question–Answer Pairs Figure 1 shows how we generate QA pairs. Each QA pair corresponds to a dependency such that if the answer is correct, it indicates that the dependency is in the correct parse. We determine a verb's set of arguments by the CCG supertag assigned to it in the parse (see Steedman (2000) for an introduction to CCG). For example, in Figure 1 the word *put* takes the category ((S\NP)/PP)/NP (not shown), indicating that it has a subject, a prepositional phrase argument, and an object. CCG parsing assigns dependencies to each argument position, even when the arguments are reordered (as with *put* → *pizza*) or span long distances (as with *eat* → *I*).

[1]Our code and data are available at `https://github.com/luheng/hitl_parsing`.



Dependency	Question	Answer
want → I	What wants to eat something?	I
eat → I	What would eat something?	I
eat → pizza	What would something eat?	the pizza
put → you	What put something?	you
put → pizza	What did something put?	the pizza
on → table	What did something put something on?	the table

Figure 1: From a labeled dependency graph, we extract phrases corresponding to every argument of every verb using simple heuristics. We then create questions about dependencies, adding a *would* modal to untensed verbs and placing arguments to the left or right of the verb based on its CCG category. We only generate QA pairs for *subj*, *obj*, and *pobj* dependencies. To identify multiple answer options, we create QA pairs from all parses in the 100-best list and pool equivalent questions with different answers. See Table 2 for example queries.

To reduce the chance of parse errors causing nonsensical questions (for example, *What did the pizza put something on?*), we replace all noun phrases with *something* and delete unnecessary prepositional phrases. The exception to this is with copular predicates, where we include the span of the argument in the question (see Example 4 in Table 2).

Grouping QA Pairs into Queries After generating QA pairs for every parse in the 100-best output of the parser, we pool the QA pairs by the head of the dependency used to generate them, its CCG category, and their question strings. We also compute marginalized scores for each question and answer phrase by summing over the scores of all the parses that generated them. Each pool becomes a query, and for each unique dependency used to generate QA pairs in that pool, we add a candidate answer to the query by choosing the answer phrase that has the highest marginalized score for that dependency. For example, if some parses generated the answer phrase *pizza* for the dependency *eat* → *pizza*, but most of the high-scoring parses generated the answer phrase *the pizza*, then only *the pizza* appears as an answer.

Sentence	Question	Votes	Answers
(1) Structural Dynamics Research Corp. . . . said it introduced new technology in mechanical design automation that will ***improve*** mechanical engineering productivity.	What will ***improve*** something?	0	Structural Dynamics Research Corp
		5	**new technology**
		0	mechanical design automation
(2) He said disciplinary proceedings are confidential and declined to ***comment*** on whether any are being held against Mr. Trudeau.	What would ***comment***?	5	**he**
		0	disciplinary proceedings
(3) To ***avoid*** these costs, and a possible default, immediate action is imperative.	What would something ***avoid***?	4	**these costs**
		3	**a possible default**
(4) The price is a new high for California Cabernet Sauvignon, but it ***is*** not the highest.	What ***is*** not the highest?	2	the price
		3	**it**
(5) Kalipharma is a New Jersey-based pharmaceuticals concern that ***sells*** products under the Purepac label.	What ***sells*** something?	5	Kalipharma
		0	**a New Jersey-based pharmaceuticals concern**
(6) Further, he said, the company doesn't have the capital needed to ***build*** the business over the next year or two.	What would ***build*** something?	4	the company
		1	**the capital**
(7) Timex had requested duty-free treatment for many types of watches, ***covered*** by 58 different U.S. tariff classifications.	What would be ***covered***?	0	Timex
		0	duty-free treatment
		2	**many types of watches**
		3	watches
(8) You either believe Seymour can do it again or you ***do*** n't .	What ***does***?	3	**you**
		0	Seymour
		2	None of the above

Table 2: Example annotations from the CCGbank development set. Answers that agree with the gold parse are in bold. The answer choice *None of the above* was present for all examples, but we only show it when it was chosen by annotators.

From the resulting queries, we filter out questions and answers whose marginalized scores are below a certain threshold and queries that only have one answer choice. This way we only ask confident questions with uncertain answer lists.

3 Crowdsourcing

We collected data on the crowdsourcing platform CrowdFlower.[2] Annotators were shown a sentence, a question, and a list of answer choices. Annotators could choose multiple answers, which was useful in case of coordination (see Example 3 in Table 2). There was also a *None of the above* option for when no answer was applicable or the question was nonsensical.

We instructed annotators to only choose options that *explicitly* and *directly* answer the question, to encourage their answers to closely mirror syntax. We also instructed them to ignore *who*/*what* and *someone*/*something* distinctions and overlook mistakes where the question was missing a negation. The instructions included 6 example queries with answers and explanations. We used CrowdFlower's quality control mechanism, displaying pre-annotated queries 20% of the time and requiring annotators to maintain high accuracy.

Dataset Statistics Table 3 shows how many sentences we asked questions for and the total number of queries annotated. We collected annotations for the development and test set for CCGbank (Hockenmaier and Steedman, 2007) as in-domain data and the test set of the Bioinfer corpus (Pyysalo et al., 2007) as out-of-domain. The CCGbank development set was used for building question generation heuristics and setting hyperparameters for reparsing.

5 annotators answered each query; on CCGbank we required 85% accuracy on test questions and on Bioinfer we set the threshold at 80% because of the difficulty of the sentences. Table 4 shows inter-annotator agreement. Annotators unanimously chose the same set of answers for over 40% of the queries; an absolute majority is achieved for over 90% of the queries.

[2] www.crowdflower.com

Dataset	Sentences	Covered	Queries	Q/S
CCG-Dev	1913	1155	1904	1.7
CCG-Test	2407	1460	2511	1.7
Bioinfer	500	360	680	1.9

Table 3: Sentence coverage, number of queries annotated, and average number of queries per sentence (Q/S).

k-Agreed	CCG-Dev	CCG-Test	Bioinfer
5	48.0%	40.2%	47.7%
$\geq$ **4**	76.6%	68.0%	75.0%
$\geq$ **3**	94.9%	91.5%	94.0%

Table 4: The percentage of queries with at least k annotators agreeing on the exact same set of answers.

Qualitative Analysis Table 2 shows example queries from the CCGbank development set. Examples 1 and 2 show that workers could annotate long-range dependencies and scoping decisions, which are challenging for existing parsers.

However, there are some cases where annotators disagree with the gold syntax, mostly involving semantic phenomena which are not reflected in the syntactic structure. Many cases involve co-reference, where annotators often prefer a proper noun referent over a pronoun or indefinite (see Examples 4 and 5), even if it is not the syntactic argument of the verb. Example 6 shows a complex control structure, where the gold CCGbank syntax does not recover the true agent of *build*. CCGbank also does not distinguish between subject and object control. For these cases, our method could be used to extend existing treebanks. Another common error case involved partitives and related constructions, where the correct attachment is subtle—as reflected by the annotators' split decision in Example 7.

Question Quality Table 5 shows the percentage of questions that are answered with *None of the above* (written *N/A* below) by at most k annotators. On all domains, about 80% of the queries are considered answerable by all 5 annotators. To have a better understanding of the quality of automatically generated questions, we did a manual analysis on 50 questions for sentences from the CCGbank development set that are marked *N/A* by more than one annotator. Among the 50 questions, 31 of them are either generated from an incorrect supertag or unanswerable given the candidates. So the *N/A* answer

k-N/A	CCG-Dev	CCG-Test	Bioinfer
0	77.6%	81.6%	79.3%
$\leq$ **1**	89.6%	92.6%	89.1%
$\leq$ **2**	93.8%	96.1%	92.8%

Table 5: The percentage of queries with at most k annotators choosing the *None of the above* (N/A) option.

can provide useful signal that the parses that generated the question are likely incorrect. Common mistakes in question generation include: bad argument span in a copula question (4 questions), bad modality/negation (3 questions), and missing argument or particle (5 questions). Example 8 in Table 2 shows an example of a nonsensical question. While the parses agreed with the gold category $\mathsf{S\backslash NP}$, the question they generated omitted the negation and the verb phrase that was elided in the original sentence. In this case, 3 out of 5 annotators were able to answer with the correct dependency, but such mistakes can make re-parsing more challenging.

Cost and Speed We paid 6 cents for each answer. With 5 judgments per query, 20% test questions, and CrowdFlower's 20% service fee, the average cost per query was about 46 cents. On average, we collected about 1000 judgments per hour, so we were able to annotate all the queries generated from the CCGbank test set within 15 hours.

4 Re-Parsing with QA Annotation

To improve the output of the parser, we re-parse each sentence with an augmented scoring function that penalizes parses for disagreeing with annotators' choices. If q is a question, a is an answer to q, d is the dependency that produced the QA pair $\langle q, a \rangle$, and $v(a)$ annotators chose a, we add re-parsing constraints as follows:

- If $v(\textit{None of the above}) \geq T^+$, penalize parses that agree with q's supertag on the verb by w^t
- If $v(a) \leq T^-$, penalize parses containing d by w^-
- If $v(a) \geq T^+$, penalize parses that do not contain d by w^+

where T^+, T^-, w^t, w^-, and w^+ are hyperparameters. We incorporate these penalties into the parsing model during decoding. By using soft constraints, we mitigate the risk of incorrect annotations worsening a high-confidence parse.

Data	L16	HITL
CCG-Dev	87.9	88.4
CCG-Test	88.1	88.3
Bioinfer	82.2	82.8

Table 6: CCG parsing accuracy with human in the loop (HITL) versus the state-of-the-art baseline (L16) in terms of labeled F1 score. For both in-domain and out-domain, we have a modest gain over the entire corpus.

Data	L16	HITL	Pct.
CCG-Dev	83.9	87.1	12%
CCG-Test	84.2	85.9	10%

Table 7: Improvements of CCG parsing accuracy on changed sentences for in-domain data. We achieved significant improvement over the 10%–12% (Pct.) sentences that were changed by re-parsing.

Some errors are predictable: for example, if a is a non-possessive pronoun and is closer to the verb than its referent a', annotators often choose a' when a is correct (See Example 4 in Table 2). If a is a subspan of another answer a' and their votes differ by at most one (See Example 7 in Table 2), it is unlikely that both a and a' are correct. In these cases we use disjunctive constraints, where the parse needs to have at least one of the desired dependencies.

Experimental Setup We use Lewis et al. (2016)'s state-of-the-art CCG parser for our baseline. We chose the following set of hyperparameters based on performance on development data (CCG-Dev): $w^+ = 2.0$, $w^- = 1.5$, $w^t = 1.0$, $T^+ = 3$, $T^- = 0$. In the Bioinfer dataset, we found during development that the pronoun/subspan heuristics were not as useful, so we did not use them in re-parsing.

Results Table 6 shows our end-to-end parsing results. The larger improvement on out-of-domain sentences shows the potential for using our method for domain adaptation. There is a much smaller improvement on test data than development data, which may be related to the lower annotator agreement reported in Table 4.

There was much larger improvement (1.7 F1) on the subset of sentences that are changed after re-parsing, as shown in Table 7. This suggests that our method could be effective for semi-supervised learning or re-training parsers. Overall improvements on CCGbank are modest, due to only modifying 10% of sentences.

5 Discussion and Future Work

We introduced a human-in-the-loop framework for automatically correcting certain parsing mistakes. Our method identifies attachment uncertainty for core arguments of verbs and automatically generates questions that can be answered by untrained annotators. These annotations improve performance, particularly on out-of-domain data, demonstrating for the first time that untrained annotators can improve state-of-the-art parsers.

Sentences modified by our framework show substantial improvements in accuracy, but only 10% of sentences are changed, limiting the effect on overall accuracy. This work is a first step towards a complete approach to human-in-the-loop parsing.

Future work will explore the possibility of asking questions about other types of parsing uncertainties, such as nominal and adjectival argument structure, and a more thorough treatment of prepositional-phrase attachment, including distinctions between arguments and adjuncts. We hope to scale these methods to large unlabelled corpora or other languages, to provide data for re-training parsers.

Acknowledgments

This work was supported by the NSF (IIS-1252835, IIS-1562364), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. We are grateful to Chloé Kiddon for helpful comments on the paper, and Kenton Lee for help with the CCG parser. We would also like to thank our workers on Crowdflower for their annotation and the anonymous reviewers for their valuable feedback.

References

Do Kook Choe and David McClosky. 2015. Parsing paraphrases with joint inference. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Manjuan Duan, Ethan Hill, and Michael White. 2016. Generating disambiguating paraphrases for struc-

turally ambiguous sentences. In *Proceedings of the 10th Linguistic Annotation Workshop*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*.

Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. 2010. Corpus creation for new genres: A crowdsourced approach to pp attachment. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 Shared Task on Parsing the Web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Mark Steedman. 2000. *The syntactic process*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Keenon Werling, Arun Tejasvi Chaganty, Percy S Liang, and Christopher D Manning. 2015. On-the-job learning with bayesian decision theory. In *Advances in Neural Information Processing Systems*.

Unsupervised Timeline Generation for Wikipedia History Articles

Sandro Bauer and **Simone Teufel**
Computer Laboratory
University of Cambridge
Cambridge, United Kingdom
{sandro.bauer, simone.teufel}@cl.cam.ac.uk

Abstract

This paper presents a generic approach to content selection for creating timelines from individual history articles for which no external information about the same topic is available. This scenario is in contrast to existing works on timeline generation, which require the presence of a large corpus of news articles. To identify salient events in a given history article, we exploit lexical cues about the article's subject area, as well as time expressions that are syntactically attached to an event word. We also test different methods of ensuring timeline coverage of the entire historical time span described. Our best-performing method outperforms a new unsupervised baseline and an improved version of an existing supervised approach. We see our work as a step towards more semantically motivated approaches to single-document summarisation.

1 Introduction

While there has been much work on generating history timelines automatically, these approaches are commonly evaluated on events that took place in recent decades, as they depend on the availability of large numbers of articles describing the same historical period. If such a rich data source is available, it is possible to exploit document creation times, redundancy across documents, as well as back-references to earlier events in order to identify salient events. For instance, the start of the Iraq War in 2003 is mentioned frequently in a general news corpus, including in articles published years after the event took place. The high number of mentions suggests that the beginning of the Iraq War was an important historical event.

However, for most historical periods covered in history articles (e.g., Antiquity or the Middle Ages), such cues are not commonly available, as no news articles from these eras exist. Generating event timelines for arbitrary historical periods is therefore a much harder problem, which requires methods that rely less heavily on the types of rich, parallel and dense information contained in news clusters.

To investigate this problem, we approach timeline generation as a special single-document summarisation task. In other words, we assume that the information to be summarised is contained in a single history article, and that no further mentions of specific events exist externally. This is a realistic scenario, for instance, for a specialist article describing the history of music in Ancient China.

We introduce a method for selecting salient content in history articles of any subject area, as long as the events in the text are roughly ordered chronologically. The hypothesis is that knowledge of an text's subject area can help decide which content should be selected. Another intuition is that certain combinations of events should be avoided in a timeline. We therefore investigate ways of encouraging a balanced selection of content from all parts of the text.

2 Related work

Timeline extraction has mostly been explored in a multi-document summarisation setting using corpora of news articles (Tran et al., 2015; Swan and Allan, 2000; Yan et al., 2011; Chieu and Lee, 2004;

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2343–2349,
Austin, Texas, November 1-5, 2016.

Allan et al., 2001). This task definition allows the exploitation of features such as document creation times and headlines. The most important feature is redundancy between articles, which facilitates the identification of salient events.

A second important strand of work focuses on extracting *all* events from a single input text and anchoring them in time. The creation of the TimeML specification language (Pustejovsky et al., 2003) laid the foundations for the TempEval series of shared tasks (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013), in which systems had to identify TimeML events and temporal expressions in free-form text. Further subtasks included the normalisation of temporal expressions and the creation of links between events and temporal expressions. A further shared task investigated the use of TimeML annotation for the downstream task of question answering (Llorens et al., 2015). Kolomiyets et al. (2012) created a connected timeline for a text based on TimeML annotations; a dependency parser infers dependency structures between events. Finally, a recent SemEval task (Minard et al., 2015) explored the related problem of cross-document event ordering. Here, relevant events and temporal expressions concerning a single target entity of interest have to be identified in more than one input document.

Chasin et al. (2014) try to identify important events in single texts, but their approach is limited to articles on wars and battles, and the problem is not approached as a summarisation task. Their method is lightly supervised, using features such as the presence of negation or past tense verbs in the sentence, and TextRank (Mihalcea and Tarau, 2004) for identifying salient sentences. We use an improved version of this system as a baseline.

3 Overall approach

Our problem is that of finding an optimal sequence of events (of a given maximum length) in a given input article. We follow the literature on event extraction and use TimeML events (Pustejovsky et al., 2003). Most TimeML events are verbs, but some are nominalisations such as "invasion" or other event-like words such as "war". The use of TimeML events, aside from the practical advantage that commonly-available event extraction algorithms exist, allows us to evaluate content selection at the event rather than at the sentence level.

We assume that there are both local and global factors that determine which events should be contained in the timeline. Local factors reflect how important an event is in its own right. Global factors represent intuitions about which combinations of events should or should not be selected. Our approach, which is unsupervised, takes into account the factors described in what follows.

3.1 Date presence

Intuitively, we expect that many important events have a date attached to them, as authors tend to give the reader this information if it is available. This is true for all historical periods from prehistory onwards, since for most events at least an approximate date is known. We considered two alternatives: The simplest approach is to only use sentences that contain a date, regardless of where in the sentence the date is located. A more sophisticated alternative verifies that the date is syntactically attached to the event, such as in "Richelieu died in 1642". To identify such cases, we constructed a parse tree using the C&C dependency parser (Clark and Curran, 2007) and only considered a TimeML event to be "dated" if it is at most two outgoing dependencies away from a temporal expression. We used HeidelTime (Strötgen and Gertz, 2013), a the state-of-the-art temporal expression software package, to identify such temporal expressions.

3.2 Lexical cues

The key component we use to judge the importance of any event are lexical cues about the input text's subject area. Examples of such subject areas include `INVENTION` and `FOOD/DRINK`. The subject area of a text should give us prior knowledge about which types of events are likely to be important. For instance, we would expect that a timeline describing the history of a country should contain information about revolutions, invasions, elections and similar events, whereas a timeline about science will instead focus on discoveries, publications, and Nobel prizes.

To mine knowledge about such subject-area-specific preferences, we make use of Wikipedia as a background corpus. Only history-specific articles whose title starts with "History of" are considered.

We start by generating sets of all Wikipedia history articles belonging to a given subject area, e.g. $A_{\texttt{GPE}}$ or $A_{\texttt{INVENTION}}$. To do this, we make use of the Wikipedia category system. For instance, for constructing a set of articles for the subject area `FIELD OF SCIENCE`, we collected all history articles that belong to the Wikipedia category "History of science by discipline". For each subject area g, we then calculate a *preference score* for each word lemma l found in any of the articles in the corresponding list A_g, using the following formula:

$$sc(g,l) = \frac{\frac{freq(A_g,l)}{freq(A_g,*)}}{\frac{freq(*,l)}{freq(*,*)}}$$

where $freq(A_g, l)$ is the summed frequency of word lemma l in all documents belonging to subject area g, and "*" stands for any. The numerator denotes how often lemma l appears in the subject-area-specific set of articles A_g, normalised by the total number of tokens found in this set. The denominator is invariant across all subject areas. If the ratio is high, lemma l is more likely to appear in articles of subject area g than in Wikipedia overall, suggesting that it is typical for the given subject area.

For each event e in the input text, a *local importance score* $imp(e)$ is calculated as

$$imp(e) = \frac{\sum_{w \in R(e)} \frac{sc(g,l)}{1+dist(w_e,w)}}{N}$$

where $R(e)$ is a window of words around the word representing the event (including the event word w_e itself), and $dist(w_1, w_2)$ refers to the absolute distance in words between two words w_1 and w_2. $imp(e)$ is a weighted average of the preference scores of all words in a window. The intuition is that context words of the event word can also be expected to be indicative of the subject area (consider "publish a paper") in many cases. $1 + dist(w_e, w)$ is used as a distance penalty in order to give more importance to words that are closer to the event word w_e. N is a constant which normalises the score by the sum of all distance penalties, to account for cases where the event word occurs at the beginning or end of a sentence. Table 1 shows examples of words with high and low preference scores.

3.3 Temporal coverage

We would like to avoid cases where too many events are selected from a small portion of the document, even if all these events are relevant. For instance, an article might list all a country's elections of the past few years, while mentioning only very important elections in earlier time periods. In this case, knowing that elections are important in the history of a country is not helpful, since this would lead to insufficient coverage of the remaining events in the article. We therefore take into account global factors as well. We experiment with two different methods:

GPE	INVENTION	FOOD/DRINK
absolutism	gas-works	yerba
protectorate	reverse-angle	hamburger
serfdom	flashback	saffron
club	season	play
game	team	member
season	school	bear

Table 1: Words with high (top) and low (bottom) preference scores for three subject areas

Exploiting document structure. We select salient events from each section of the Wikipedia article in a round-robin fashion. The algorithm operates in a greedy fashion by selecting the most locally important remaining event for each section, until the desired timeline length has been reached.

Integer linear program. We use an integer linear program to encode the intuition that no two timeline entries should have the same year. The ILP maximises the following objective function for each article (E refers to the set of all dated events):

$$\sum_{e_i \in E} x_i \cdot imp(e_i) - \sum_{e_i \in E} \sum_{e_j \in E} b_{ij} \cdot pen(e_i, e_j)$$

subject to the constraints:

$$b_{ij} \leq x_i \; \forall i, j \in E$$
$$x_i + x_j - b_{ij} \leq 1 \; \forall i, j \in E$$
$$x_i \in \{0,1\} \; \forall i \in E \qquad b_{ij} \in \{0,1\} \; \forall i, j \in E$$
$$\sum_{e_i \in E} x_i = L_{max}$$

This is similar to the model used by McDonald (2007) for multi-document summarisation. The model tries to find a set of locally important events while discouraging the selection of events that have the same date. x_i is a variable denoting whether the corresponding event e_i has been selected. b_{ij} is a variable which is 1 if and only if both events i and j have been selected. $pen(e_i, e_j)$ is a penalty function that is 1 if the two events e_i and e_j have

the same date, otherwise 0. Each event was linked to the preceding temporal expression identified by HeidelTime; this heuristic was found to work well. The last constraint ensures that not more than L_{max} events are chosen, where L_{max} is the desired timeline length for the article considered.

4 Evaluation

For evaluating our algorithms, the methodology we introduced in (Bauer and Teufel, 2015) is used, along with the accompanying Cambridge Single-Document Timeline Corpus (CSDTC, version 2.0), which has been made publicly available[1].

4.1 Cambridge Single-Document Timeline Corpus

The CSDTC contains 10 articles from 3 subject areas: `GPE` (geo-political entities such as countries and cities), `FIELD OF SCIENCE` and `INVENTION`. To tune our algorithms, we constructed a development set of a further 30 annotated history articles from the subject areas in the CSDTC and one additional subject area (`FOOD/DRINK`). Due to the high annotation cost, only a single timeline creator was used. Important events were directly marked up in the source text (as opposed to the CSDTC, where timeline entries were written by hand), and exactly one HCU[2] was created per event. Using this development corpus, the window size of words considered for calculating local importance scores (cf. Section 3.2) was set to 3. We report the performance of all algorithms on both the development set and the test set (the CSDTC).

Although the number of subject areas in the two corpora is rather small owing to the considerable annotation effort, we believe that the resulting system would generalise rather well to other subject areas, were they added, as the subject areas in the corpus are very different in nature from each other. Care was taken when constructing the CSDTC to use a set of subject areas that is representative for human-written timelines on the Web.

[1]The corpus is available on the first author's website: `http://www.cl.cam.ac.uk/~smb89/form.html`

[2]As opposed to the CSDTC, HCUs in the development set always have a weight of 1, as only timeline writer was used.

4.2 Evaluation based on Historical Content Units

The evaluation is based on abstract ("deep") meaning units called *Historical Content Units* (HCUs). HCUs were derived on the basis of human-created timelines. Between 32 and 80 HCUs per article were annotated for the articles in the CSDTC.

Each HCU is weighted by the number of timeline creators who expressed its semantic content in their timelines. Because HCUs are linked to TimeML events in the surface text, it is possible to perform automatic deep evaluation without requiring any manual annotation of system summaries.

Algorithms are evaluated on a given input article using an adapted version of the pyramid score (Nenkova and Passonneau, 2004), which is calculated as the ratio between the sum of all rewards for HCUs chosen by the algorithm normalised by the maximum possible score $score_{max}$:

$$score = \frac{\sum_{h \in HCUs} w_h \cdot Cov(h,E,T)}{score_{max}}$$

where w_h is the weight of HCU h (a number between 1 and the number of annotators), E is the set of events in the article, T are the events in the system timeline, and the *coverage score* $Cov(h, E, T)$ is a number between 0 and 1 that indicates to what extent the events chosen by the algorithm jointly express the semantic content of HCU h. The basic version of $Cov(h, E, T)$ is defined as follows:

$$Cov(h, E, T) = min(1.0, \sum_{e_j \in E} v_{h,e_j} \cdot s(T, e_j))$$

where v_{h,e_j} is an *anchor weight* between 0 and 1 which denotes to what extent event e_j expresses the semantic content of HCU h, and $s(T, e)$ is a helper function that returns 1 if the set of selected events T includes event e, and 0 otherwise.

The coverage score for each HCU is calculated by summing up the anchor weights of those events that the algorithm has selected. A coverage score of 0 means that the events mentioned in the timeline do not express the HCU's semantic content at all, while a score of 1 occurs where the HCU's content is fully expressed by the timeline. Scores between 0 and 1 occur in a large number of cases. For instance, an HCU may express the fact that a country was invaded and destroyed. If the system timeline merely contains a TimeML event that refers to the invasion, it is assigned a coverage score of 0.5 for this HCU,

as it expresses only half of the HCU's semantic content. Where the sum exceeds 1, the coverage score is set to a hard upper limit of 1. This ensures that algorithms are not doubly rewarded for selecting multiple TimeML events expressing the same semantic content. The final formula we used to calculate coverage scores is slightly more complex, as some TimeML events in the CSDTC have been grouped together into *event groups*. A detailed description is given in the documentation of the corpus.

Pyramid scores are recall-based: The evaluation assumes a maximum number of timeline entries n, and the maximum possible score is the sum of the HCU weights of the n most highly weighted HCUs. The values for n are given in the CSDTC.

4.3 System and baselines

We report the performance of two systems. Both systems first remove all events that do not have a date, or whose date is too far away, as described in Section 3.1. Our first system ("ILP-based") selects events based on the integer linear program described, while the second system ("Round-robin") selects locally important events per section.

We have speculated above that dates are important for our task. We therefore compare against a date baseline which selects events randomly from the list of all dated events. We also compare against several modified versions of our method: To investigate the influence of the parser in identifying suitable dated events, we report the results for a simpler method which considers all events that have a date in the same sentence ("Round-robin, simple date criterion"). Two alternative systems select locally important events from all (not only dated) events ("Round-robin, without date criterion") or salient dated events from the entire article without considering document structure ("Local importance + date criterion").

The supervised baseline ("Chasin et al. (2014)") was re-implemented using LibSVM (Chang and Lin, 2011), and SVM parameters were tuned using grid search. 25 of the 30 articles were used for training and 5 for development. We improved their system by defining some of their sentence-level features at the event level. Probability estimates as described by Platt (2000) were used as importance scores.

System	Dev	Test
ILP-based	**0.22**▲	**0.30**▲
Round-robin	0.20▲	0.30▲
Round-robin w/o local importance	0.18	0.26
Local importance + date criterion	0.21▲	0.29
Round-robin, simple date criterion	0.19	0.25
Round-robin without date criterion	0.14	0.18
Date baseline	0.18	0.25
Chasin et al. (2014) (improved)	–	0.12
Random baseline	0.08	0.10

Table 2: Average pyramid scores across all articles (▲ = significantly better than the date baseline)

4.4 Results

The results in Table 2 show that only a combination of all three factors (date presence, local importance, coverage) results in a statistically significant improvement over the date baseline at $\alpha = 0.05$ according to Wilcoxon's signed-rank test on the test set. Both our systems perform comparably on the test set; removing any of the three components results in lower performance. Using a parser to identify dated events has a strong positive effect (see "Round-robin, simple date criterion"). Our system also outperforms the improved supervised baseline by a large margin. The fact that a completely unsupervised system performs best is encouraging, as training data for this task is very expensive to obtain. Our results suggest that it might be worth investigating other types of prior knowledge about the semantics of an input text in further research. The crucial advantage of such generic methods is that no texts on exactly the same topic are needed, which is a requirement with texts about niche topics.

5 Conclusion

We have introduced an unsupervised method for the challenging problem of timeline generation from single history articles, a scenario where parallel texts cannot be assumed to exist. Our method results in a significant improvement over a novel unsupervised baseline as well as an existing supervised approach.

Acknowledgments

The first author received financial support from Microsoft Research, St John's College Cambridge and the Cambridge University Computer Laboratory.

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal Summaries of New Topics. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 10–18, New York, NY, USA. ACM.

Sandro Bauer and Simone Teufel. 2015. A Methodology for Evaluating Timeline Generation Algorithms based on Deep Semantic Units. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 834–839.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Rachel Chasin, Daryl Woodward, Jeremy Witmer, and Jugal Kalita. 2014. Extracting and Displaying Temporal and Geospatial Entities from Articles on Historical Events. *Comput. J.*, 57(3):403–426.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query Based Event Extraction Along a Timeline. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 425–432, New York, NY, USA. ACM.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2012. Extracting Narrative Timelines as Temporal Dependency Structures. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 88–97.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TempEval - Evaluating Temporal Information Understanding with Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 792–800, Denver, Colorado, June. Association for Computational Linguistics.

Ryan McDonald. 2007. A Study of Global Inference Algorithms in Multi-document Summarization. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado, June. Association for Computational Linguistics.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: The Pyramid Method. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 145–152, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

J. Platt. 2000. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 1–11.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298.

Russell Swan and James Allan. 2000. TimeMine (Demonstration Session): Visualizing Automatically Constructed Timelines. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 393–, New York, NY, USA. ACM.

Giang Tran, Eelco Herder, and Katja Markert. 2015. Joint Graphical Models for Date Selection in Timeline Summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1598–1607, Beijing, China, July. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations.

In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary Timeline Summarization: A Balanced Optimization Framework via Iterative Substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 745–754, New York, NY, USA. ACM.

Encoding Temporal Information for Time-Aware Link Prediction

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang and **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{tingsong,tianyu0421,taoge,shalei,lisujian,chbb,szf}@pku.edu.cn

Abstract

Most existing knowledge base (KB) embedding methods solely learn from time-unknown fact triples but neglect the temporal information in the knowledge base. In this paper, we propose a novel time-aware KB embedding approach taking advantage of the happening time of facts. Specifically, we use temporal order constraints to model transformation between time-sensitive relations and enforce the embeddings to be temporally consistent and more accurate. We empirically evaluate our approach in two tasks of link prediction and triple classification. Experimental results show that our method outperforms other baselines on the two tasks consistently.

1 Introduction

Knowledge bases (KBs) such as Freebase (Bollacker et al., 2008) and YAGO (Fabian et al., 2007) play a pivotal role in many NLP related applications. KBs consist of facts in the form of triplets (e_i, r, e_j), indicating that head entity e_i and tail entity e_j is linked by relation r. Although KBs are large, they are far from complete. Link prediction is to predict relations between entities based on existing triplets, which can alleviate the incompleteness of current KBs. Recently a promising approach for this task called knowledge base embedding (Nickel et al., 2011; Bordes et al., 2011; Socher et al., 2013) aims to embed entities and relations into a continuous vector space while preserving certain information of the KB graph. TransE (Bordes et al., 2013) is a typical model considering relation vector as translating operations between head and tail vector, i.e., $\mathbf{e}_i + \mathbf{r} \approx \mathbf{e}_j$ when (e_i, r, e_j) holds.

Most existing KB embedding methods solely learn from time-unknown facts but ignore the useful temporal information in the KB. In fact, there are many temporal facts (or events) in the KB, e.g., (*Obama, wasBornIn, Hawaii*) happened at August 4, 1961. (*Obama, presidentOf, USA*) is true since 2009. Current KBs such as YAGO and Freebase store such temporal information either directly or indirectly. The happening time of time-sensitive facts may indicate special temporal order of facts and time-sensitive relations. For example, *(Einstein, wasBornIn, Ulm)* happened in 1879, *(Einstein, wonPrize, Nobel Prize)* happened in 1922, *(Einstein, diedIn, Princeton)* occurred in 1955. We can infer the temporal order of time-sensitive relations from all such kinds of facts: *wasBornIn* $\rightarrow$ *wonPrize* $\rightarrow$ *diedIn*. Traditional KB embedding models such as TransE often confuse relations such as *wasBornIn* and *diedIn* when predicting *(person,?,location)* because TransE learns only from time-unknown facts and cannot distinguish relations with similar semantic meaning. To make more accurate predictions, it is non-trivial for existing KB embedding methods to incorporate temporal order information.

This paper mainly focuses on incorporating the temporal order information and proposes a time-aware link prediction model. A new temporal dimension is added to fact triples, denoted as a quadruple: (e_i, r, e_j, t_r), indicating the fact happened at time t_r[1]. To make the embedding space compati-

[1] t_r is the absolute beginning time when the fact is true, e.g., "1961-08-04" for (*Obama, wasBornIn, Hawaii*).

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2350–2354,
Austin, Texas, November 1-5, 2016.

ble with the observed triple in the fact dimension, relation vectors behave as translations between entity vectors similarly as TransE models. To incorporate temporal order information between pair-wise temporal facts, we assume that prior time-sensitive relation vector can evolve into a subsequent time-sensitive relation vector through a temporal transition. For example, we have two temporal facts sharing the same *head entity*: (e_i, r_i, e_j, t_1) and (e_i, r_j, e_k, t_2) and the temporal order constraint $t_1 < t_2$, i.e., r_i happens before r_j, then we propose the assumption that prior relation $\mathbf{r}_i$ after temporal transition should lie close to subsequent relation $\mathbf{r}_j$, i.e., $\mathbf{r}_i\mathbf{M} \approx \mathbf{r}_j$, here matrix $\mathbf{M}$ captures the temporal order information between relations. In this way, both semantic and temporal information are embedded into a continuous vector space during learning. To the best of our knowledge, we are the first to consider such temporal information for KB embedding.

We evaluate our approach on public available datasets and our method outperforms state-of-the-art methods in the time-aware link prediction and triple classification tasks.

2 Time-Aware KB Embedding

Traditional KB embedding methods encode only observed fact triples but neglect temporal constraints between time-sensitive entities and facts. To deal with this limitation, we introduce *Time-Aware KB Embedding* which constrains the task by incorporating temporal constraints.

To consider the happening time of facts, we formulate a temporal order constraint as an optimization problem based on a manifold regularization term. Specially, temporal order of relations in time-sensitive facts should affect KB representation. If r_i and r_j share the same head entity e_i, and r_i occurs before r_j, then prior relation's vector r_i could evolve into subsequent relation's vector r_j in the temporal dimension.

To encode the transition between time-sensitive relations, we define a transition matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ between pair-wise temporal ordering relation pair (r_i, r_j). Our optimization requires that positive temporal ordering relation pairs should have lower scores (energies) than negative pairs, so we define a **temporal order** score function as

$$g(r_i, r_j) = \|\mathbf{r}_i\mathbf{M} - \mathbf{r}_j\|_1, \tag{1}$$

which is expected to be a low score when the relation pair is in chronological order, and high otherwise.

To make the embedding space compatible with the observed triples, we make use of the triple set Δ and follow the same strategy adopted in previous methods such as TransE.

$$f(e_i, r, e_j) = \|\mathbf{e}_i + \mathbf{r} - \mathbf{e}_j\|_1. \tag{2}$$

For each candidate triple, it requires positive triples to have lower scores than negative triples.

The optimization is to minimize the **joint** score function,

$$\begin{aligned} L = \sum_{x^+\in\Delta} \Big[\sum_{x^-\in\Delta'} [\gamma_1 + f(x^+) - f(x^-)]_+ \\ +\lambda \sum_{y^+\in\Omega_{e_i}, y^-\in\Omega'_{e_i}} [\gamma_2 + g(y^+) - g(y^-)]_+ \Big] \end{aligned} \tag{3}$$

where $x^+ = (e_i, r_i, e_j) \in \Delta$ is the positive triple (quad), $x^- = (e'_i, r_i, e'_j) \in \Delta'$ is corresponding the negative triple. $y^+ = (r_i, r_j) \in \Omega_{e_i}$ is the positive relation ordering pair with respect to (e_i, r_i, e_j, t_x). It's defined as

$$\begin{aligned} \Omega_{e_i} = \{(r_i, r_j) | (e_i, r_i, e_j, t_x) \in \Delta_\tau, \\ (e_i, r_j, e_k, t_y) \in \Delta_\tau, t_x < t_y\}, \end{aligned} \tag{4}$$

where r_i and r_j share the same head entity e_i, and $y^- = (r_j, r_i) \in \Omega'_{e_i}$ are the corresponding negative relation order pairs by inverse. In experiments, we enforce constrains as $\|\mathbf{e}_i\|_2 \leq 1$, $\|\mathbf{r}_i\|_2 \leq 1$, $\|\mathbf{r}_j\| \leq 1$ and $\|\mathbf{r}_i\mathbf{M}\|_2 \leq 1$.

The first term in Equation (3) enforces the resultant embedding space compatible with all the observed triples, and the second term further requires the space to be temporally consistent and more accurate. Hyperparameter λ makes a trade-off between the two cases. Stochastic gradient descent (in mini-batch mode) is adopted to solve the minimization problem.

3 Experiments

We adopt the same evaluation metrics for time-aware KB embedding in two tasks: link prediction (Bordes et al., 2013) and triple classification (Socher et al., 2013).

Dataset	#Rel	#Ent	#Train/#Valid/#Test	#Trip.	#Quad
YG15k	10	9513	13345/1320/1249	15914	15914
YG36k	10	9513	29757/3252/3058	36067	15914

Table 1: Statistics of data sets.

3.1 Datasets

We create two temporal datasets from YAGO2 (Hoffart et al., 2013), consisting of time-sensitive facts. In YAGO2, *MetaFacts* contains all happening time for facts. *DateFacts* contains all creation time for entities. First, to make a pure time-sensitive dataset where all facts have time annotations, we selected the subset of entities that have at least 2 mentions in *MetaFacts* and *DateFacts*. This resulted in 15,914 triples (quadruples) which were randomly split with the ratio shown in Table 1. This dataset is denoted *YG15k*. Second, to make a mixed dataset, we created *YG36k* where 50% facts have time annotations and 50% do not. We will release the data upon request.

3.2 Link Prediction

Link prediction is to complete the triple (h, r, t) when h, r or t is missing.

3.2.1 Entity Prediction

Evaluation protocol. For each test triple with missing head or tail entity, various methods are used to compute the scores for all candidate entities and rank them in descending order. We use two metrics for our evaluation as in (Bordes et al., 2013): the mean of correct entity ranks (Mean Rank) and the proportion of valid entities ranked in top-10 (Hits@10). As mentioned in (Bordes et al., 2013), the metrics are desirable but flawed when a corrupted triple exists in the KB. As a countermeasure, we filter out all these valid triples in the KB before ranking. We name the first evaluation set as *Raw* and the second as *Filter*.

Baseline methods. For comparison, we select translating methods such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014b) and TransR (Lin et al., 2015b) as our baselines. We then use time-aware embedding based on these methods to obtain the corresponding time-aware embedding models. A model with time-aware embedding is denoted as "tTransE" for example.

Implementation details. For all methods, we create 100 mini-batches on each data set. The dimension of the embedding n is set in the range of $\{20,50,100\}$, the margin γ_1 and γ_2 are set in the range $\{1,2,4,10\}$. The learning rate is set in the range $\{0.1,\ 0.01,\ 0.001\}$. The regularization hyperparameter λ is tuned in $\{10^{-1},10^{-2},10^{-3},10^{-4}\}$. The best configuration is determined according to the mean rank in validation set. The optimal configurations are n=100,γ_1=γ_2=4,λ=10^{-2}, learning rate is 0.001 and taking ℓ_1-norm.

	Mean Rank		Hits@1 (%)	
Metric	Raw	Filter	Raw	Filter
TransE	1.53	1.48	69.4	73.0
tTransE	**1.42**	**1.35**	**71.1**	**75.7**
TransH	1.51	1.37	70.5	72.2
tTransH	**1.38**	**1.30**	**74.6**	**76.9**
TransR	1.40	1.28	71.1	74.3
tTransR	**1.27**	**1.12**	**74.5**	**78.9**

Table 3: Evaluation results on relation prediction.

Results. Table 2 reports the results on the test set. From the results, we can see that time-aware embedding methods outperform all the baselines on all the data sets and with all the metrics. The improvements are usually quite significant. The Mean Rank drops by about 75%, and Hits@10 rises about 19% to 30%. This demonstrates the superiority and generality of our method. When dealing with sparse data YG15k, all the temporal information is utilized to model temporal associations and make the embeddings more accurate, so it obtains better improvement than mixing the time-unknown triples in YG36k.

3.2.2 Relation Prediction

Relation prediction aims to predict relations given two entities. Evaluation results are shown in Table 3 on only YG15K due to limited space, where we report Hits@1 instead of Hits@10. Example prediction results for TransE and tTransE are compared in Table 4. For example, when testing *(Billy_Hughes,?,London,1862)*, it's easy for TransE to mix relations *wasBornIn* and *diedIn* because they act similarly for a person and a place. But known that *(Billy_Hughes, isAffiliatedTo, National_Labor_Party)* happened in 1916, and tTransE have learnt temporal order that *wasBornIn*→*isAffiliatedTo*→*diedIn*, so the regularization term $|\mathbf{r}_{born}\mathbf{T} - \mathbf{r}_{affiliated}|$ is smaller than $|\mathbf{r}_{died}\mathbf{T} - \mathbf{r}_{affiliated}|$, so correct answer *wasBornIn* ranks higher than *diedIn*.

Dataset	YG15k				YG36k			
Metric	MeanRank		Hits@10(%)		MeanRank		Hits@10(%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	990	971	26.6	29.5	179	163	65.7	75.6
tTransE	**235**	**233**	**35.4**	**36.1**	**60**	**55**	**76.1**	**82.8**
TransH	986	966	25.7	28.0	174	158	65.3	77.8
tTransH	**232**	**230**	**36.1**	**37.2**	**61**	**54**	**76.6**	**82.9**
TransR	976	955	29.5	30.2	175	153	68.3	80.1
tTransR	**228**	**221**	**37.3**	**38.4**	**55**	**46**	**79.5**	**84.2**

Table 2: Evaluation results on link prediction.

Testing quad	TransE predictions	tTransE predictions
(Billy_Hughes,?,London,1862)	diedIn,**wasBornIn**	**wasBornIn**,diedIn
(John_Schoenherr,?,Caldecott_Medal,1988)	owns,**hasWonPrize**	**hasWonPrize**,owns
(John_G._Thompson,?,University_of_Cambridge,1961)	graduatedFrom,**worksAt**	**worksAt**,graduatedFrom
(Tommy_Douglas,?,New_Democratic_Party,1961)	isMarriedTo,**isAffiliatedTo**	**isAffiliatedTo**,worksAt

Table 4: Example results of relation prediction in descending order. Correct predictions are in **bold**.

3.3 Triple Classification

Triple classification aims to judge whether an unseen triple is correct or not.

Evaluation protocol. We follow the same evaluation protocol used in Socher et al. (2013). To create labeled data for classification, for each triple in the test and validation sets, we construct a corresponding negative triple by randomly corrupting the entities. To corrupt a position (head or tail), only entities that have appeared in that position are allowed. During triple classification, a triple is predicted as positive if the score is below a relation-specific threshold δ_r; otherwise as negative. We report averaged accuracy on the test sets.

Implementation details. We use the same hyperparameter settings as in the link prediction task. The relation-specific threshold δ_r is determined by maximizing averaged accuracy on the validation sets.

Results. Table 5 reports the results on the test sets. The results indicate that time-aware embedding outperforms all the baselines consistently. Temporal order information may help to distinguish positive and negative triples as different head entities may have different temporally associated relations. If the temporal order is the same with most facts, the regularization term helps it get lower energies and vice versa.

Datasets	YG15K	YG36K
TransE	63.9	71.9
tTransE	**75.0**	**82.7**
TransH	63.4	72.1
tTransH	**75.1**	**82.3**
TransR	64.5	74.9
tTransR	**78.5**	**83.9**

Table 5: Evaluation results on triple classification (%).

4 Related Work

Many models have been proposed for KB embedding (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013). External information is employed to improve KB embedding such as text (Riedel et al., 2013; Wang et al., 2014a; Zhao et al., 2015), entity type and relationship domain (Guo et al., 2015; Chang et al., 2014), and relation path (Lin et al., 2015a; Gu et al., 2015). However, these methods solely rely on triple facts but neglect temporal order constraints between facts. Temporal information such as relation ordering in text has been explored (Talukdar et al., 2012; Chambers et al., 2014; Bethard, 2013; Cassidy et al., 2014; Chambers et al., 2007; Chambers and Jurafsky, 2008). This paper proposes a time-aware embedding approach that employs temporal order constraints to improve KB embedding.

5 Conclusion and Future Work

In this paper, we propose a general time-aware KB embedding, which incorporates creation time of entities and imposes temporal order constraints on the geometric structure of the embedding space and enforce it to be temporally consistent and accurate. As future work: (1) We will incorporate the valid time of facts. (2) Some time-sensitive facts lack temporal information in YAGO2, we will mine such temporal information from texts.

Acknowledgments

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The contact author for this paper is Baobao Chang and Zhifang Sui.

References

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *ACL*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. *ACL*, 94305:789–797.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*, pages 1568–1579.

MS Fabian, K Gjergji, and W Gerhard. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference, WWW*, pages 697–706.

Kelvin Gu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 84–94.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012. Acquiring temporal constraints between relations. In *CIKM*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Representation learning for measuring entity relatedness with rich information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1412–1418. AAAI Press.

Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning

Karthik Narasimhan
CSAIL, MIT
karthikn@mit.edu

Adam Yala
CSAIL, MIT
adamyala@mit.edu

Regina Barzilay
CSAIL, MIT
regina@csail.mit.edu

Abstract

Most successful information extraction systems operate with access to a large collection of documents. In this work, we explore the task of acquiring and incorporating external evidence to improve extraction accuracy in domains where the amount of training data is scarce. This process entails issuing search queries, extraction from new sources and reconciliation of extracted values, which are repeated until sufficient evidence is collected. We approach the problem using a reinforcement learning framework where our model learns to select optimal actions based on contextual information. We employ a deep Q-network, trained to optimize a reward function that reflects extraction accuracy while penalizing extra effort. Our experiments on two databases – of shooting incidents, and food adulteration cases – demonstrate that our system significantly outperforms traditional extractors and a competitive meta-classifier baseline.[1]

1 Introduction

In many realistic domains, information extraction (IE) systems require exceedingly large amounts of annotated data to deliver high performance. Increases in training data size enable models to handle robustly the multitude of linguistic expressions that convey the same semantic relation. Consider, for instance, an IE system that aims to identify entities such as the perpetrator and the number of victims in a shooting incident (Figure 1). The document does not explicitly mention the shooter (*Scott Westerhuis*), but instead refers to him as a suicide victim. Extraction of the number of fatally shot victims is similarly difficult, as the system needs to infer that *"A couple and four children"* means *six people*. Even a large annotated training set may not provide sufficient coverage to capture such challenging cases.

ShooterName: Scott Westerhuis
NumKilled: 6

A couple and four children found dead in their burning South Dakota home had been shot in an apparent murder-suicide, officials said Monday.
...
Scott Westerhuis's cause of death was "shotgun wound with manner of death as suspected suicide," it added in a statement.

Figure 1: Sample news article on a shooting case. Note how the article contains both the name of the shooter and the number of people killed but both pieces of information require complex extraction schemes.

In this paper, we explore an alternative approach for boosting extraction accuracy, when a large training corpus is not available. Instead, the proposed method utilizes external information sources to resolve ambiguities inherent in text interpretation. Specifically, our strategy is to find other documents that contain the information sought, expressed in a form that a basic extractor can "understand". For instance, Figure 2 shows two other articles describing the same event, wherein the entities of interest

[1]Code is available at http://people.csail.mit.edu/karthikn/rl-ie/

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2355–2365,
Austin, Texas, November 1-5, 2016.

The **six** members of a South Dakota family found dead in the ruins of their burned home were fatally shot, with one death believed to be a suicide, authorities said Monday.

AG Jackley says all evidence supports the story he told based on preliminary findings back in September: **Scott Westerhuis** shot his wife and children with a shotgun, lit his house on fire with an accelerant, then shot himself with his shotgun.

Figure 2: Two other articles on the same shooting case. The first article clearly mentions that six people were killed. The second one portrays the shooter in an easily extractable form.

– the number of people killed and the name of the shooter – are expressed explicitly. Processing such stereotypical phrasing is easier for most extraction systems, compared to analyzing the original source document. This approach is particularly suitable for extracting information from news where a typical event is covered by multiple news outlets.

The challenges, however, lie in (1) performing *event coreference* (i.e. retrieving suitable articles describing the same incident) and (2) reconciling the entities extracted from these different documents. Querying the web (using the source article's title for instance) often retrieves documents about other incidents with a tangential relation to the original story. For example, the query "*4 adults, 1 teenager shot in west Baltimore 3 april 2015*" yields only two relevant articles among the top twenty results on Bing search, while returning other shooting events at the same location. Moreover, the values extracted from these different sources require resolution since some of them might be inaccurate.

One solution to this problem would be to perform a single search to retrieve articles on the same event and then reconcile values extracted from them (say, using a *meta-classifier*). However, if the confidence of the new set of values is still low, we might wish to perform further queries. Thus, the problem is inherently sequential, requiring alternating phases of querying to retrieve articles and integrating the extracted values.

We address these challenges using a Reinforcement Learning (RL) approach that combines query formulation, extraction from new sources, and value reconciliation. To effectively select among possible actions, our state representation encodes information about the current and new entity values along with the similarity between the source article and the newly retrieved document. The model learns to select good actions for both article retrieval and value reconciliation in order to optimize the reward function, which reflects extraction accuracy and includes penalties for extra moves. We train the RL agent using a Deep Q-Network (DQN) (Mnih et al., 2015) that is used to predict both querying and reconciliation choices simultaneously. While we use a maximum entropy model as the base extractor, this framework can be inherently applied to other extraction algorithms.

We evaluate our system on two datasets where available training data is inherently limited. The first dataset is constructed from a publicly available database of mass shootings in the United States. The database is populated by volunteers and includes the source articles. The second dataset is derived from a FoodShield database of illegal food adulterations. Our experiments demonstrate that the final RL model outperforms basic extractors as well as a meta-classifier baseline in both domains. For instance, in the *Shootings* domain, the average accuracy improvement over the meta-classifier is 7%.

2 Related Work

Open Information Extraction Existing work in open IE has used external sources from the web to improve extraction accuracy and coverage (Agichtein and Gravano, 2000; Etzioni et al., 2011; Fader et al., 2011; Wu and Weld, 2010). Such research has focused on identifying multiple instances of the same relation, independent of the context in which this information appears. In contrast, our goal is to extract information from additional sources about a specific event described in a source article. Therefore, the novel challenge of our task resides in performing event coreference (Lee et al., 2012; Bejan and Harabagiu, 2014) (i.e identifying other sources describing the same event) while simultaneously reconciling extracted information. Moreover, relations typically considered by open IE systems have significantly higher coverage in online documents than a specific incident described in

a few news sources. Hence, we require a different mechanism for finding and reconciling online information.

Entity linking, multi-document extraction and event coreference Our work also relates to the task of multi-document information extraction, where the goal is to connect different mentions of the same entity across input documents (Mann and Yarowsky, 2005; Han et al., 2011; Durrett and Klein, 2014). Since this setup already includes multiple input documents, the model is not required to look for additional sources or decide on their relevance. Also, while the set of input documents overlap in terms of entities mentioned, they do not necessarily describe the same event. Given these differences in setup, the challenges and opportunities of the two tasks are distinct.

Knowledge Base Completion and Online Search Recent work has explored several techniques to perform Knowledge Base Completion (KBC) such as vector space models and graph traversal (Socher et al., 2013; Yang et al., 2014; Gardner et al., 2014; Neelakantan et al., 2015; Guu et al., 2015). Though our work also aims at increasing extraction recall for a database, traditional KBC approaches do not require searching for additional sources of information. West et al. (2014) explore query reformulation in the context of KBC. Using existing search logs, they learn how to formulate effective queries for different types of database entries. Once query learning is completed, the model employs several selected queries, and then aggregates the results based on retrieval ranking. This approach is complementary to the proposed method, and can be combined with our approach if search logs are available.

Kanani and McCallum (2012) also combine search and information extraction. In their task of faculty directory completion, the system has to find documents from which to extract desired information. They employ reinforcement learning to address computational bottlenecks, by minimizing the number of queries, document downloads and extraction action. The extraction accuracy is not part of this optimization, since the baseline IE system achieves high performance on the relations of interest. Hence, given different design goals, the two RL formulations are very different. Our approach is also close in spirit to the AskMSR system (Banko et al., 2002) which aims at using information redundancy on the web to better answer questions. Though our goal is similar, we learn to query and consolidate the different sources of information instead of using predefined rules. Several slot-filling methods have experimented with query formulation over web-based corpora to populate knowledge bases (Surdeanu et al., 2010; Ji and Grishman, 2011).

3 Framework

We model the information extraction task as a markov decision process (MDP), where the model learns to utilize external sources to improve upon extractions from a source article (see Figure 3). The MDP framework allows us to dynamically incorporate entity predictions while also providing flexibility to choose the type of articles to extract from. At each step, the system has to reconcile extracted values from a related article (e_{new}) with the current set of values (e_{cur}), and decide on the next query for retrieving more articles.

We represent the MDP as a tuple $\langle S, A, T, R \rangle$, where $S = \{s\}$ is the space of all possible states, $A = \{a = (d, q)\}$ is the set of all actions, $R(s, a)$ is the reward function, and $T(s'|s, a)$ is the transition function. We describe these in detail below.

States The state s in our MDP consists of the extractor's confidence in predicted entity values, the context from which the values are extracted and the similarity between the new document and the original one. We represent the state as a continuous real-valued vector (Figure 3) incorporating these pieces of information:

1. Confidence scores of current and newly extracted entity values.
2. One-hot encoding of matches between current and new values.
3. Unigram/tf-idf counts[2] of context words. These are words that occur in the neighborhood of the entity values in a document (e.g. the words *which*, *left*, *people* and *wounded* in the phrase "*which left 5 people wounded*").
4. *tf-idf* similarity between the original article and the new article.

[2]Counts are computed on the documents used to train the basic extraction system.

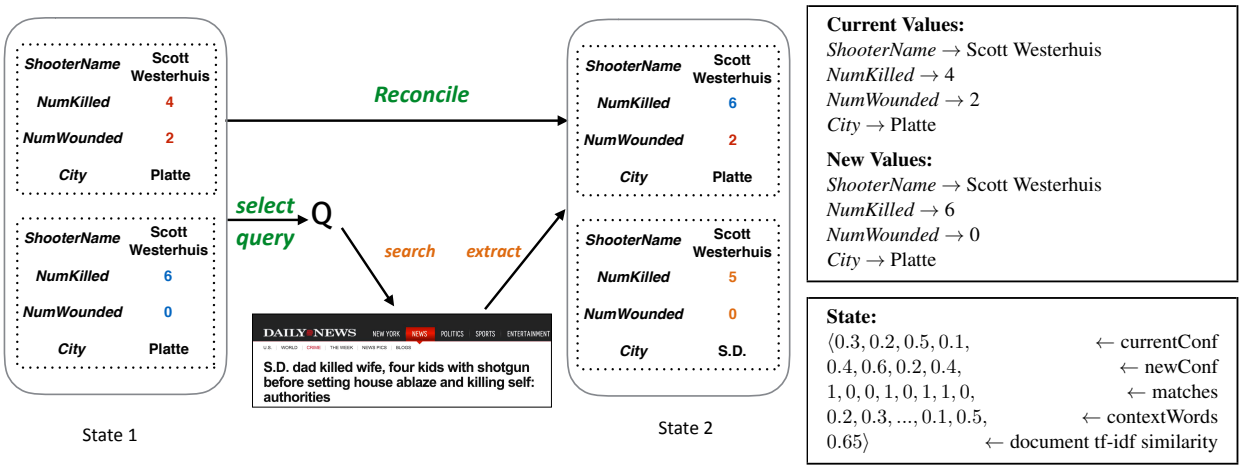


Figure 3: Left: Illustration of a transition in the MDP – the top box in each state shows the current entities and the bottom one consists of the new entities extracted from a downloaded article on the same event. **Right:** Sample state representation (bottom) in the MDP based on current and new values of entities (top). *currentConf*: confidence scores of current entities, *newConf*: confidence scores of new entities, *contextWords*: tf-idf counts of context words.

Actions At each step, the agent is required to take two actions - a reconciliation decision d and a query choice q. The decision d on the newly extracted values can be one of the following types: (1) accept a specific entity's value (one action per entity)[3], (2) accept all entity values, (3) reject all values or (4) stop. In cases 1-3, the agent continues to inspect more articles, while the episode ends if a stop action (4) is chosen. The current values and confidence scores are simply updated with the accepted values and the corresponding confidences.[4] The choice q is used to choose the next query from a set of automatically generated alternatives (details below) in order to retrieve the next article.

Rewards The reward function is chosen to maximize the final *extraction accuracy* while minimizing the number of queries. The accuracy component is calculated using the difference between the accuracy of the current and the previous set of entity values:

$$R(s,a) = \sum_{\text{entity } j} \text{Acc}(e^j_{cur}) - \text{Acc}(e^j_{prev})$$

There is a negative reward per step to penalize the agent for longer episodes.

[3]No entity-specific features are used for action selection.

[4]We also experiment with other forms of value reconciliation. See Section 5 for details.

Queries The queries are based on automatically generated templates, created using the title of an article along with words[5] most likely to co-occur with each entity type in the training data. Table 1 provides some examples – for instance, the second template contains words such as *arrested* and *identified* which often appear around the name of the shooter.

⟨*title*⟩
⟨*title*⟩ + (police | identified | arrested | charged)
⟨*title*⟩ + (killed | shooting | injured | dead | people)
⟨*title*⟩ + (injured | wounded | victim)
⟨*title*⟩ + (city | county | area)

Table 1: Examples of different query templates for web search for articles on mass shootings. The \| symbol represents logical OR. The last 4 queries contain context words around values for entity types ShooterName, NumKilled, NumWounded and City, respectively. At query time, ⟨*title*⟩ is replaced by the source article's title.

We use a search engine to query the web for articles on the same event as the source article and retrieve the top k links per query.[6] Documents that are more than a month older than the original article are filtered out of the search results.

Transitions Each episode starts off with a single source article x_i from which an initial set of entity

[5]Stop words, numeric terms and proper nouns are filtered.

[6]We use k=20 in our experiments.

values are extracted. The subsequent steps in the episode involve the extra articles, downloaded using different types of query formulations based on the source article. A single transition in the episode consists of the agent being given the state s containing information about the current and new set of values (extracted from a single article) using which the next action $a = (d, q)$ is chosen. The transition function $T(s'|s, a)$ incorporates the reconciliation decision d from the agent in state s along with the values from the next article retrieved using query q and produces the next state s'. The episode stops whenever d is a stop decision.

Algorithm 1 details the entire MDP framework for the training phase. During the test phase, each source article is handled only once in a single episode (lines 8-23).

Algorithm 1 MDP framework for Information Extraction (Training Phase)

1: Initialize set of original articles X
2: **for** $x_i \in X$ **do**
3: **for** each query template T^q **do**
4: Download articles with query $T^q(x_i)$
5: Queue retrieved articles in Y_i^q
6: **for** $epoch = 1, M$ **do**
7: **for** $i = 1, |X|$ **do** //episode
8: Extract entities e_0 from x_i
9: $e_{cur} \leftarrow e_0$
10: q $\leftarrow 0$, r $\leftarrow 0$ //query type, reward
11: **while** Y_i^q not empty **do**
12: Pop next article y from Y_i^q
13: Extract entities e_{new} from y
14: Compute tf-idf similarity $\mathcal{Z}(x_i, y)$
15: Compute context vector $\mathcal{C}(y)$
16: Form state s using e_{cur}, e_{new}, $\mathcal{Z}(x_i, y)$ and $\mathcal{C}(y)$
17: Send (s, r) to agent
18: Get decision d, query q from agent
19: **if** q == "end_episode" **then break**
20: $e_{prev} \leftarrow e_{cur}$
21: $e_{cur} \leftarrow$ *Reconcile*(e_{cur}, e_{new}, d)
22: $r \leftarrow \sum_{\text{entity } j} \text{Acc}(e_{cur}^j) - \text{Acc}(e_{prev}^j)$
23: Send (s_{end}, r) to agent

4 Reinforcement Learning for Information Extraction

In order to learn a good policy for an agent, we utilize the paradigm of reinforcement learning (RL). The MDP described in the previous section can be viewed in terms of a sequence of transitions (s, a, r, s'). The agent typically utilizes a state-action value function $Q(s, a)$ to determine which action a to perform in state s. A commonly used technique for learning an optimal value function is Q-learning (Watkins and Dayan, 1992), in which the agent iteratively updates $Q(s, a)$ using the rewards obtained from episodes. The updates are derived from the recursive Bellman equation (Sutton and Barto, 1998) for the optimal Q:

$$Q_{i+1}(s, a) = \mathrm{E}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a]$$

Here, $r = R(s, a)$ is the reward and γ is a factor discounting the value of future rewards and the expectation is taken over all transitions involving state s and action a.

Since our problem involves a continuous state space S, we use a deep Q-network (DQN) (Mnih et al., 2015) as a function approximator $Q(s, a) \approx Q(s, a; \theta)$. The DQN, in which the Q-function is approximated using a deep neural network, has been shown to learn better value functions than linear approximators (Narasimhan et al., 2015; He et al., 2015) and can capture non-linear interactions between the different pieces of information in our state.

We use a DQN consisting of two linear layers (20 hidden units each) followed by rectified linear units (ReLU), along with two separate output layers.[7] The network takes the continuous state vector s as input and predicts $Q(s, d)$ and $Q(s, q)$ for reconciliation decisions d and query choices q simultaneously using the different output layers (see Supplementary material for the model architecture).

Parameter Learning The parameters θ of the DQN are learnt using stochastic gradient descent with RMSprop (Tieleman and Hinton, 2012). Each parameter update aims to close the gap between the $Q(s_t, a_t; \theta)$ predicted by the DQN and the expected Q-value from the Bellman equation, $r_t + \gamma \max_a Q(s_{t+1}, a; \theta)$. Following Mnih et al. (2015), we make use of a (separate) target Q-network to calculate the expected Q-value, in order

[7] We did not observe significant differences with additional linear layers or the choice of non-linearity (Sigmoid/ReLU).

Algorithm 2 Training Procedure for DQN agent with ϵ-greedy exploration

Initialize experience memory $\mathcal{D}$
Initialize parameters θ randomly
for $episode = 1, M$ **do**
  Initialize environment and get start state s_1
  for $t = 1, N$ **do**
    if $random() < \epsilon$ **then**
      Select a random action a_t
    else
      Compute $Q(s_t, a)$ for all actions a
      Select $a_t = \text{argmax}\ Q(s_t, a)$
    Execute action a_t and observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random mini batch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    $y_j = \begin{cases} r_j, & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta_t), & \text{else} \end{cases}$
    Perform gradient descent step on the loss $\mathcal{L}(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
    if $s_{t+1} == s_{end}$ **then break**

to have 'stable updates'. The target Q-network is periodically updated with the current parameters θ. We also make use of an experience replay memory $\mathcal{D}$ to store transitions. To perform updates, we sample a batch of transitions $(\hat{s}, \hat{a}, \hat{s}', r)$ at random from $\mathcal{D}$ and minimize the loss function[8]:

$$\mathcal{L}(\theta) = \mathrm{E}_{\hat{s},\hat{a}}[(y - Q(\hat{s}, \hat{a}; \theta))^2]$$

where $y = r + \gamma \max_{a'} Q(\hat{s}', a'; \theta_t)$ is the target Q-value. The learning updates are made every training step using the following gradients:

$$\nabla_\theta \mathcal{L}(\theta) = \mathrm{E}_{\hat{s},\hat{a}}[2(y - Q(\hat{s}, \hat{a}; \theta)) \nabla_\theta Q(\hat{s}, \hat{a}; \theta)]$$

Algorithm 2 details the DQN training procedure.

5 Experimental Setup

Data We perform experiments on two different datasets. For the first set, we collected data from the Gun Violence archive,[9] a website tracking shootings in the United States. The data contains a news article on each shooting and annotations for (1) the name of the shooter, (2) the number of people killed, (3) the number of people wounded, and (4) the city where the incident took place. We consider these as the entities of interest, to be extracted from the articles. The second dataset we use is the Foodshield EMA database[10] documenting adulteration incidents since 1980. This data contains annotations for (1) the affected food product, (2) the adulterant and (3) the location of the incident. Both datasets are classic examples where the number of recorded incidents is insufficient for large-scale IE systems to leverage.

Number	Shootings			Adulteration		
	Train	**Test**	**Dev**	**Train**	**Test**	**Dev**
Source articles	306	292	66	292	148	42
Downloaded articles	8201	7904	1628	7686	5333	1537

Table 2: Stats for *Shootings* and *Adulteration* datasets

For each source article in the above databases, we download extra articles (top 20 links) using the Bing Search API[11] with different automatically generated queries. We use only the source articles from the *train* portion to learn the parameters of the base extractor. The entire *train* set with downloaded articles is used to train the DQN agent and the meta-classifier baseline (described below). All parameters are tuned on the *dev* set. For the final results, we train the models on the combined train and dev sets and use the entire *test* set (source + downloaded articles) to evaluate. Table 2 provides data statistics.

Extraction model We use a maximum entropy classifier as the base extraction system, since it provides flexibility to capture various local context features and has been shown to perform well for information extraction (Chieu and Ng, 2002). The classifier is used to tag each word in a document as one of the entity types or not (e.g. {*ShooterName, NumKilled, NumWounded, City, Other*} in the *Shootings* domain). Then, for each tag except *Other*, we choose the mode of the values to obtain the set of entity extractions from the article.[12] Features used in the classifier are provided in the Supplementary material.

The features and context window $c = 4$ of neighboring words are tuned to maximize performance on a dev set. We also experimented with a conditional random field (CRF) (with the same features) for the sequence tagging (Culotta and McCallum, 2004)

[8] The expectation is over the transitions sampled uniformly at random from $\mathcal{D}$.

[9] www.shootingtracker.com/Main_Page

[10] www.foodshield.org/member/login/

[11] www.bing.com/toolbox/bingsearchapi

[12] We normalize numerical words (e.g. "one" to "1") before taking the mode.

but obtained worse empirical performance (see Section 6). The parameters of the base extraction model are not changed during training of the RL model.

Evaluation We evaluate the extracted entity values against the gold annotations and report the corpus-level average accuracy on each entity type. For entities like ShooterName, the annotations (and the news articles) often contain multiple names (first and last) in various combinations, so we consider retrieving either name as a successful extraction. For all other entities, we look for exact matches.

Baselines We explore 4 types of baselines:

Basic extractors: We use the CRF and the Maxent classifier mentioned previously.

Aggregation systems: We examine two systems that perform different types of value reconciliation. The first model (*Confidence*) chooses entity values with the highest confidence score assigned by the base extractor. The second system (*Majority*) takes a majority vote over all values extracted from these articles. Both methods filter new entity values using a threshold τ on the cosine similarity over the tf-idf representations of the source and new articles.

Meta-classifer: To demonstrate the importance of modeling the problem in the RL framework, we consider a meta-classifier baseline. The classifier operates over the same input state space and produces the same set of reconciliation decisions $\{d\}$ as the DQN. For training, we use the original source article for each event along with a related downloaded article to compute the state. If the downloaded article has the correct value and the original one does not, we label it as a positive example for that entity class. If multiple such entity classes exist, we create several training instances with appropriate labels, and if none exist, we use the label corresponding to the *reject all* action. For each *test* event, the classifier is used to provide decisions for all the downloaded articles and the final extraction is performed by aggregating the value predictions using the *Confidence*-based scheme described above.

Oracle: Finally, we also have an ORACLE score which is computed assuming perfect reconciliation and querying decisions on top of the Maxent base extractor. This helps us analyze the contribution of the RL system in isolation of the inherent limitations of the base extractor.

RL models We perform experiments using three variants of RL agents – (1) *RL-Basic*, which performs only reconciliation decisions[13], (2) *RL-Query*, which takes only query decisions with the reconciliation strategy fixed (similar to Kanani and McCallum (2012)), and (3) *RL-Extract*, our full system incorporating both reconciliation and query decisions.

We train the models for 10000 steps every epoch using the Maxent classifier as the base extractor, and evaluate on the entire *test* set every epoch. The final accuracies reported are averaged over 3 independent runs; each run's score is averaged over 20 epochs after 100 epochs of training. The penalty per step is set to -0.001. For the DQN, we use the dev set to tune all parameters. We used a replay memory $\mathcal{D}$ of size 500k, and a discount (γ) of 0.8. We set the learning rate to 2.5E^{-5}. The ϵ in ϵ-greedy exploration is annealed from 1 to 0.1 over 500k transitions. The target-Q network is updated every 5k steps.

6 Results

Performance Table 3 demonstrates that our system (RL-Extract) obtains a substantial gain in accuracy over the basic extractors on all entity types over both domains. For instance, RL-Extract is 11.4% more accurate than the basic Maxent extractor on City and 7.1% better on NumKilled, while also achieving gains of more than 5% on the other entities on the *Shootings* domain. The gains on the *Adulteration* dataset are also significant, up to a 11.5% increase on the Location entity.

We can also observe that simple aggregation schemes like the *Confidence* and *Majority* baselines don't handle the complexity of the task well. RL-Extract outperforms these by 7.2% on *Shootings* and 5% on *Adulteration* averaged over all entities. Further, the importance of sequential decision-making is established by RL-Extract performing significantly better than the meta-classifier (7.0% on *Shootings* over all entities). This is also due to the fact that the meta-classifier aggregates over the entire set of extra documents, including the long tail of noisy, irrelevant documents. Finally, we see the advantage of enabling the RL system to select queries as our full model RL-Extract obtains significant im-

[13] Articles are presented to the agent in a round-robin fashion from the different query lists.

System	Shootings				Adulteration		
	ShooterName	**NumKilled**	**NumWounded**	**City**	**Food**	**Adulterant**	**Location**
CRF extractor	9.5	65.4	64.5	47.9	41.2	28.3	51.7
Maxent extractor	45.2	69.7	68.6	53.7	56.0	52.7	67.8
Confidence Agg. (τ)	45.2 (0.6)	70.3 (0.6)	72.3 (0.6)	55.8 (0.6)	56.0 (0.8)	54.0 (0.8)	69.2 (0.6)
Majority Agg. (τ)	47.6 (0.6)	69.1 (0.9)	68.6 (0.9)	54.7 (0.7)	56.7 (0.5)	50.6 (0.95)	72.0 (0.4)
Meta-classifier	45.2	70.7	68.4	55.3	55.4	52.7	72.0
RL-Basic	45.2	71.2	70.1	54.0	57.0	55.1	76.1
RL-Query (conf)	39.6	66.6	69.4	44.4	39.4	35.9	66.4
RL-Extract	**50.0**	**77.6***	**74.6***	**65.6***	**59.6***	**58.9***	**79.3***
ORACLE	57.1	86.4	83.3	71.8	64.8	60.8	83.9

Table 3: Accuracy of various baselines (italics), our system (DQN) and the Oracle on *Shootings* and *Adulteration* datasets. **Agg.** refers to aggregation baselines. Bold indicates best system scores. *statistical significance of $p < 0.0005$ vs basic Maxent extractor using the Student-t test. Numbers in parentheses indicate the optimal threshold (τ) for the aggregation baselines. Confidence-based reconciliation was used for RL-Query.

Entity	System: Value	Example
ShooterName	**Basic**: Stewart	A source tells Channel 2 Action News that Thomas Lee has been arrested in Mississippi ... Sgt . Stewart Smith, with the Troup County Sheriff's office, said.
	RL-Extract: Lee	Lee is accused of killing his wife, Christie; ...
NumKilled	**Basic**: 0	Shooting leaves 25 year old Pittsfield man dead , 4 injured
	RL-Extract: 1	One man is dead after a shooting Saturday night at the intersection of Dewey Avenue and Linden Street.
NumWounded	**Basic**: 0	Three people are dead and a fourth is in the hospital after a murder suicide
	RL-Extract: 1	3 dead, 1 injured in possible Fla. murder-suicide
City	**Basic**: Englewood	A 2 year old girl and four other people were wounded in a shooting in West Englewood Thursday night, police said
	RL-Extract: Chicago	At least 14 people were shot across Chicago between noon and 10:30 p.m. Thursday. The last shooting left five people wounded.

Table 4: Sample outputs (along with corresponding article snippets) on the *Shootings* domain showing correct predictions from RL-Extract where the basic extractor (Maxent) fails.

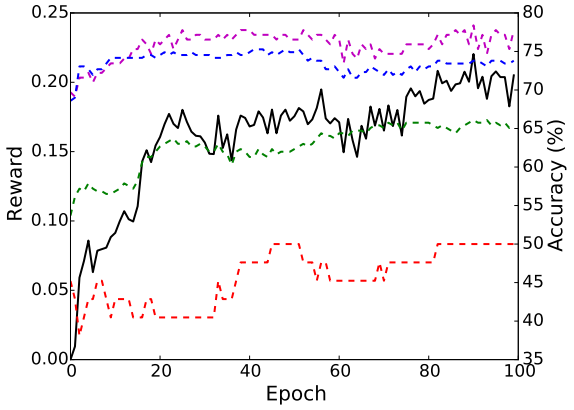


Figure 4: Evolution of average reward (solid black) and accuracy on various entities (dashed lines; red=*ShooterName*, magenta=*NumKilled*, blue=*NumWounded*, green=*City*) on the *test* set of the *Shootings* domain.

provements over RL-Basic on both domains. The full model also outperforms RL-Query, demonstrating the importance of performing both query selection and reconciliation in a joint fashion.

Figure 4 shows the learning curve of the agent by measuring reward on the test set after each training epoch. The reward improves gradually and the accuracy on each entity increases simultaneously. Table 4 provides some examples where our model is able to extract the right values when the baseline fails. One can see that in most cases this is due to the model making use of articles with prototypical language or articles containing the entities in readily extractable form.

Analysis We also analyze the importance of different reconciliation schemes, rewards and context-vectors in RL-Extract on the Shootings domain (Table 5). In addition to simple replacement (Re-

Reconciliation (RL-Extract)	Context	Reward	Accuracy				Steps
			S	K	W	C	
Confidence	tf-idf	Step	47.5	71.5	70.4	60.1	8.4
Majority	tf-idf	Step	43.6	71.8	69.0	59.2	9.9
Replace	*No context*	Step	44.4	77.1	72.5	63.4	8.0
Replace	*Unigram*	Step	48.9	76.8	74.0	63.2	10.0
Replace	tf-idf	*Episode*	42.6	62.3	68.9	52.7	6.8
Replace	tf-idf	Step	**50.0**	**77.6**	**74.6**	**65.6**	9.4

Table 5: Effect of using different reconciliation schemes, context-vectors, and rewards in our RL framework (*Shootings* domain). The last row is the overall best scheme (deviations from this are in *italics*). Context refers to the type of word counts used in the state vector to represent entity context. Rewards are either per step or per episode. (**S**: ShooterName, **K**: NumKilled, **W**: NumWounded, **C**: City, **Steps**: Average number of steps per episode)

place), we also experiment with using Confidence and Majority-based reconciliation schemes for RL-Extract. We observe that the Replace scheme performs much better than the others (2-6% on all entities) and believe this is because it provides the agent with more flexibility in choosing the final values.

From the same table, we see that using the tf-idf counts of context words as part of the state provides better performance than using no context or using simple unigram counts. In terms of reward structure, providing rewards after each step is empirically found to be significantly better (>10% on average) compared to a single delayed reward per episode. The last column shows the average number of steps per episode – the values range from 6.8 to 10.0 steps for the different schemes. The best system (RL-Extract with Replace, tf-idf and step-based rewards) uses 9.4 steps per episode.

7 Conclusions

In this paper, we explore the task of acquiring and incorporating external evidence to improve information extraction accuracy for domains with limited access to training data. This process comprises issuing search queries, extraction from new sources and reconciliation of extracted values, repeated until sufficient evidence is obtained. We use a reinforcement learning framework and learn optimal action sequences to maximize extraction accuracy while penalizing extra effort. We show that our model, trained as a deep Q-network, outperforms traditional extractors by 7.2% and 5% on average on two different domains, respectively. We also demonstrate the importance of sequential decision-making by comparing our model to a meta-classifier operating on the same space, obtaining up to a 7% gain.

Acknowledgements

We thank David Alvarez, Tao Lei and Ramya Ramakrishnan for helpful discussions and feedback, and the members of the MIT NLP group and the anonymous reviewers for their insightful comments. We also gratefully acknowledge support from a Google faculty award.

References

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

Michele Banko, Eric Brill, Susan Dumais, and Jimmy Lin. 2002. Askmsr: Question answering using the worldwide web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 7–9.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Hai Leong Chieu and Hwee Tou Ng. 2002. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of AAAI*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 109–112. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Trans-*

actions of the Association for Computational Linguistics, 2:477–490.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, Doha, Qatar, October. Association for Computational Linguistics.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal, September. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2015. Deep reinforcement learning with an action space defined by natural language. *arXiv preprint arXiv:1511.04636*.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1148–1158, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pallika H Kanani and Andrew K McCallum. 2012. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 253–262. ACM.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 489–500, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gideon S Mann and David Yarowsky. 2005. Multi-field information extraction and cross-document fusion. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 483–490. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China, July. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *Proceedings of Text Analysis Conference 2010 Workshop*.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. MIT Press.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th*

Annual Meeting of the Association for Computational Linguistics, pages 118–127. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Global Neural CCG Parsing with Optimality Guarantees

Kenton Lee **Mike Lewis** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{kentonl,mlewis,lsz}@cs.washington.edu

Abstract

We introduce the first global recursive neural parsing model with optimality guarantees during decoding. To support global features, we give up dynamic programs and instead search directly in the space of all possible subtrees. Although this space is exponentially large in the sentence length, we show it is possible to learn an efficient A* parser. We augment existing parsing models, which have informative bounds on the outside score, with a global model that has loose bounds but only needs to model non-local phenomena. The global model is trained with a novel objective that encourages the parser to search both efficiently and accurately. The approach is applied to CCG parsing, improving state-of-the-art accuracy by 0.4 F1. The parser finds the optimal parse for 99.9% of held-out sentences, exploring on average only 190 subtrees.

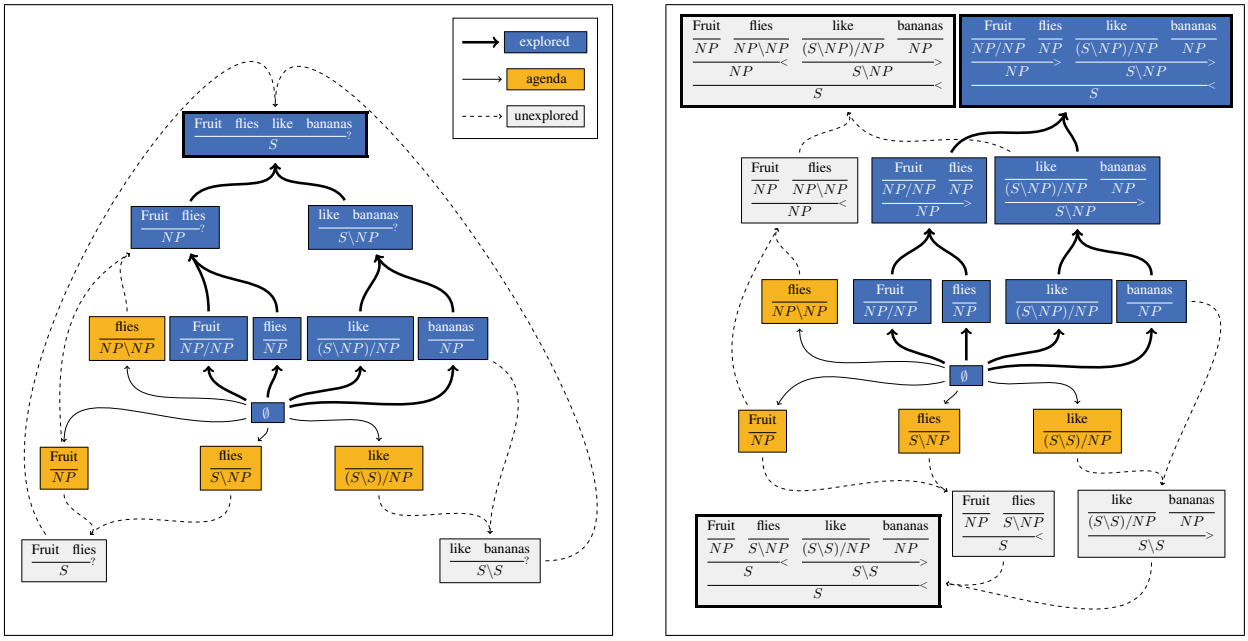
1 Introduction

Recursive neural models perform well for many structured prediction problems, in part due to their ability to learn representations that depend globally on all parts of the output structures. However, global models of this sort are incompatible with existing exact inference algorithms, since they do not decompose over substructures in a way that allows effective dynamic programming. Existing work has therefore used greedy inference techniques such as beam search (Vinyals et al., 2015; Dyer et al., 2015) or reranking (Socher et al., 2013). We introduce the first global recursive neural parsing approach with optimality guarantees for decoding and use it to build a state-of-the-art CCG parser.

To enable learning of global representations, we modify the parser to search directly in the space of all possible parse trees with no dynamic programming. Optimality guarantees come from A* search, which provides a certificate of optimality if run to completion with a heuristic that is a bound on the future cost. Generalizing A* to global models is challenging; these models also break the locality assumptions used to efficiently compute existing A* heuristics (Klein and Manning, 2003; Lewis and Steedman, 2014). Rather than directly replacing local models, we show that they can simply be augmented by adding a score from a global model that is constrained to be non-positive and has a trivial upper bound of zero. The global model, in effect, only needs to model the remaining non-local phenomena. In our experiments, we use a recent factored A* CCG parser (Lewis et al., 2016) for the local model, and we train a Tree-LSTM (Tai et al., 2015) to model global structure.

Finding a model that achieves these A* guarantees in practice is a challenging learning problem. Traditional structured prediction objectives focus on ensuring that the gold parse has the highest score (Collins, 2002; Huang et al., 2012). This condition is insufficient in our case, since it does not guarantee that the search will terminate in subexponential time. We instead introduce a new objective that optimizes efficiency as well as accuracy. Our loss function is defined over states of the A* search agenda, and it penalizes the model whenever the top agenda item is not a part of the gold parse.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2366–2376,
Austin, Texas, November 1-5, 2016.



(a) The search space in chart parsing, with one node for each labeling of a span.

(b) The search space in this work, with one node for each partial parse.

Figure 1: Illustrations of CCG parsing as hypergraph search, showing partial views of the search space. Weighted hyperedges from child nodes to a parent node represent rule productions scored by a parsing model. A path starting at ∅, for example the set of bolded hyperedges, represents the derivation of a parse. During decoding, we find the highest scoring path to a complete parse. Both figures show an ideal exploration that efficiently finds the optimal path. Figure 1a depicts the traditional search space, and Figure 1b depicts the search space in this work. Hyperedge scores can only depend on neighboring nodes, so our model can condition on the entire parse structure, at the price of an exponentially larger search space.

Minimizing this loss encourages the model to return the correct parse as quickly as possible.

The combination of global representations and optimal decoding enables our parser to achieve state-of-the-art accuracy for Combinatory Categorial Grammar (CCG) parsing. Despite being intractable in the worst case, the parser in practice is highly efficient. It finds optimal parses for 99.9% of held out sentences while exploring just 190 subtrees on average—allowing it to outperform beam search in both speed and accuracy.

2 Overview

Parsing as hypergraph search Many parsing algorithms can be viewed as a search problem, where parses are specified by paths through a hypergraph.

A node y in this hypergraph is a labeled span, representing structures within a parse tree, as shown in Figure 1. Each hyperedge e in the hypergraph represents a rule production in a parse. The head node of the hyperedge HEAD(e) is the parent of the rule production, and the tail nodes of the hyperedge are the children of the rule production. For example, consider the hyperedge in Figure 1b whose head is *like bananas*. This hyperedge represents a forward application rule applied to its tails, *like* and *bananas*.

To define a path in the hypergraph, we first include a special start node ∅ that represents an empty parse. ∅ has outgoing hyperedges that reach every leaf node, representing assignments of labels to words (supertag assignments in Figure 1). We then define a path to be a set of hyperedges E starting at ∅ and ending at a single destination node. A path therefore specifies the derivation of the parse constructed from the labeled spans at each node. For example, in Figure 1, the set of bolded hyperedges form a path deriving a complete parse.

Each hyperedge e is weighted by a score $s(e)$ from a parsing model. The score of a path E is the

sum of its hyperedge scores:

$$g(E) = \sum_{e \in E} s(e)$$

Viterbi decoding is equivalent to finding the highest scoring path that forms a complete parse.

Search on parse forests Traditionally, the hypergraph represents a packed parse chart. In this work, our hypergraph instead represents a *forest* of parses. Figure 1 contrasts the two representations.

In the parse chart, labels on the nodes represent local properties of a parse, such as the category of a span in Figure 1a. As a result, multiple parses that contain the same property include the same node in their path, (e.g. the node spanning the phrase *Fruit flies* with category NP). The number of nodes in this hypergraph is polynomial in the sentence length, permitting exhaustive exploration (e.g. CKY parsing). However, the model scores can only depend on local properties of a parse. We refer to these models as *locally factored* models.

In contrast, nodes in the parse forest are labeled with entire subtrees, as shown in Figure 1b. For example, there are two nodes spanning the phrase *Fruit flies* with the same category NP but different internal substructures. While the parse forest requires an exponential number of nodes in the hypergraph, the model scores can depend on entire subtrees.

A* parsing A* parsing has been successfully applied in locally factored models (Klein and Manning, 2003; Lewis and Steedman, 2014; Lewis et al., 2015; Lewis et al., 2016). We present a special case of A* parsing that is conceptually simpler, since the parse forest constrains each node to be reachable via a unique path. During exploration, we maintain the unique (and therefore highest scoring) path to a hyperedge e, which we define as $\text{PATH}(e)$.

Similar to the standard A* search algorithm, we maintain an agenda $\mathcal{A}$ of hyperedges to explore and a forest $\mathcal{F}$ of explored nodes that initially contains only the start node $\emptyset$.

Each hyperedge e in the agenda is sorted by the sum of its inside score $g(\text{PATH}(e))$ and an admissible heuristic $h(e)$. A heuristic $h(e)$ is admissible if it is an upper bound of the sum of hyperedge scores leading to any complete parse reachable from e (the Viterbi outside score). The efficiency of the search improves when this bound is tighter.

At every step, the parser removes the top of the agenda, $e_{max} = \text{argmax}_{e \in \mathcal{A}}(g(\text{PATH}(e)) + h(e))$. e_{max} is expanded by combining $\text{HEAD}(e_{max})$ with previously explored parses from $\mathcal{F}$ to form new hyperedges. These new hyperedges are inserted into $\mathcal{A}$, and $\text{HEAD}(e_{max})$ is added it to $\mathcal{F}$. We repeat these steps until the first complete parse y^* is explored. The bounds provided by $h(e)$ guarantee that the path to y^* has the highest possible score. Figure 1b shows an example of the agenda and the explored forest at the end of perfectly efficient search, where only the optimal path is explored.

Approach The enormous search space described above presents a challenge for an A* parser, since computing a tight and admissible heuristic is difficult when the model does not decompose locally.

Our key insight in addressing this challenge is that existing locally factored models with an informative A* heuristic can be augmented with a global score (Section 3). By constraining the global score to be non-positive, the A* heuristic from the locally factored model is still admissible.

While the heuristic from the local model offers some estimate of the future cost, the efficiency of the parser requires learning a well-calibrated global score, since the heuristic becomes looser as the global score provides stronger penalties (Section 5).

As we explore the search graph, we incrementally construct a neural network, which computes representations of the parses and allows backpropagation of errors from bad search steps (Section 4).

In the following sections, we present our approach in detail, assuming an existing locally factored model $s_{local}(e)$ for which we can efficiently compute an admissible A* heuristic $h(e)$.

In the experiments, we apply our model to CCG parsing, using the locally factored model and A* heuristic from Lewis et al. (2016).

3 Model

Our model scores a hyperedge e by combining the score from the local model with a global score that conditions on the entire parse at the head node:

$$s(e) = s_{local}(e) + s_{global}(e)$$

In $s_{global}(e)$, we first compute a hidden representation encoding the parse structure of $y = \text{HEAD}(e)$. We use a variant of the Tree-LSTM (Tai et al., 2015) connected to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) at the leaves. The combination of linear and tree LSTMs allows the hidden representation of partial parses to condition on both the partial structure and the full sentence. Figure 2 depicts the neural network that computes the hidden representation for a parse.

Formally, given a sentence $\langle w_1, w_2, \ldots, w_n \rangle$, we compute hidden states h_t and cell states c_t in the forward LSTM for $1 < t \leq n$:

$$\begin{aligned} i_t &= \sigma(W_i[c_{t-1}, h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o[\tilde{c}_t, h_{t-1}, x_t] + b_o) \\ \tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) \\ c_t &= i_t \circ \tilde{c}_t + (\mathbf{1} - i_t) \circ c_{t-1} \\ h_t &= o_t \circ \tanh(c_t) \end{aligned}$$

where σ is the logistic sigmoid, $\circ$ is the componentwise product, and x_t denotes a learned word embedding for w_t. We also construct a backward LSTM, which produces the analogous hidden and cell states starting at the end of the sentence, which we denote as c'_t and h'_t respectively. The start and end latent states, c_{-1}, h_{-1}, c'_{n+1}, and h'_{n+1}, are learned embeddings. This variant of the LSTM includes peephole connections and couples the input and forget gates.

The bidirectional LSTM over the words serves as a base case when we recursively compute a hidden representation for the parse y using the tree-structured generalization of the LSTM:

$$\begin{aligned} i_y &= \sigma(W_i^R[c_l, h_l, c_r, h_r, x_y] + b_i^R) \\ f_y &= \sigma(W_f^R[c_l, h_l, c_r, h_r, x_y] + b_f^R) \\ o_y &= \sigma(W_o^R[\widetilde{c}_y, h_l, h_r, x_y] + b_o^R) \\ c_{lr} &= f_y \circ c_l + (\mathbf{1} - f_y) \circ c_r \\ \widetilde{c}_y &= \tanh(W_c^R[h_l, h_r, x_y] + b_c^R) \\ c_y &= i_y \circ \widetilde{c}_y + (\mathbf{1} - i_y) \circ c_{lr} \\ h_y &= o_y \circ \tanh(c_y) \end{aligned}$$

where the weights and biases are parametrized by the rule R that produces y from its children, and x_y denotes a learned embedding for the category at the root of y. For example, in CCG, the rule would correspond to the CCG combinator, and the label would correspond to the CCG category.

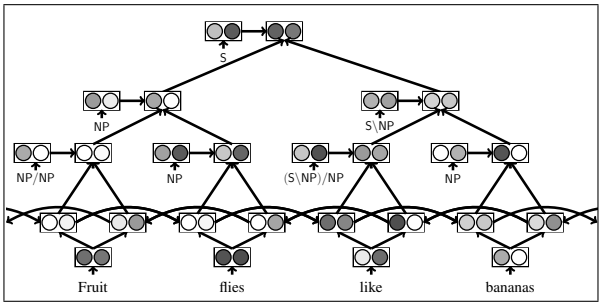


Figure 2: Visualization of the Tree-LSTM which computes vector embeddings for each parse node. The leaves of the Tree-LSTM are connected to a bidirectional LSTM over words, encoding lexical information within and outside of the parse.

We assume that nodes are binary, unary, or leaves. Their left and right latent states, c_l, h_l, c_r, and h_r are defined as follows:

- In a binary node, c_l and h_l are the cell and hidden states of the left child, and c_r and h_r are the cell and hidden states of the right child.
- In a unary node, c_l and h_l are learned embeddings, and c_r and h_r are the cell and hidden states of the singleton child.
- In a leaf node, let w denote the index of the corresponding word. Then c_l and h_l are c_w and h_w from the forward LSTM, and c_r and h_r are c'_w and h'_w from the backward LSTM.

The cell state of the recursive unit is a linear combination of the intermediate cell state $\widetilde{c}_y$, the left cell state c_l, and the right cell state c_r. To preserve the normalizing property of coupled gates, we perform coupling in a hierarchical manner: the input gate i_y decides the weights for $\widetilde{c}_y$, and the forget gate f_y shares the remaining weights between c_l and c_r.

Given the hidden representation h_y at the root, we score the global component as follows:

$$s_{global}(e) = \log(\sigma(W \cdot h_y))$$

This definition of the global score ensures that it is non-positive—an important property for inference.

4 Inference

Using the hyperedge scoring model $s(e)$ described in Section 3, we can find the highest scoring path that derives a complete parse tree by using the A* parsing algorithm described in Section 2.

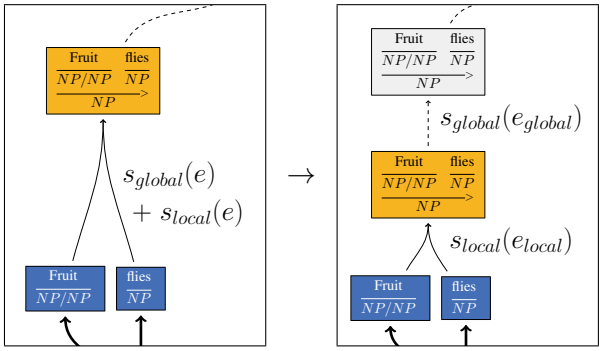


Figure 3: The hyperedge on the left requires computing both the local and global score when placed on the agenda. Splitting the hyperedge, as shown on the right, saves expensive computation of the global score if the local score alone indicates that the parse is not worth exploring.

Admissible A* heuristic Since our full model adds non-positive global scores to the existing local scores, path scores under the full model cannot be greater than path scores under the local model. Upper bounds for path scores under the local model also hold for path scores under the full model, and we simply reuse the A* heuristic from the local model to guide the full model during parsing without sacrificing optimality guarantees.

Incremental neural network construction The recursive hidden representations used in $s_{global}(e)$ can be computed in constant time during parsing. When scoring a new hyperedge, its children must have been previously scored. Instead of computing the full recursion, we reuse the existing latent states of the children and compute $s_{global}(e)$ with an incremental forward pass over a single recursive unit in the neural network. By maintain the latent states of each parse, we incrementally build a single DAG-structured LSTM mirroring the explored subset of the hypergraph. This not only enables quick forward passes during decoding, but also allows back-propagation through the search space after decoding, which is crucial for efficient learning (see Section 5).

Lazy global scoring The global score is expensive to compute. We introduce an optimization to avoid computing it when provably unnecessary. We split each hyperedge e into two successive hyperedges, e_{local} and e_{global}, as shown in Figure 3. The score for e, previously $s(e) = s_{local}(e) + s_{global}(e)$, is also split between the two new hyperedges:

$$s(e_{local}) = s_{local}(e_{local})$$
$$s(e_{global}) = s_{global}(e_{global})$$

Intuitively, this transformation requires A* to verify that the local score is good enough before computing the global score, which requires an incremental forward pass over a recursive unit in the neural network. In the example, this involves first summing the supertag scores of *Fruit* and *flies* and inserting the result back into the agenda. The score for applying the forward application rule to the recursive representations is only computed if that item appears again at the head of the agenda. In practice, the lazy global scoring reduces the number of recursive units by over 91%, providing a 2.4X speed up.

5 Learning

During training (Algorithm 1), we assume access to sentences labeled with gold parse trees $\hat{y}$ and gold derivations $\hat{E}$. The gold derivation $\hat{E}$ is a path from $\emptyset$ to $\hat{y}$ in the parse forest.

A* search with our global model is not guaranteed to terminate in sub-exponential time. This creates challenges for learning—for example, it is not possible in practice to use the standard structured perceptron update (Collins, 2002), because the search procedure rarely terminates early in training. Other common loss functions assume inexact search (Huang et al., 2012), and do not optimize efficiency.

Instead, we optimize a new objective that is tightly coupled with the search procedure. During parsing, we would like hyperedges from the gold derivation to appear at the top of the agenda $\mathcal{A}$. When this condition does not hold, A* is searching inefficiently, and we refer to this as a *violation* of the agenda, which we formally define as:

$$v(\hat{E}, \mathcal{A}) = \max_{e \in \mathcal{A}} (g(\text{PATH}(e)) + h(e)) - \max_{e \in \mathcal{A} \cap \hat{E}} (g(\text{PATH}(e)) + h(e))$$

where $g(\text{PATH}(e))$ is the score of the unique path to e, and $h(e)$ is the A* heuristic. If all violations are zero, we find the gold parse without exploring any incorrect partial parses—maximizing both accuracy and efficiency. Figure 1b shows such a case—if any other nodes were explored, they would be violations.

Update	Loss($\mathcal{V}$)
Greedy	$\mathcal{V}_1$
Max violation	$\max_{t=1}^{T} \mathcal{V}_t$
All violations	$\sum_{t=1}^{T} \mathcal{V}_t$

Table 1: Loss functions optimized by the different update methods. The updates depend on the list of T non-zero violations, $\mathcal{V} = \langle \mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_T \rangle$, as defined in Section 5.

In existing work on violation-based updates, comparisons are only made between derivations with the same number of steps (Huang et al., 2012; Clark et al., 2015)—whereas our definition allows subtrees of arbitrary spans to compete with each other, because hyperedges are not explored in a fixed order. Our violations also differ from Huang et al.'s in that we optimize efficiency as well as accuracy.

We define loss functions over these violations, which are minimized to encourage correct and efficient search. During training, we parse each sentence until either the gold parse is found or we reach computation limits. We record $\mathcal{V}$, the list of non-zero violations of the agenda $\mathcal{A}$ observed:

$$\mathcal{V} = \langle v(\hat{E}, \mathcal{A}) \mid v(\hat{E}, \mathcal{A}) > 0 \rangle$$

We can optimize several loss functions over $\mathcal{V}$, as defined in Table 1. The greedy and max-violation updates are roughly analogous to the violation-fixing updates proposed by Huang et al. (2012), but adapted to exact agenda-based parsing. We also introduce a new *all-violations* update, which minimizes the sum of all observed violations. The all-violations update encourages correct parses to be explored early (similar to the greedy update) while being robust to parses with multiple deviations from the gold parse (similar to the max-violation update).

The violation losses are optimized with subgradient descent and backpropagation. For our experiments, $s_{local}(e)$ and $h(e)$ are kept constant. Only the parameters θ of $s_{global}(e)$ are updated. Therefore, a subgradient of a violation $v(\hat{E}, \mathcal{A})$ can be computed by summing subgradients of the global scores.

$$\frac{\partial v(\hat{E}, \mathcal{A})}{\partial \theta} = \sum_{e \in \text{PATH}(e_{max})} \frac{\partial s_{global}(e)}{\partial \theta} - \sum_{e \in \text{PATH}(\hat{e}_{max})} \frac{\partial s_{global}(e)}{\partial \theta}$$

where e_{max} denotes the hyperedge at the top of the agenda $\mathcal{A}$ and $\hat{e}_{max}$ denotes the hyperedge in the gold derivation $\hat{E}$ that is closest to the top of $\mathcal{A}$.

Algorithm 1 Violation-based learning algorithm

Definitions D is the training data containing input sentences x and gold derivations $\hat{E}$. e variables denote scored hyperedges. TAG(x) returns a set of scored pre-terminals for every word. ADD($\mathcal{F}$, y) adds partial parse y to forest $\mathcal{F}$. RULES($\mathcal{F}$, y) returns the set of scored hyperedges that can be created by combining y with entries in F. SIZE_OK($\mathcal{F}$, $\mathcal{A}$) returns whether the sizes of the forest and agenda are within predefined limits.

```
function VIOLATIONS(Ê, x, θ)
    V ← ∅                                  ▷ Initialize list of violations V
    F ← ∅                                  ▷ Initialize forest F
    A ← ∅                                  ▷ Initialize agenda A
    for e ∈ TAG(x) do
        PUSH(A, e)
    while |A ∩ Ê| > 0 and SIZE_OK(F, A) do
        if v(Ê, A) > 0 then
            APPEND(V, v(Ê, A))             ▷ Record violation
        e_max ← EXTRACT_MAX(A)             ▷ Pop agenda
        ADD(F, HEAD(e_max))                ▷ Explore hyperedge
        for e ∈ RULES(F, HEAD(e_max), θ) do
            PUSH(A, e)                     ▷ Expand hyperedge
    return V

function LEARN(D)
    for i = 1 to T do
        for x, Ê ∈ D do
            V ← VIOLATIONS(Ê, x, θ)
            L ← LOSS(V)
            θ ← OPTIMIZE(L, θ)
    return θ
```

6 Experiments

6.1 Data

We trained our parser on Sections 02-21 of CCGbank (Hockenmaier and Steedman, 2007), using Section 00 for development and Section 23 for test. To recover a single gold derivation for each sentence to use during training, we find the right-most branching parse that satisfies the gold dependencies.

6.2 Experimental Setup

For the local model, we use the *supertag-factored* model of Lewis et al. (2016). Here, $s_{local}(e)$ corresponds to a supertag score if a HEAD(e) is a leaf and zero otherwise. The outside score heuristic is computed by summing the maximum supertag score for every word outside of each span. In the reported results, we back off to the supertag-factored model after the forest size exceeds 500,000, the agenda size exceeds 2 million, or we build more than 200,000 recursive units in the neural network.

Model	Dev F1	Test F1
C & C	83.8	85.2
C & C + RNN	86.3	87.0
Xu (2016)	87.5	87.8
Vaswani et al. (2016)	87.8	88.3
Supertag-factored	87.5	88.1
Global A*	**88.4**	**88.7**

Table 2: Labeled F1 for CCGbank dependencies on the CCGbank development and test set for our system **Global A*** and the baselines.

Model	Dev F1	Optimal	Explored
Supertag-factored	87.5	100.0%	402.5
− dynamic program	87.5	97.1%	17119.6
Span-factored	87.9	99.9%	176.5
− dynamic program	87.8	99.5%	578.5
Global A*	**88.4**	99.8%	309.6
− lexical inputs	87.8	99.6%	538.5
− lexical context	88.1	99.4%	610.5

Table 3: Ablations of our full model (**Global A***) on the development set. *Explored* refers to the size of the parse forest. Results show the importance of global features and lexical information in context.

Our full system is trained with all-violations updates. During training, we lower the forest size limit to 2000 to reduce training times. The model is trained for 30 epochs using ADAM (Kingma and Ba, 2014), and we use early stopping based on development F1. The LSTM cells and hidden states have 64 dimensions. We initialize word representations with pre-trained 50-dimensional embeddings from Turian et al. (2010). Embeddings for categories have 16 dimensions and are randomly initialized. We also apply dropout with a probability of 0.4 at the word embedding layer during training. Since the structure of the neural network is dynamically determined, we do not use mini-batches. The neural networks are implemented using the CNN library,[1] and the CCG parser is implemented using the EasySRL library.[2] The code is available online.[3]

6.3 Baselines

We compare our parser to several baseline CCG parsers: the C&C parser (Clark and Curran, 2007); C&C + RNN (Xu et al., 2015), which is the C&C parser with an RNN supertagger; Xu (2016), a LSTM shift-reduce parser; Vaswani et al. (2016) who combine a bidirectional LSTM supertagger with a beam search parser using global features (Clark et al., 2015); and *supertag-factored* (Lewis et al., 2016), which uses deterministic A* decoding and an LSTM supertagging model.

6.4 Parsing Results

Table 2 shows parsing results on the test set. Our global features let us improve over the supertag-factored model by 0.6 F1. Vaswani et al. (2016) also use global features, but our optimal decoding leads to an improvement of 0.4 F1.

Although we observed an overall improvement in parsing performance, the supertag accuracy was not significantly different after applying the parser.

On the test data, the parser finds the optimal parse for 99.9% sentences before reaching our computational limits. On average, we parse 27.1 sentences per second,[4] while exploring only 190.2 subtrees.

6.5 Model Ablations

We ablate various parts of the model to determine how they contribute to the accuracy and efficiency of the parser, as shown in Table 3. For each model, the comparisons include the average number of parses explored and the percentage of sentences for which an optimal parse can be found without backing off.

Structure ablation We first ablate the global score, $s_{global}(y)$, from our model, thus relying entirely on the local supertag-factors that do not explicitly model the parse structure. This ablation allows dynamic programming and is equivalent to the backoff model (*supertag-factored* in Table 3). Surprisingly, even in the exponentially larger search space, the global model explores *fewer* nodes than the supertag-factored model—showing that the global model efficiently prune large parts of the search space. This effect is even larger when not using dynamic programming in the supertag-factored model.

Global structure ablation To examine the importance of global features, we ablate the recursive hidden representation (*span-factored* in Table 3). The model in this ablation decomposes over labels for

[1] https://github.com/clab/cnn

[2] https://github.com/mikelewis0/EasySRL

[3] https://github.com/kentonl/neuralccg

[4] We use a single 3.5GHz CPU core.

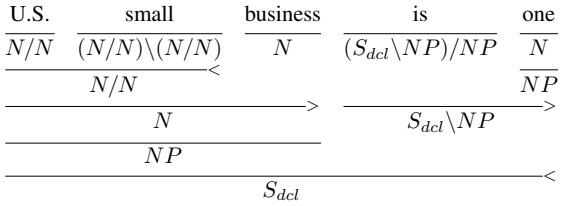


Figure 4: Example of an incorrect partial parse that appears syntactically plausible in isolation. The full sentence is *'Indeed, for many Japanese trading companies, the favorite **U.S. small business is one** whose research and development can be milked for future Japanese use.'* The global model heavily penalizes this garden path, thereby avoiding regions that lead to dead ends and allowing the global model to explore fewer nodes.

spans, as in Durrett and Klein (2015). In this model, the recursive unit uses, instead of latent states from its children, the latent states of the backward LSTM at the start of the span and the latent states of the forward LSTM at the end of the span. Therefore, this model encodes the lexical information available in the full model but does not encode the parse structure beyond the local rule production. While the dynamic program allows this model to find the optimal parse with fewer explorations, the lack of global features significantly hurts its parsing accuracy.

Lexical ablation We also show lexical ablations instead of structural ablations. We remove the bidirectional LSTM at the leaves, thus delexicalizing the global model. This ablation degrades both accuracy and efficiency, showing that the model uses lexical information to discriminate between parses.

To understand the importance of contextual information, we also perform a partial lexical ablation by using word embeddings at the leaves instead of the bidirectional LSTM, thus propagating only lexical information from within the span of each parse. The degradation in F1 is about half of the degradation from the full lexical ablation, suggesting that a significant portion of the lexical cues comes from the context of a parse. Figure 4 illustrates the importance of context with an incorrect partial parse that appears syntactically plausible in isolation. These bottom-up garden paths are typically problematic for parsers, since their incompatibility with the remaining sentence is difficult to recognize until later stages of decoding. However, our global model learns to heavily penalize these garden paths by using the context provided by the bidirectional LSTM and avoid paths that lead to dead ends or bad regions of the search space.

Update	Dev F1	Optimal	Explored
Greedy	87.9	99.2%	2313.8
Max-violation	88.1	99.9%	217.3
All-violations	**88.4**	99.8%	309.6

Table 4: Parsing results trained with different update methods. Our system uses **all-violations** updates and is the most accurate.

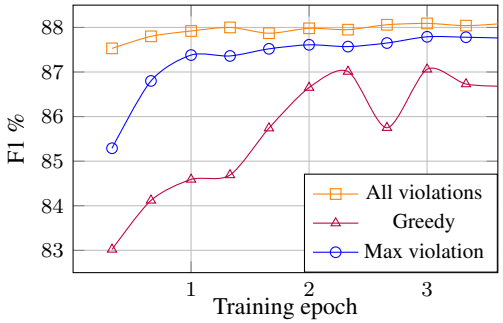


Figure 5: Learning curves for the first 3 training epochs on the development set when training with different updates strategies. The all-violations update shows the fastest convergence.

6.6 Update Comparisons

Table 4 compares the different violation-based learning objectives, as discussed in Section 5. Our novel *all-violation* updates outperform the alternatives. We attribute this improvement to the robustness over poor search spaces, which the greedy update lacks, and the incentive to explore good parses early, which the max-violation update lacks. Learning curves in Figure 5 show that the all-violations update also converges more quickly.

6.7 Decoder Comparisons

Lastly, to show that our parser is both more accurate and efficient than other decoding methods, we decode our full model using best-first search, reranking, and beam search. Table 5 shows the F1 scores with and without the backoff model, the portion of the sentences that each decoder is able to parse, and the time spent decoding relative to the A* parser.

In the best-first search comparison, we do not include the informative A* heuristic, and the parser completes very few parses before reaching computational limits—showing the importance of heuristics in large search spaces. In the reranking comparison,

Decoder	Dev F1	Dev F1 − backoff	Relative Time
Global A*	**88.4**	88.4 (99.8%)	1X
Best-first	87.5	2.8 (6.7%)	293.4X
10-best reranking	87.9	87.9 (99.7%)	8.5X
100-best reranking	88.2	88.0 (99.4%)	72.3X
2-best beam search	88.2	85.7 (94.0%)	2.0X
4-best beam search	88.3	88.1 (99.2%)	6.7X
8-best beam search	88.2	86.8 (98.1%)	26.3X

Table 5: Comparison of various decoders using the same model from our full system (**Global A***). We report F1 with and without the backoff model, the percentage of sentences that the decoder can parse, and the time spent decoding relative to A*.

we obtain n-best lists from the backoff model and rerank each result with the full model. In the beam search comparison, we use the approach from Clark et al. (2015) which greedily finds the top-n parses for each span in a bottom-up manner. Results indicate that both approximate methods are less accurate and slower than A*.

7 Related Work

Many structured prediction problems are based around dynamic programs, which are incompatible with recursive neural networks because of their real-valued latent variables. Some recent models have neural factors (Durrett and Klein, 2015), but these cannot condition on global parse structure, making them less expressive. Our search explores fewer nodes than dynamic programs, despite an exponentially larger search space, by allowing the recursive neural network to guide the search.

Previous work on structured prediction with recursive or recurrent neural models has used beam search–e.g. in shift reduce parsing (Dyer et al., 2015), string-to-tree transduction (Vinyals et al., 2015), or reranking (Socher et al., 2013)–at the cost of potentially recovering suboptimal solutions. For our model, beam search is both less efficient and less accurate than optimal A* decoding. In the non-neural setting, Zhang et al. (2014) showed that global features with greedy inference can improve dependency parsing. The CCG beam search parser of Clark et al. (2015), most related to this work, also uses global features. By using neural representations and exact search, we improve over their results.

A* parsing has been previously proposed for locally factored models (Klein and Manning, 2003; Pauls and Klein, 2009; Auli and Lopez, 2011; Lewis and Steedman, 2014). We generalize these methods to enable global features. Vaswani and Sagae (2016) apply best-first search to an unlabeled shift-reduce parser. Their use of error states is related to our global model that penalizes local scores. We demonstrated that best-first search is infeasible in our setting, due to the larger search space.

A close integration of learning and decoding has been shown to be beneficial for structured prediction. SEARN (Daumé III et al., 2009) and DAGGER (Ross et al., 2011) learn greedy policies to predict structure by sampling classification examples over actions from single states. We similarly generate classification examples over hyperedges in the agenda, but actions from multiple states compete against each other. Other learning objectives that update parameters based on a beam or agenda of partial structures have also been proposed (Collins and Roark, 2004; Daumé III and Marcu, 2005; Huang et al., 2012; Andor et al., 2016; Wiseman and Rush, 2016), but the impact of search errors is unclear.

8 Conclusion

We have shown for the first time that a parsing model with global features can be decoded with optimality guarantees. This enables the use of powerful recursive neural networks for parsing without resorting to approximate decoding methods. Experiments show that this approach is effective for CCG parsing, resulting in a new state-of-the-art parser. In future work, we will apply our approach to other structured prediction tasks, where neural networks—and greedy beam search—have become ubiquitous.

Acknowledgements

We thank Luheng He, Julian Michael, and Mark Yatskar for valuable discussion, and the anonymous reviewers for feedback and comments.

This work was supported by the NSF (IIS-1252835, IIS-1562364), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452.

Michael Auli and Adam Lopez. 2011. Efficient CCG parsing: A* versus Adaptive Supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Stephen Clark and James R Curran. 2007. Wide-coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4).

Stephen Clark, Darren Foong, Luana Bulat, and Wenduan Xu. 2015. The Java Version of the C&C Parser: Version 0.95. Technical report, University of Cambridge Computer Laboratory, August.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate Large Margin Methods for Structured Prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Greg Durrett and Dan Klein. 2015. Neural CRF Parsing. In *Proceedings of the Association for Computational Linguistics*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based Dependency Parsing with Stack Long Short-Term Memory. In *Proc. ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a Corpus of CCG derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3).

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Dan Klein and Christopher D Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labelling. In *Empirical Methods in Natural Language Processing*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Adam Pauls and Dan Klein. 2009. K-best A* Parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 958–966. Association for Computational Linguistics.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 627–635.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with Compositional Vector Grammars. In *Proceedings of the ACL conference*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations from Tree-structured Long Short-term Memory Networks. *arXiv preprint arXiv:1503.00075*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Ashish Vaswani and Kenji Sagae. 2016. Efficient Structured Inference for Transition-Based Parsing with

Neural Networks and Error States. *Transactions of the Association for Computational Linguistics*.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging With LSTMs. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of EMNLP*.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG Supertagging with a Recurrent Neural Network. *Volume 2: Short Papers*, page 250.

Wenduan Xu. 2016. LSTM Shift-Reduce CCG Parsing . In *Empirical Methods in Natural Language Processing*.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is Good if Randomized: New Inference for Dependency Parsing. In *Proceedings of EMNLP*.

Learning a Lexicon and Translation Model from Phoneme Lattices

Oliver Adams,[♠♡] **Graham Neubig,**[♣♡] **Trevor Cohn,**[♠]
Steven Bird,[♠] **Quoc Truong Do,**[♡] **Satoshi Nakamura**[♡]
[♠]The University of Melbourne, Australia
[♣]Carnegie Mellon University, Pittsburgh, PA, USA
[♡]Nara Institute of Science and Technology, Japan

Abstract

Language documentation begins by gathering speech. Manual or automatic transcription at the word level is typically not possible because of the absence of an orthography or prior lexicon, and though manual phonemic transcription is possible, it is prohibitively slow. On the other hand, translations of the minority language into a major language are more easily acquired. We propose a method to harness such translations to improve automatic phoneme recognition. The method assumes no prior lexicon or translation model, instead learning them from phoneme lattices and translations of the speech being transcribed. Experiments demonstrate phoneme error rate improvements against two baselines and the model's ability to learn useful bilingual lexical entries.

1 Introduction

Most of the world's languages are dying out and have little recorded data or linguistic documentation (Austin and Sallabank, 2011). It is important to adequately document languages while they are alive so that they may be investigated in the future. Language documentation traditionally involves one-on-one elicitation of speech from native speakers in order to produce lexicons and grammars that describe the language. However, this does not scale: linguists must first transcribe the speech phonemically as most of these languages have no standardized orthography. This is a critical bottleneck since it takes a trained linguist about 1 hour to transcribe the phonemes of 1 minute of speech (Do et al., 2014).

Smartphone apps for rapid collection of bilingual data have been increasingly investigated (De Vries et al., 2011; De Vries et al., 2014; Reiman, 2010; Bird et al., 2014; Blachon et al., 2016). It is common for these apps to collect speech segments paired with spoken translations in another language, making spoken translations quicker to obtain than phonemic transcriptions.

We present a method to improve automatic phoneme transcription by harnessing such bilingual data to learn a lexicon and translation model directly from source phoneme lattices and their written target translations, assuming that the target side is a major language that can be efficiently transcribed.[1] A Bayesian non-parametric model expressed with a weighted finite-state transducer (WFST) framework represents the joint distribution of source acoustic features, phonemes and latent source words given the target words. Sampling of alignments is used to learn source words and their target translations, which are then used to improve transcription of the source audio they were learnt from. Importantly, the model assumes no prior lexicon or translation model.

This method builds on work on phoneme translation modeling (Besacier et al., 2006; Stüker et al., 2009; Stahlberg et al., 2012; Stahlberg et al., 2014; Adams et al., 2015; Duong et al., 2016), speech translation (Casacuberta et al., 2004; Matusov et al., 2005), computer-aided translation, (Brown et al., 1994; Vidal et al., 2006; Khadivi and Ney, 2008; Reddy and Rose, 2010; Pelemans et al., 2015), translation modeling from automatically transcribed

[1]Code is available at https://github.com/oadams/latticetm.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2377–2382,
Austin, Texas, November 1-5, 2016.

speech (Paulik and Waibel, 2013), word segmentation and translation modeling (Chang et al., 2008; Dyer, 2009; Nguyen et al., 2010; Chen and Xu, 2015), Bayesian word alignment (Mermer et al., 2013; Zezhong et al., 2013) and language model learning from lattices (Neubig et al., 2012). While we previously explored learning a translation model from word lattices (Adams et al., 2016), in this paper we extend the model to perform unsupervised word segmentation over phoneme lattices in order to improve phoneme recognition.

Experiments demonstrate that our method significantly reduces the phoneme error rate (PER) of transcriptions compared with a baseline recogniser and a similar model that harnesses only monolingual information, by up to 17% and 5% respectively. We also find that the model learns meaningful bilingual lexical items.

2 Model description

Our model extends the standard automatic speech recognition (ASR) problem by seeking the best phoneme transcription $\hat{\boldsymbol{\phi}}$ of an utterance in a joint probability distribution that incorporates acoustic features $\boldsymbol{x}$, phonemes $\boldsymbol{\phi}$, latent source words $\boldsymbol{f}$ and observed target transcriptions $\boldsymbol{e}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi},\boldsymbol{f}}{\operatorname{argmax}} P(\boldsymbol{x}|\boldsymbol{\phi})P(\boldsymbol{\phi}|\boldsymbol{f})P(\boldsymbol{f}|\boldsymbol{e})\,, \qquad (1)$$

assuming a Markov chain of conditional independence relationships (bold symbols denote utterances as opposed to tokens). Deviating from standard ASR, we replace language model probabilities with those of a translation model, and search for phonemes instead of words. Also, no lexicon or translation model are given in training.

2.1 Expression of the distribution using finite-state transducers

We use a WFST framework to express the factors of (1) since it offers computational tractability and simple inference in a clear, modular framework. Figure 1 uses a toy German–English error resolution example to illustrate the components of the framework: a phoneme lattice representing phoneme uncertainty according to $P(\boldsymbol{x}|\boldsymbol{\phi})$; a lexicon that transduces phoneme substrings $\boldsymbol{\phi}_s$ of $\boldsymbol{\phi}$ to source tokens f according to $P(\boldsymbol{\phi}_s|f)$; and a lexical translation model representing $P(f|e)$ for each e in the written translation. The composition of these components is also shown at the bottom of Figure 1, illustrating how would-be transcription errors can be resolved. This framework is reminiscent of the WFST framework used by Neubig et al. (2012) for lexicon and language model learning from monolingual data.

2.2 Learning the lexicon and translation model

Because we do not have knowledge of the source language, we must learn the lexicon and translation model from the phoneme lattices and their written translation. We model lexical translation probabilities using a Dirichlet process. Let A be both the transcription of each source utterance $\mathbf{f}$ and its word alignments to the translation $\mathbf{e}$ that generated them. The conditional posterior can be expressed as:

$$P(f|e;A) = \frac{c_A(f,e) + \alpha P_0(f)}{c_A(e) + \alpha}\,, \qquad (2)$$

where $c_A(f,e)$ is a count of how many times f has aligned to e in A and $c_A(e)$ is a count of how many times e has been aligned to; P_0 is a base distribution that influences how phonemes are clustered; and α determines the emphasis on the base distribution.

In order to express the Dirichlet process using the WFST components, we take the union of the lexicon with a *spelling model* base distribution that consumes phonemes $\phi_i \ldots \phi_j$ and produces a special $\langle\text{unk}\rangle$ token with probability $P_0(\phi_i \ldots \phi_j)$. This $\langle\text{unk}\rangle$ token is consumed by a designated arc in the translation model WFST with probability $\frac{\alpha}{c_A(e)+\alpha}$, yielding a composed probability of $\frac{\alpha P_0(f)}{c_A(e)+\alpha}$. Other arcs in the translation model express the probability $\frac{c_A(f,e)}{c_A(e)+\alpha}$ of entries already in the lexicon. The sum of these two probabilities equates to (2).

As for the spelling model P_0, we consider three distributions and implement WFSTs to represent them: a geometric distribution, *Geometric(γ)*, a Poisson distribution, *Poisson(λ)*,[2] and a 'shifted' geometric distribution, *Shifted(α, γ)*. The shifted geometric distribution mitigates a shortcoming of the geometric distribution whereby words of length 1 have the highest probability. It does so by having

[2]While the geometric distribution can be expressed recursively, we cap the number of states in the Poisson WFST to 100.

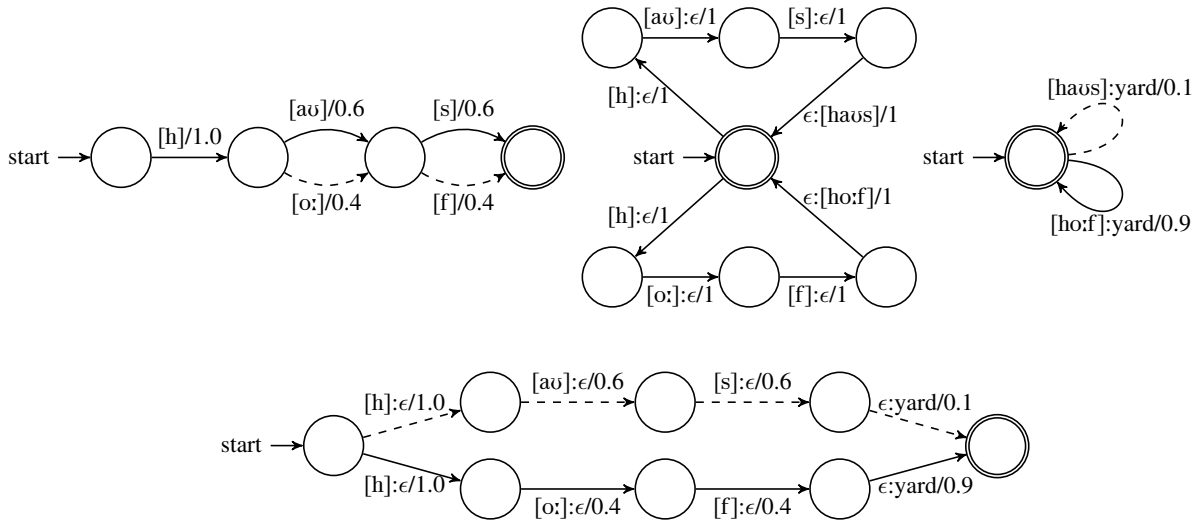


Figure 1: Top left to right: the phoneme lattice, the lexicon, and the translation model. Bottom: the resulting composed WFST. Given an English translation *'yard'*, the most likely transcription is corrected to [hoːf] (*'Hof'*) in the composed WFST, while in the original phoneme lattice it is [haʊs] (*'Haus'*). Solid edges represent most likely paths.

another parameter α that specifies the probability of a word of length 1, with the remaining probability mass distributed geometrically. All phonemes types are treated the same in these distributions, with uniform probability.

2.3 Inference

In order to determine the translation model parameters as described above, we require the alignments A. We sample these proportionally to their probability given the data and our prior, in effect integrating over all parameter configurations T:

$$P(A|\mathcal{X};\alpha,P_0) = \int_T P(A|\mathcal{X},T)P(T;\alpha,P_0)dT\,, \tag{3}$$

where $\mathcal{X}$ is our dataset of source phoneme lattices paired with target sentences.

This is achieved using blocked Gibbs sampling, with each utterance constituting one block. To sample from WFSTs, we use *forward-filtering/backward-sampling* (Scott, 2002; Neubig et al., 2012), creating forward probabilities using the forward algorithm for hidden Markov models before *backward-sampling* edges proportionally to the product of the forward probability and the edge weight.[3]

[3]No Metropolis-Hastings rejection step was used.

3 Experimental evaluation

We evaluate the lexicon and translation model by their ability to improve phoneme recognition, measuring phoneme error rate (PER).

3.1 Experimental setup

We used less than 10 hours of English–Japanese data from the BTEC corpus (Takezawa et al., 2002), comprised of spoken utterances paired with textual translations. This allows us to assess the approach assuming quality acoustic models. We used acoustic models similar to Heck et al. (2015) to obtain source phoneme lattices. Gold phoneme transcriptions were obtained by transforming the text with pronunciation lexicons and, in the Japanese case, first segmenting the text into tokens using KyTea (Neubig et al., 2011).

We run experiments in both directions: English–Japanese and Japanese–English (*en–ja* and *ja–en*), while comparing against three settings: the ASR 1-best path uninformed by the model (*ASR*); a monolingual version of our model that is identical except without conditioning on the target side (*Mono*); and the model applied using the source language sentence as the target (*Oracle*).

We tuned on the first 1,000 utterences (about 1 hour) of speech and trained on up to 9 hours of the

	English (en)			Japanese (ja)		
	Mono	–ja	Oracle	Mono	–en	Oracle
ASR		22.1			24.3	
Vague	17.7	18.5	17.2	21.5	20.8	21.6
Shifted	17.4	16.9	16.6	21.2	20.1	20.2
Poisson	17.3	17.2	16.8	21.3	20.1	20.8

Table 1: Phoneme error rates (percent) when training on 9 hours of speech, averaged over 4 runs.

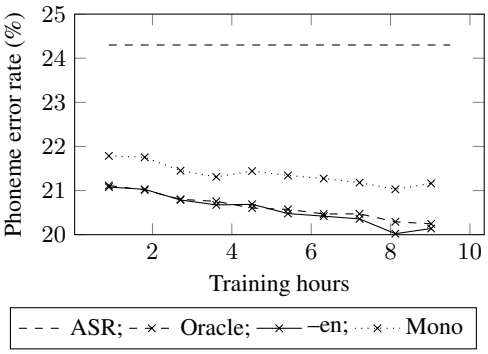


Figure 2: Japanese phoneme error rates using the *shifted* geometric prior when training data is scaled up from 1–9 hours, averaged over 3 runs.

remaining data.[4] Only the oracle setup was used for tuning, with Geometric(0.01) (taking the form of a *vague* prior), Shifted(10^{-5}, 0.25) and Poisson(7) performing best.

3.2 Results and Discussion

Table 1 shows en–ja and ja–en results for all methods with the full training data. Figure 2 shows improvements of ja–en over both the ASR baseline and the Mono method as the training data increases, with translation modeling gaining an increasing advantage with more training data.

Notably, English recognition gains less from using Japanese as the target side (en–ja) than the other way around, while the 'oracle' approach for Japanese recognition, which also uses Japanese as the target, underperforms ja–en. These observations suggest that using the Japanese target is less helpful, likely explained by the fine-grained morphological segmentation we used, making it harder for the model to relate source phonemes to target tokens.

The vague geometric prior significantly underperforms the other priors. In the en–ja/vague case, the model actually underperforms its monolingual counterpart. The vague prior biases slightly towards fine-grained English source segmentation, with words of length 1 most common. In this case, fine-grained Japanese is also used as the target which results in most lexical entries arising from uninformative alignments between single English phonemes and Japanese syllables, such as *[t]*⇔す. For similar reasons, the shifted geometric prior gains an advantage over Poisson, likely because of its ability to even further penalize single-phoneme lexical items, which regularly end up in all lexicons anyway due to their combinatorical advantage when sampling.

While many bilingual lexical entries are correct, such as *[wʌn]*⇔一 (*'one'*), most are not. Some have segmentation errors *[liːz]*⇔ください (*'please'*); some are correctly segmented but misaligned to commonly co-occurring words *[wʌt]*⇔時 (*'what'* aligned to *'time'*); others do not constitute individual words, but morphemes aligned to common Japanese syllables *[iːŋ]*⇔く (*'-ing'*); others still align multi-word units correctly *[haʊmʌtʃ]*⇔いくら (*'how much'*). Note though that entries such as those listed above capture information that may nevertheless help to reduce phoneme transcription errors.

4 Conclusion and Future Work

We have demonstrated that a translation model and lexicon can be learnt directly from phoneme lattices in order to improve phoneme transcription of those very lattices.

One of the appealing aspects of this modular framework is that there is much room for extension and improvement. For example, by using adaptor grammars to encourage syllable segmentation (Johnson, 2008), or incorporating language model probabilities in addition to our translation model probabilities (Neubig et al., 2012).

We assume a good acoustic model with phoneme error rates between 20 and 25%. In a language documentation scenario, acoustic models for the low-resource source language won't exist. Future work should use a universal phoneme recognizer or acoustic model of a similar language, thus making a step towards true generalizability.

[4] A 1 hour subset was used for PER evaluation.

Acknowledgments

We gratefully acknowledge support from the DARPA LORELEI program.

References

Oliver Adams, Graham Neubig, Trevor Cohn, and Steven Bird. 2015. Inducing bilingual lexicons from small quantities of sentence-aligned phonemic transcriptions. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2015)*, Da Nang, Vietnam.

Oliver Adams, Graham Neubig, Trevor Cohn, and Steven Bird. 2016. Learning a translation model from word lattices. In *17th Annual Conference of the International Speech Communication Association (INTERSPEECH 2016)*, San Francisco, California, USA.

Peter Austin and Julia Sallabank. 2011. *The Cambridge Handbook of Endangered Languages*. Cambridge Handbooks in Language and Linguistics. Cambridge University Press.

Laurent Besacier, Bowen Zhou, and Yuqing Gao. 2006. Towards speech translation of non written languages. In *2006 IEEE Spoken Language Technology Workshop (SLT 2006)*, pages 222–225, Palm Beach, Aruba.

Steven Bird, Florian R Hanke, Oliver Adams, and Haejoong Lee. 2014. Aikuma: A mobile app for collaborative language documentation. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 1–5, Baltimore, Maryland, USA.

David Blachon, Elodie Gauthier, Laurent Besacier, Guy-Noël Kouarata, Martine Adda-Decker, and Annie Rialland. 2016. Parallel speech collection for under-resourced language studies using the lig-aikuma mobile device app. *Procedia Computer Science*, 81:61–66.

Peter F Brown, Stanley F Chen, Stephen A Della Pietra, Vincent J Della Pietra, Andrew S Kehler, and Robert L Mercer. 1994. Automatic speech recognition in machine-aided translation. *Computer Speech & Language*, 8(3):177–187.

Francisco Casacuberta, Hermann Ney, Franz Josef Och, Enrique Vidal, Juan Miguel Vilar, Sergio Barrachina, Ismael García-Varea, David Llorens, César Martínez, Sirko Molau, and Others. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech & Language*, 18(1):25–47.

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT 2008)*, pages 224–232, Columbus, Ohio, USA.

Wei Chen and Bo Xu. 2015. Semi-supervised Chinese word segmentation based on bilingual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1207–1216, Lisbon, Portugal.

Nic J De Vries, Jaco Badenhorst, Marelie H Davel, Etienne Barnard, and Alta De Waal. 2011. Woefzela - an open-source platform for ASR data collection in the developing world. In *12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pages 3177–3180, Florence, Italy.

Nic J De Vries, Marelie H Davel, Jaco Badenhorst, Willem D Basson, Febe De Wet, Etienne Barnard, and Alta De Waal. 2014. A smartphone-based ASR data collection tool for under-resourced languages. *Speech Communication*, 56:119–131.

Thi-Ngoc-Diep Do, Alexis Michaud, and Eric Castelli. 2014. Towards the automatic processing of Yongning Na (Sino-Tibetan): developing a 'light' acoustic model of the target language and testing 'heavyweight' models from five national languages. In *4th International Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2014)*, pages 153–160, St Petersburg, Russia.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*, pages 949–959, San Diego, California, USA.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2009)*, pages 406–414, Boulder, Colorado, USA.

M Heck, Q T Do, S Sakti, G Neubig, T Toda, and S Nakamura. 2015. The NAIST ASR system for IWSLT 2015. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2015)*, Da Nang, Vietnam.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON 2008)*, pages 20–27, Columbus, Ohio, USA.

Shahram Khadivi and Hermann Ney. 2008. Integration of speech recognition and machine translation in computer-assisted translation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(8):1551–1564.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. On the integration of speech recognition and statistical machine translation. In *6th Interspeech 2005 and 9th European Conference on Speech Communication and Technology (INTERSPEECH 2005)*, pages 3177–3180, Lisbon, Portugal.

Coskun Mermer, Murat Saraçlar, and Ruhi Sarikaya. 2013. Improving statistical machine translation using Bayesian word alignment and Gibbs sampling. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(5):1090–1101.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2 (ACL HLT 2011)*, pages 529–533, Portland, Oregon, USA.

Graham Neubig, Masato Mimura, and Tatsuya Kawahara. 2012. Bayesian learning of a language model from continuous speech. *IEICE TRANSACTIONS on Information and Systems*, 95(2):614–625.

ThuyLinh Nguyen, Stephan Vogel, and Noah A Smith. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 815–823, Beijing, China.

Matthias Paulik and Alex Waibel. 2013. Training speech translation from audio recordings of interpreter-mediated communication. *Computer Speech & Language*, 27(2):455–474.

Joris Pelemans, Tom Vanallemeersch, Kris Demuynck, Patrick Wambacq, and Others. 2015. Efficient language model adaptation for automatic speech recognition of spoken translations. In *16th Annual Conference of the International Speech Communication Association (INTERSPEECH 2015)*, pages 2262–2266, Dresden, Germany.

Aarthi Reddy and Richard C Rose. 2010. Integration of statistical models for dictation of document translations in a machine-aided human translation task. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(8):2015–2027.

D Will Reiman. 2010. Basic oral language documentation. In *Language Documentation & Conservation*, pages 254–268.

Steven L Scott. 2002. Bayesian methods for hidden Markov models. *Journal of the American Statistical Association*, pages 337–351.

Felix Stahlberg, Tim Schlippe, Sue Vogel, and Tanja Schultz. 2012. Word segmentation through cross-lingual word-to-phoneme alignment. In *2012 IEEE Workshop on Spoken Language Technology (SLT 2012)*, pages 85–90, Miami, Florida, USA.

Felix Stahlberg, Tim Schlippe, Stephan Vogel, and Tanja Schultz. 2014. Word segmentation and pronunciation extraction from phoneme sequences through cross-lingual word-to-phoneme alignment. *Computer Speech & Language*, pages 234–261.

Sebastian Stüker, Laurent Besacier, and Alex Waibel. 2009. Human translations guided language discovery for ASR systems. In *10th Annual Conference of the International Speech Communication Association (INTERSPEECH 2009)*, pages 3023–3026, Brighton, United Kingdom.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 147–152, Las Palmas, Canary Islands.

Enrique Vidal, Francisco Casacuberta, Luis Rodriguez, Jorge Civera, and Carlos D Martínez Hinarejos. 2006. Computer-assisted translation using speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(3):941–951.

L I Zezhong, Hideto Ikeda, and Junichi Fukumoto. 2013. Bayesian word alignment and phrase table training for statistical machine translation. *IEICE TRANSACTIONS on Information and Systems*, 96(7):1536–1543.

SQuAD: 100,000+ Questions for Machine Comprehension of Text

Pranav Rajpurkar and **Jian Zhang** and **Konstantin Lopyrev** and **Percy Liang**
{pranavsr,zjian,klopyrev,pliang}@cs.stanford.edu
Computer Science Department
Stanford University

Abstract

We present the Stanford Question Answering Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. We analyze the dataset to understand the types of reasoning required to answer the questions, leaning heavily on dependency and constituency trees. We build a strong logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research. The dataset is freely available at https://stanford-qa.com.

1 Introduction

Reading Comprehension (RC), or the ability to read text and then answer questions about it, is a challenging task for machines, requiring both understanding of natural language and knowledge about the world. Consider the question "*what causes precipitation to fall?*" posed on the passage in Figure 1. In order to answer the question, one might first locate the relevant part of the passage "*precipitation ... falls under gravity*", then reason that "*under*" refers to a cause (not location), and thus determine the correct answer: "*gravity*".

How can we get a machine to make progress on the challenging task of reading comprehension? Historically, large, realistic datasets have played a critical role for driving fields forward—famous examples include ImageNet for object recognition (Deng et al., 2009) and the Penn Treebank for syntactic parsing (Marcus et al., 1993). Existing datasets for RC have one of two shortcomings: (i) those that are high in quality (Richardson et al., 2013; Berant et al., 2014) are too small for training modern data-intensive models, while (ii) those that are large (Hermann et al., 2015; Hill et al., 2015) are semi-synthetic and do not share the same characteristics as explicit reading comprehension questions.

To address the need for a large and high-quality reading comprehension dataset, we present the Stan-

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2383–2392,
Austin, Texas, November 1-5, 2016.

ford Question Answering Dataset v1.0 (SQuAD), freely available at https://stanford-qa.com, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or *span*, from the corresponding reading passage. SQuAD contains 107,785 question-answer pairs on 536 articles, and is almost two orders of magnitude larger than previous manually labeled RC datasets such as MCTest (Richardson et al., 2013).

In contrast to prior datasets, SQuAD does not provide a list of answer choices for each question. Rather, systems must select the answer from all possible spans in the passage, thus needing to cope with a fairly large number of candidates. While questions with span-based answers are more constrained than the more interpretative questions found in more advanced standardized tests, we still find a rich diversity of questions and answer types in SQuAD. We develop automatic techniques based on distances in dependency trees to quantify this diversity and stratify the questions by difficulty. The span constraint also comes with the important benefit that span-based answers are easier to evaluate than free-form answers.

To assess the difficulty of SQuAD, we implemented a logistic regression model with a range of features. We find that lexicalized and dependency tree path features are important to the performance of the model. We also find that the model performance worsens with increasing complexity of (i) answer types and (ii) syntactic divergence between the question and the sentence containing the answer; interestingly, there is no such degradation for humans. Our best model achieves an F1 score of 51.0%,[1] which is much better than the sliding window baseline (20%). Over the last four months (since June 2016), we have witnessed significant improvements from more sophisticated neural network-based models. For example, Wang and Jiang (2016) obtained 70.3% F1 on SQuAD v1.1 (results on v1.0 are similar). These results are still well behind human performance, which is 86.8% F1 based on inter-annotator agreement. This suggests that there is plenty of room for advancement in modeling and learning on the SQuAD dataset.

[1] All experimental results in this paper are on SQuAD v1.0.

Dataset	Question source	Formulation	Size
SQuAD	**crowdsourced**	**RC, spans in passage**	**100K**
MCTest (Richardson et al., 2013)	crowdsourced	RC, multiple choice	2640
Algebra (Kushman et al., 2014)	standardized tests	computation	514
Science (Clark and Etzioni, 2016)	standardized tests	reasoning, multiple choice	855
WikiQA (Yang et al., 2015)	query logs	IR, sentence selection	3047
TREC-QA (Voorhees and Tice, 2000)	query logs + human editor	IR, free form	1479
CNN/Daily Mail (Hermann et al., 2015)	summary + cloze	RC, fill in single entity	1.4M
CBT (Hill et al., 2015)	cloze	RC, fill in single word	688K

Table 1: A survey of several reading comprehension and question answering datasets. SQuAD is much larger than all datasets except the semi-synthetic cloze-style datasets, and it is similar to TREC-QA in the open-endedness of the answers.

2 Existing Datasets

We begin with a survey of existing reading comprehension and question answering (QA) datasets, highlighting a variety of task formulation and creation strategies (see Table 1 for an overview).

Reading comprehension. A data-driven approach to reading comprehension goes back to Hirschman et al. (1999), who curated a dataset of 600 real 3rd–6th grade reading comprehension questions. Their pattern matching baseline was subsequently improved by a rule-based system (Riloff and Thelen, 2000) and a logistic regression model (Ng et al., 2000). More recently, Richardson et al. (2013) curated MCTest, which contains 660 stories created by crowdworkers, with 4 questions per story and 4 answer choices per question. Because many of the questions require commonsense reasoning and reasoning across multiple sentences, the dataset remains quite challenging, though there has been noticeable progress (Narasimhan and Barzilay, 2015; Sachan et al., 2015; Wang et al., 2015). Both curated datasets, although real and difficult, are too small to support very expressive statistical models.

Some datasets focus on deeper reasoning abilities. Algebra word problems require understanding a story well enough to turn it into a system of equa-

tions, which can be easily solved to produce the answer (Kushman et al., 2014; Hosseini et al., 2014). BAbI (Weston et al., 2015), a fully synthetic RC dataset, is stratified by different types of reasoning required to solve each task. Clark and Etzioni (2016) describe the task of solving 4th grade science exams, and stress the need to reason with world knowledge.

Open-domain question answering. The goal of open-domain QA is to answer a question from a large collection of documents. The annual evaluations at the Text REtreival Conference (TREC) (Voorhees and Tice, 2000) led to many advances in open-domain QA, many of which were used in IBM Watson for Jeopardy! (Ferrucci et al., 2013). Recently, Yang et al. (2015) created the WikiQA dataset, which, like SQuAD, use Wikipedia passages as a source of answers, but their task is sentence selection, while ours requires selecting a specific span in the sentence.

Selecting the span of text that answers a question is similar to *answer extraction*, the final step in the open-domain QA pipeline, methods for which include bootstrapping surface patterns (Ravichandran and Hovy, 2002), using dependency trees (Shen and Klakow, 2006), and using a factor graph over multiple sentences (Sun et al., 2013). One key difference between our RC setting and answer extraction is that answer extraction typically exploits the fact that the answer occurs in multiple documents (Brill et al., 2002), which is more lenient than in our setting, where a system only has access to a single reading passage.

Cloze datasets. Recently, researchers have constructed cloze datasets, in which the goal is to predict the missing word (often a named entity) in a passage. Since these datasets can be automatically generated from naturally occurring data, they can be extremely large. The Children's Book Test (CBT) (Hill et al., 2015), for example, involves predicting a blanked-out word of a sentence given the 20 previous sentences. Hermann et al. (2015) constructed a corpus of cloze style questions by blanking out entities in abstractive summaries of CNN / Daily News articles; the goal is to fill in the entity based on the original article. While the size of this dataset is impressive, Chen et al. (2016) showed that the dataset requires less reasoning than previously thought, and concluded that performance is almost saturated.

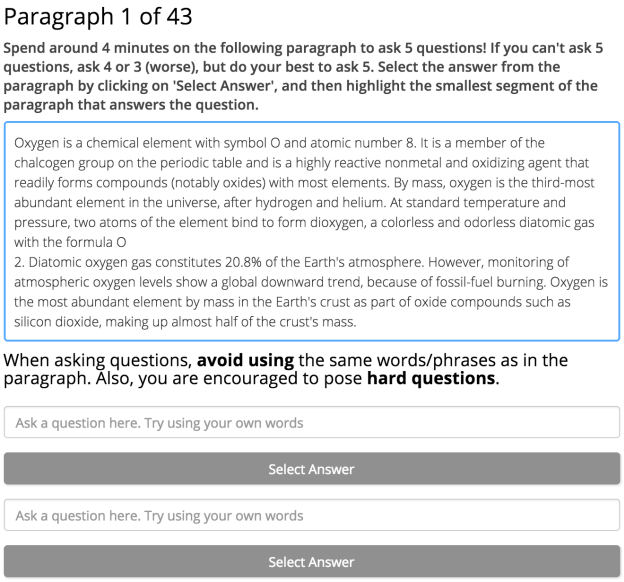


Figure 2: The crowd-facing web interface used to collect the dataset encourages crowdworkers to use their own words while asking questions.

One difference between SQuAD questions and cloze-style queries is that answers to cloze queries are single words or entities, while answers in SQuAD often include non-entities and can be much longer phrases. Another difference is that SQuAD focuses on questions whose answers are entailed by the passage, whereas the answers to cloze-style queries are merely suggested by the passage.

3 Dataset Collection

We collect our dataset in three stages: curating passages, crowdsourcing question-answers on those passages, and obtaining additional answers.

Passage curation. To retrieve high-quality articles, we used Project Nayuki's Wikipedia's internal PageRanks to obtain the top 10000 articles of English Wikipedia, from which we sampled 536 articles uniformly at random. From each of these articles, we extracted individual paragraphs, stripping away images, figures, tables, and discarding paragraphs shorter than 500 characters. The result was 23,215 paragraphs for the 536 articles covering a wide range of topics, from musical celebrities to abstract concepts. We partitioned the articles randomly into a training set (80%), a development set (10%),

and a test set (10%).

Question-answer collection. Next, we employed crowdworkers to create questions. We used the Daemo platform (Gaikwad et al., 2015), with Amazon Mechanical Turk as its backend. Crowdworkers were required to have a 97% HIT acceptance rate, a minimum of 1000 HITs, and be located in the United States or Canada. Workers were asked to spend 4 minutes on every paragraph, and paid $9 per hour for the number of hours required to complete the article. The task was reviewed favorably by crowdworkers, receiving positive comments on Turkopticon.

On each paragraph, crowdworkers were tasked with asking and answering up to 5 questions on the content of that paragraph. The questions had to be entered in a text field, and the answers had to be highlighted in the paragraph. To guide the workers, tasks contained a sample paragraph, and examples of good and bad questions and answers on that paragraph along with the reasons they were categorized as such. Additionally, crowdworkers were encouraged to ask questions in their own words, without copying word phrases from the paragraph. On the interface, this was reinforced by a reminder prompt at the beginning of every paragraph, and by disabling copy-paste functionality on the paragraph text.

Additional answers collection. To get an indication of human performance on SQuAD and to make our evaluation more robust, we obtained at least 2 additional answers for each question in the development and test sets. In the secondary answer generation task, each crowdworker was shown only the questions along with the paragraphs of an article, and asked to select the shortest span in the paragraph that answered the question. If a question was not answerable by a span in the paragraph, workers were asked to submit the question without marking an answer. Workers were recommended a speed of 5 questions for 2 minutes, and paid at the same rate of $9 per hour for the number of hours required for the entire article. Over the development and test sets, 2.6% of questions were marked unanswerable by at least one of the additional crowdworkers.

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quietly

Table 2: We automatically partition our answers into the following categories. Our dataset consists of large number of answers beyond proper noun entities.

4 Dataset Analysis

To understand the properties of SQuAD, we analyze the questions and answers in the development set. Specifically, we explore the (i) diversity of answer types, (ii) the difficulty of questions in terms of type of reasoning required to answer them, and (iii) the degree of syntactic divergence between the question and answer sentences.

Diversity in answers. We automatically categorize the answers as follows: We first separate the numerical and non-numerical answers. The non-numerical answers are categorized using constituency parses and POS tags generated by Stanford CoreNLP. The proper noun phrases are further split into person, location and other entities using NER tags. In Table 2, we can see dates and other numbers make up 19.8% of the data; 32.6% of the answers are proper nouns of three different types; 31.8% are common noun phrases answers; and the remaining 15.8% are made up of adjective phrases, verb phrases, clauses and other types.

Reasoning required to answer questions. To get a better understanding of the reasoning required to answer the questions, we sampled 4 questions from each of the 48 articles in the development set, and then manually labeled the examples with the categories shown in Table 3. The results show that all examples have some sort of lexical or syntactic divergence between the question and the answer in the passage. Note that some examples fall into more than one category.

Reasoning	Description	Example	Percentage
Lexical variation (synonymy)	Major correspondences between the question and the answer sentence are synonyms.	Q: What is the Rankine cycle sometimes **called**? Sentence: The Rankine cycle is sometimes **referred** to as a practical Carnot cycle.	33.3%
Lexical variation (world knowledge)	Major correspondences between the question and the answer sentence require world knowledge to resolve.	Q: Which **governing bodies** have veto power? Sen.: **The European Parliament and the Council of the European Union** have powers of amendment and veto during the legislative process.	9.1%
Syntactic variation	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence even after local modifications.	Q: What Shakespeare scholar **is currently on the faculty**? Sen.: **Current faculty include** the anthropologist Marshall Sahlins, ..., Shakespeare scholar David Bevington.	64.1%
Multiple sentence reasoning	There is anaphora, or higher-level fusion of multiple sentences is required.	Q: What collection does **the V&A Theatre & Performance galleries** hold? Sen.: **The V&A Theatre & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of material about live performance.	13.6%
Ambiguous	We don't agree with the crowdworkers' answer, or the question does not have a unique answer.	Q: What is the main goal of criminal punishment? Sen.: **Achieving crime control via incapacitation and deterrence** is a major goal of criminal punishment.	6.1%

Table 3: We manually labeled 192 examples into one or more of the above categories. Words relevant to the corresponding reasoning type are bolded, and the crowdsourced answer is underlined.

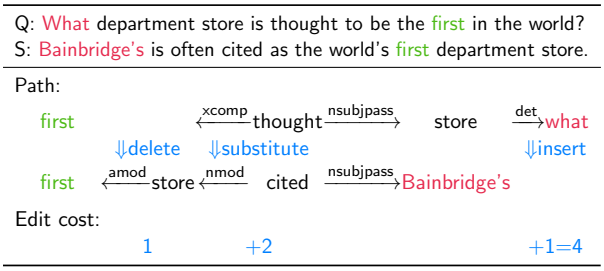


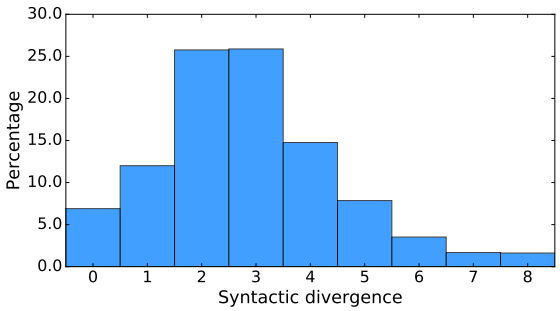
Figure 3: An example walking through the computation of the syntactic divergence between the question Q and answer sentence S.

Stratification by syntactic divergence. We also develop an automatic method to quantify the syntactic divergence between a question and the sentence containing the answer. This provides another way to measure the difficulty of a question and to stratify the dataset, which we return to in Section 6.3.

We illustrate how we measure the divergence with the example in Figure 3. We first detect anchors (word-lemma pairs common to both the question and answer sentences); in the example, the anchor is "*first*". The two unlexicalized paths, one from the anchor "*first*" in the question to the wh-word "*what*", and the other from the anchor in the answer sentence and to the answer span "*Bainbridge's*", are then extracted from the dependency parse trees. We measure the edit distance between these two paths, which we define as the minimum number of deletions or insertions to transform one path into the other. The syntactic divergence is then defined as the minimum edit distance over all possible anchors. The histogram in Figure 4a shows that there is a wide range of syntactic divergence in our dataset. We also show a concrete example where the edit distance is 0 and another where it is 6. Note that our syntactic divergence ignores lexical variation. Also, small divergence does not mean that a question is easy since there could be other candidates with similarly small divergence.

5 Methods

We developed a logistic regression model and compare its accuracy with that of three baseline methods.



(a) Histogram of syntactic divergence.

Q: Who went to Wittenberg to hear Luther speak?
S: Students thronged to Wittenberg to hear Luther speak.
Path:

Wittenberg $\xleftarrow{\text{nmod}}$ went $\xrightarrow{\text{nsubj}}$ Who
Wittenberg $\xleftarrow{\text{nmod}}$ thronged $\xrightarrow{\text{nsubj}}$ Students

(b) An example of a question-answer pair with edit distance 0 between the dependency paths (note that lexical variation is ignored in the computation of edit distance).

Q: What impact did the high school education movement have on the presence of skilled workers?
S: During the mass high school education movement from 1910 – 1940 , there was an increase in skilled workers.
Path:

school $\xleftarrow{\text{compound}}$ movement $\xleftarrow{\text{nsubj}}$ have $\xrightarrow{\text{dobj}}$ impact $\xrightarrow{\text{det}}$ What
school $\xleftarrow{\text{compound}}$ movement $\xrightarrow{\text{nmod}}$ 1910 $\xrightarrow{\text{acl}}$ was $\xrightarrow{\text{nsubj}}$ increase

(c) An example of a question-answer pair with edit distance 6.

Figure 4: We use the edit distance between the unlexicalized dependency paths in the question and the sentence containing the answer to measure *syntactic divergence*.

Candidate answer generation. For all four methods, rather than considering all $O(L^2)$ spans as candidate answers, where L is the number of words in the sentence, we only use spans which are constituents in the constituency parse generated by Stanford CoreNLP. Ignoring punctuation and articles, we find that 77.3% of the correct answers in the development set are constituents. This places an effective ceiling on the accuracy of our methods. During training, when the correct answer of an example is not a constituent, we use the shortest constituent containing the correct answer as the target.

5.1 Sliding Window Baseline

For each candidate answer, we compute the unigram/bigram overlap between the sentence containing it (excluding the candidate itself) and the question. We keep all the candidates that have the maximal overlap. Among these, we select the best one using the sliding-window approach proposed in Richardson et al. (2013).

In addition to the basic sliding window approach, we also implemented the distance-based extension (Richardson et al., 2013). Whereas Richardson et al. (2013) used the entire passage as the context of an answer, we used only the sentence containing the candidate answer for efficiency.

5.2 Logistic Regression

In our logistic regression model, we extract several types of features for each candidate answer. We discretize each continuous feature into 10 equally-sized buckets, building a total of 180 million features, most of which are lexicalized features or dependency tree path features. The descriptions and examples of the features are summarized in Table 4.

The matching word and bigram frequencies as well as the root match features help the model pick the correct sentences. Length features bias the model towards picking common lengths and positions for answer spans, while span word frequencies bias the model against uninformative words. Constituent label and span POS tag features guide the model towards the correct answer types. In addition to these basic features, we resolve lexical variation using lexicalized features, and syntactic variation using dependency tree path features.

The multiclass log-likelihood loss is optimized using AdaGrad with an initial learning rate of 0.1. Each update is performed on the batch of all questions in a paragraph for efficiency, since they share the same candidates. L_2 regularization is used, with a coefficient of 0.1 divided by the number of batches. The model is trained with three passes over the train-

Feature Groups	Description	Examples
Matching Word Frequencies	Sum of the TF-IDF of the words that occur in both the question and the sentence containing the candidate answer. Separate features are used for the words to the left, to the right, inside the span, and in the whole sentence.	Span: $[0 \leq \text{sum} < 0.01]$ Left: $[7.9 \leq \text{sum} < 10.7]$
Matching Bigram Frequencies	Same as above, but using bigrams. We use the generalization of the TF-IDF described in Shirakawa et al. (2015).	Span: $[0 \leq \text{sum} < 2.4]$ Left: $[0 \leq \text{sum} < 2.7]$
Root Match	Whether the dependency parse tree roots of the question and sentence match, whether the sentence contains the root of the dependency parse tree of the question, and whether the question contains the root of the dependency parse tree of the sentence.	Root Match = False
Lengths	Number of words to the left, to the right, inside the span, and in the whole sentence.	Span: $[1 <= \text{num} < 2]$ Left: $[15 \leq \text{num} < 19]$
Span Word Frequencies	Sum of the TF-IDF of the words in the span, regardless of whether they appear in the question.	Span: $[5.2 \leq \text{sum} < 6.9]$
Constituent Label	Constituency parse tree label of the span, optionally combined with the wh-word in the question.	Span: NP Span: NP, wh-word: "*what*"
Span POS Tags	Sequence of the part-of-speech tags in the span, optionally combined with the wh-word in the question.	Span: [NN] Span: [NN], wh-word: "*what*"
Lexicalized	Lemmas of question words combined with the lemmas of words within distance 2 to the span in the sentence based on the dependency parse trees. Separately, question word lemmas combined with answer word lemmas.	Q: "*cause*", S: "*under*" $\xleftarrow{\text{case}}$ Q: "*fall*", A: "*gravity*"
Dependency Tree Paths	For each word that occurs in both the question and sentence, the path in the dependency parse tree from that word in the sentence to the span, optionally combined with the path from the wh-word to the word in the question. POS tags are included in the paths.	VBZ $\xrightarrow{\text{nmod}}$ NN what $\xleftarrow{\text{nsubj}}$ VBZ $\xrightarrow{\text{advcl}}$ + VBZ$\xrightarrow{\text{nmod}}$NN

Table 4: Features used in the logistic regression model with examples for the question "*What causes precipitation to fall?*", sentence "*In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.*" and answer "*gravity*". Q denotes question, A denotes candidate answer, and S denotes sentence containing the candidate answer.

ing data.

6 Experiments

6.1 Model Evaluation

We use two different metrics to evaluate model accuracy. Both metrics ignore punctuations and articles (a, an, the).

Exact match. This metric measures the percentage of predictions that match any one of the ground truth answers exactly.

(Macro-averaged) F1 score. This metric measures the average overlap between the prediction and ground truth answer. We treat the prediction and ground truth as bags of tokens, and compute their F1. We take the maximum F1 over all of the ground truth answers for a given question, and then average over all of the questions.

6.2 Human Performance

We assess human performance on SQuAD's development and test sets. Recall that each of the questions in these sets has at least three answers. To evaluate human performance, we treat the second answer to each question as the human prediction, and keep the other answers as ground truth answers. The resulting human performance score on the test set is 77.0% for the exact match metric, and 86.8% for F1. Mismatch occurs mostly due to inclusion/exclusion of non-essential phrases (e.g., *monsoon trough* versus *movement of the monsoon trough*) rather than fundamental disagreements about the answer.

6.3 Model Performance

Table 5 shows the performance of our models alongside human performance on the v1.0 of development and test sets. The logistic regression model significantly outperforms the baselines, but underperforms

	Exact Match		F1	
	Dev	Test	Dev	Test
Random Guess	1.1%	1.3%	4.1%	4.3%
Sliding Window	13.2%	12.5%	20.2%	19.7%
Sliding Win. + Dist.	13.3%	13.0%	20.2%	20.0%
Logistic Regression	40.0%	40.4%	51.0%	51.0%
Human	80.3%	77.0%	90.5%	86.8%

Table 5: Performance of various methods and humans. Logistic regression outperforms the baselines, while there is still a significant gap between humans.

	F_1	
	Train	Dev
Logistic Regression	91.7%	51.0%
– Lex., – Dep. Paths	33.9%	35.8%
– Lexicalized	53.5%	45.4%
– Dep. Paths	91.4%	46.4%
– Match. Word Freq.	91.7%	48.1%
– Span POS Tags	91.7%	49.7%
– Match. Bigram Freq.	91.7%	50.3%
– Constituent Label	91.7%	50.4%
– Lengths	91.8%	50.5%
– Span Word Freq.	91.7%	50.5%
– Root Match	91.7%	50.6%

Table 6: Performance with feature ablations. We find that lexicalized and dependency tree path features are most important.

humans. We note that the model is able to select the sentence containing the answer correctly with 79.3% accuracy; hence, the bulk of the difficulty lies in finding the exact span within the sentence.

Feature ablations. In order to understand the features that are responsible for the performance of the logistic regression model, we perform a feature ablation where we remove one group of features from our model at a time. The results, shown in Table 6, indicate that lexicalized and dependency tree path features are most important. Comparing our analysis to the one in Chen et al. (2016), we note that the dependency tree path features play a much bigger role in our dataset. Additionally, we note that with lexicalized features, the model significantly overfits the training set; however, we found that increasing L_2 regularization hurts performance on the development set.

Performance stratified by answer type. To gain more insight into the performance of our logistic regression model, we report its performance across

	Logistic Regression Dev F1	Human Dev F1
Date	72.1%	93.9%
Other Numeric	62.5%	92.9%
Person	56.2%	95.4%
Location	55.4%	94.1%
Other Entity	52.2%	92.6%
Common Noun Phrase	46.5%	88.3%
Adjective Phrase	37.9%	86.8%
Verb Phrase	31.2%	82.4%
Clause	34.3%	84.5%
Other	34.8%	86.1%

Table 7: Performance stratified by answer types. Logistic regression performs better on certain types of answers, namely numbers and entities. On the other hand, human performance is more uniform.

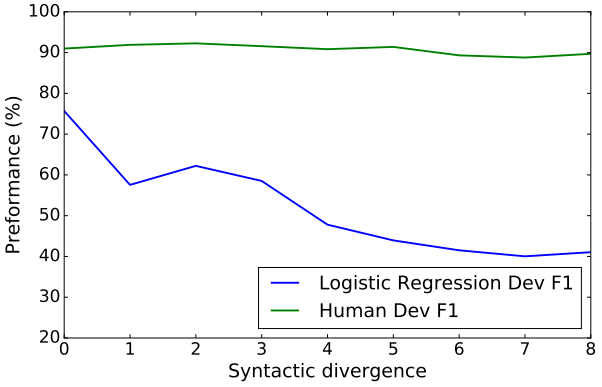


Figure 5: Performance stratified by syntactic divergence of questions and sentences. The performance of logistic regression degrades with increasing divergence. In contrast, human performance is stable across the full range of divergence.

the answer types explored in Table 2. The results (shown in Table 7) show that the model performs best on dates and other numbers, categories for which there are usually only a few plausible candidates, and most answers are single tokens. The model is challenged more on other named entities (i.e., location, person and other entities) because there are many more plausible candidates. However, named entities are still relatively easy to identify by their POS tag features. The model performs worst on other answer types, which together form 47.6% of the dataset. Humans have exceptional performance on dates, numbers and all named entities. Their performance on other answer types degrades only slightly.

Performance stratified by syntactic divergence. As discussed in Section 4, another challenging aspect of the dataset is the syntactic divergence between the question and answer sentence. Figure 5 shows that the more divergence there is, the lower the performance of the logistic regression model. Interestingly, humans do not seem to be sensitive to syntactic divergence, suggesting that deep understanding is not distracted by superficial differences. Measuring the degree of degradation could therefore be useful in determining the extent to which a model is generalizing in the right way.

7 Conclusion

Towards the end goal of natural language understanding, we introduce the Stanford Question Answering Dataset, a large reading comprehension dataset on Wikipedia articles with crowdsourced question-answer pairs. SQuAD features a diverse range of question and answer types. The performance of our logistic regression model, with 51.0% F1, against the human F1 of 86.8% suggests ample opportunity for improvement. We have made our dataset freely available to encourage exploration of more expressive models. Since the release of our dataset, we have already seen considerable interest in building models on this dataset, and the gap between our logistic regression model and human performance has more than halved (Wang and Jiang, 2016). We expect that the remaining gap will be harder to close, but that such efforts will result in significant advances in reading comprehension.

Reproducibility

All code, data, and experiments for this paper are available on the CodaLab platform: https://worksheets.codalab.org/worksheets/0xd53d03a48ef64b329c16b9baf0f99b0c/ .

Acknowledgments

We would like to thank Durim Morina and Professor Michael Bernstein for their help in crowdsourcing the collection of our dataset, both in terms of funding and technical support of the Daemo platform.

References

J. Berant, V. Srikumar, P. Chen, A. V. Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. 2014. Modeling biological processes for reading comprehension. In *Empirical Methods in Natural Language Processing (EMNLP)*.

E. Brill, S. Dumais, and M. Banko. 2002. An analysis of the AskMSR question-answering system. In *Association for Computational Linguistics (ACL)*, pages 257–264.

D. Chen, J. Bolton, and C. D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

P. Clark and O. Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of AI. *AI Magazine*, 37(1):5–12.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. 2013. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79.

S. N. Gaikwad, D. Morina, R. Nistala, M. Agarwal, A. Cossette, R. Bhanu, S. Savage, V. Narwal, K. Rajpal, J. Regino, et al. 2015. Daemo: A self-governed crowdsourcing marketplace. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 101–102.

K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

F. Hill, A. Bordes, S. Chopra, and J. Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

L. Hirschman, M. Light, E. Breck, and J. D. Burger. 1999. Deep read: A reading comprehension system. In *Association for Computational Linguistics (ACL)*, pages 325–332.

M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. 2014. Learning to automatically solve algebra word problems. In *Association for Computational Linguistics (ACL)*.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

K. Narasimhan and R. Barzilay. 2015. Machine comprehension with discourse relations. In *Association for Computational Linguistics (ACL)*.

H. T. Ng, L. H. Teo, and J. L. P. Kwan. 2000. A machine learning approach to answering questions for reading comprehension tests. In *Joint SIGDAT conference on empirical methods in natural language processing and very large corpora - Volume 13*, pages 124–132.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Association for Computational Linguistics (ACL)*, pages 41–47.

M. Richardson, C. J. Burges, and E. Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 193–203.

E. Riloff and M. Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *ANLP/NAACL Workshop on reading comprehension tests as evaluation for computer-based language understanding sytems - Volume 6*, pages 13–19.

M. Sachan, A. Dubey, E. P. Xing, and M. Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Association for Computational Linguistics (ACL)*.

D. Shen and D. Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, pages 889–896.

M. Shirakawa, T. Hara, and S. Nishio. 2015. N-gram idf: A global term weighting scheme based on information distance. In *World Wide Web (WWW)*, pages 960–970.

H. Sun, N. Duan, Y. Duan, and M. Zhou. 2013. Answer extraction from passage graph for question answering. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

E. M. Voorhees and D. M. Tice. 2000. Building a question answering test collection. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 200–207.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905.

H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Association for Computational Linguistics (ACL)*.

J. Weston, A. Bordes, S. Chopra, and T. Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv*.

Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2013–2018.

Author Index